Abstract

This guide describes how to install SmartSnippets™ Studio on your computer, and how to use it in combination with the SmartSnippets™ SDKs and RDs for Dialog Semiconductor SmartBond™ devices.
SmartSnippets™ Studio User Guide

6 Makefile Generation .................................................................................................................. 57
7 JTAG Connection ......................................................................................................................... 58
8 Dialog’s Code Formatter ............................................................................................................... 58
9 CMSIS File for DA1468x_00 and DA1469x Registers ................................................................. 59
10 Doxygen Documentation ........................................................................................................... 61
11 SEGGER Terminal Configuration .............................................................................................. 64
12 Serial Terminal View .................................................................................................................. 66
13 Launch SmartSnippets Toolbox as an External Tool ................................................................. 67
14 Advanced Settings .................................................................................................................... 68
15 Troubleshooting ......................................................................................................................... 69
   15.1 Project Errors while Builder is Still Active ........................................................................... 69
   15.2 Debug Session Launching Stops without any Error Indication ............................................. 69
   15.3 Linux Installation Errors ...................................................................................................... 69
   15.4 Debugger GDB Crash .......................................................................................................... 69
   15.5 Semihosting ......................................................................................................................... 70
   15.6 Debugger Silent Mode Issue ............................................................................................... 70
   15.7 Indexer ................................................................................................................................. 70
   15.8 JLink ...................................................................................................................................... 71
   15.9 Wrong Proposed Workspace Path ....................................................................................... 71
   15.10 Manual Plug-In Updates ..................................................................................................... 71
   15.11 Internal Browser does not Start after a Plug-In Addition or Update .................................... 71
   15.12 Unhandled Event Loop Exception when Opening a Project ............................................ 71
   15.13 Couldn’t Reserve Space for Cygwin’s Heap when Building a Project ................................. 71
   15.14 Application Crashes or Buttons and Dialogs not Responding on Linux with GTK3 .......... 72
   15.15 Errors when Running erase_qspi_jtag_win and suota_initial_flash_jtag_win scripts ....... 73
   15.16 Errors after Importing a Project .......................................................................................... 73
   15.17 Errors after Renaming a Project .......................................................................................... 74
   15.18 Errors after Copying a Project ............................................................................................. 74
   15.19 Error: BAD ELF Interpreter: No such File or Directory. Make Error 126 or 127 on Linux... 74
   15.20 Could not Determine GDB Version on Linux ..................................................................... 74
   15.21 Blank Welcome Screen when Opening SmartSnippets™ Studio .................................... 74
   15.22 Error on awk: “function strtonum never defined” when Running External Script .......... 75
   15.23 SmartSnippets™ Studio Error: Cannot Register Python Interpreter .................................. 75
   15.24 CentOS 7 and Ozone J-Link Debugger ................................................................................ 75
   15.25 Python Interpreter Error on Linux: ImportError: libtk8.6.so: Cannot Open Shared Object File: No such File or Directory ......................................................................................... 75
   15.26 Error when Starting SmartSnippets™ Studio Application ................................................ 76
   15.27 Error when Connecting DA1468x or DA1469x Family Chip with JTAG on Linux ............ 76
   15.28 Warning on Linux about Missing libqtgui4 ...................................................................... 76
   15.29 Build Error when Workspace Path Contains Spaces ....................................................... 77

Revision History ............................................................................................................................. 78
Figures

Figure 1: SmartSnippets™ Studio Install Link ................................................................. 9
Figure 2: Optionally Uninstall Previous Version ............................................................. 10
Figure 3: Segger JLink GDB Server Installation Directory ............................................. 10
Figure 4: Segger JLink GDB Server Installation Directory with Bundled GDB Server ....... 11
Figure 5: SmartSnippets™ Studio Installation Directory ............................................... 11
Figure 6: Automatically Install the J-Link ...................................................................... 13
Figure 7: Select SmartSnippets™ Studio Install Directory ............................................. 13
Figure 8: Allow Launching on Ubuntu 20.04 ................................................................. 14
Figure 9: Uninstall GNU ARM GCC ............................................................................. 14
Figure 10: Uninstall Shortcut on Linux ........................................................................... 15
Figure 11: The SDK Tools Installer Wizard Intro Page ............................................... 18
Figure 12: Installation of Segger Ozone by the SDK Tools Installer ............................... 18
Figure 13: Installation of Segger J-Link Software Components by SDK Tools Installer ... 19
Figure 14: Skipping Segger SystemView Installation in SDK Tools Installer .................. 19
Figure 15: SDK Tools Installer Wizard Summary Page ................................................. 20
Figure 16: SDK Tools Installer Wizard Preference Page ............................................... 20
Figure 17: SmartSnippets™ Toolbox Page in SDK Tools Installer ................................. 21
Figure 18: Installation path of SmartSnippets™ Toolbox is Verified by SDK Tools Installer 22
Figure 19: Change Global or Workspace Toolchain ...................................................... 22
Figure 20: Change Project Toolchain .......................................................................... 23
Figure 21: No SDK Selected Error in Welcome Page .................................................... 24
Figure 22: SDK Welcome Page .................................................................................... 25
Figure 23: More Options on SDK Welcome Page ......................................................... 26
Figure 24: Switch Workspace Verification .................................................................... 26
Figure 25: SDK Finder Select Root Directory to Search .............................................. 27
Figure 26: SDK Finder Active Search Indicator ............................................................. 27
Figure 27: SDK Finder Results .................................................................................... 27
Figure 28: Part Number Info Dialog Box ...................................................................... 28
Figure 29: SDK Inspector Icon .................................................................................... 29
Figure 30: SDK Inspector Scan Process ....................................................................... 29
Figure 31: SDK Inspector Project Table ........................................................................ 30
Figure 32: SDK Inspector Project Details of Eclipse Project ......................................... 31
Figure 33: Select Build Configuration to Open ............................................................. 31
Figure 34: SDK Inspector. Confirm Open Project in IDE ............................................ 32
Figure 35: SDK Inspector Selected Project and Build Configuration ............................. 32
Figure 36: Import a Project .......................................................................................... 33
Figure 37: No Family Selected ..................................................................................... 34
Figure 38: Blinky Project Directory .............................................................................. 35
Figure 39: Blinky Project Keil Workspace ..................................................................... 36
Figure 40: Blinky Project Options ............................................................................... 36
Figure 41: Blinky Project Scatter File .......................................................................... 37
Figure 42: Blinky Project: Debug Option ..................................................................... 37
Figure 43: Blinky Project: Jlink Setup ......................................................................... 38
Figure 44: Blinky Project: Project Building ................................................................. 38
Figure 45: Device Manager Ports ................................................................................ 39
Figure 46: Blinky Project: Start Debug Session ............................................................ 39
Figure 47: Keil Lite Pop Up Window ............................................................................. 40
Figure 48: Blinky Project: Code Execution .................................................................. 40
Figure 49: Blinky Project: Blinky Message on Terminal .............................................. 40
Figure 50: Importing a Project into the Eclipse Workspace .......................................... 41
Figure 51: Browse in Root Directory to Import a Project .............................................. 41
Figure 52: Browse and Select pxp_reporter ................................................................. 42
Figure 53: Pxp_reporter Has Been Imported ............................................................... 42
Figure 54: Create New Debug Configuration .............................................................. 43
Figure 55: New SmartBond Debug Configuration ....................................................... 43
Figure 56: Debug Configuration Setup ......................................................................... 44
Figure 57: Confirm Perspective Switch Screen ............................................................ 44
SmartSnippets™ Studio User Guide

Figure 58: Debug Perspective .......................................................... 44
Figure 59: Building with Debug Options ........................................ 45
Figure 60: xxx.elf File in Debug Folder ......................................... 45
Figure 61: Global Debug Configurations when no Project is Currently Selected ........................................................................ 46
Figure 62: Available Build Configurations ........................................ 47
Figure 63: Example of Hidden RAM Debug Configuration for a Project ...................................................................................... 47
Figure 64: Import my_project Project in SmartSnippets™ Studio .... 49
Figure 65: Change to Appropriate Name .......................................... 49
Figure 66: Match the ‘SDKROOT’ to the Root File of the SDK ........... 50
Figure 67: Edit the Right Path for “SDKROOT” ................................. 50
Figure 68: Fix the Invalid Variables ............................................... 50
Figure 69: Setup the Path for the Libraries ...................................... 51
Figure 70: Copy a Project. Step 1 .................................................... 51
Figure 71: Copy a Project. Step 2 .................................................... 52
Figure 72: Copy Project Dialog Box ............................................... 52
Figure 73: Folder for Copied Project Should be at the Same Location as the Original ................................................................. 52
Figure 74: Correct Prebuild Steps Command for Enabling Parallel Build ............................................................. 53
Figure 75: Behavior Tab to Enable Make’s Parallel Build ................ 54
Figure 76: Importing a Project into the Eclipse Workspace .............. 55
Figure 77: Browse in Root Directory to Import a Project ................. 55
Figure 78: Browse and Select Scripts .............................................. 56
Figure 79: Scripts ........................................................................ 56
Figure 80: Configure QSPI ............................................................... 57
Figure 81: Selecting the JTAG Connection ....................................... 58
Figure 82: Selecting CMSIS file for DA1468x_00 Registers ............ 59
Figure 83: The EmbSys Registers Screen ....................................... 60
Figure 84: Doxygen Wizard ............................................................ 61
Figure 85: Building Doxygen Documentation ................................ 61
Figure 86: Choosing which Doxygen File to Build ......................... 62
Figure 87: Project Explorer with Doxygen Generated Documentation ...................................................................................... 62
Figure 88: Editing File with Doxyfile Editor .................................... 63
Figure 89: Adding the Terminals View ............................................ 64
Figure 90: The Layout with the Terminal View Included ................. 64
Figure 91: The Terminal Settings Screen ........................................ 65
Figure 92: The RxTx Terminal View ............................................... 66
Figure 93: The RxTx Launch Terminal Window ............................. 66
Figure 94: An Example of the RxTx Terminal Window ................. 66
Figure 95: Create New Configuration ............................................ 67
Figure 96: New Configuration under External Tools ....................... 67
Figure 97: Errors/Warnings while Indexer is Still Running .......... 69
Figure 98: Selecting Semihosting Console .................................... 70
Figure 99: Error when Trying to Open a Project ............................. 71
Figure 100: Change Virtual Memory Settings ............................... 72
Figure 101: Switch Theme to Classic ............................................. 72
Figure 102: Could not Determine GDB Version ............................. 74
Figure 103: Unable to Register Python due to System Error ........... 75
Figure 104: Error when Starting Application ................................ 76
Figure 105: Error when Connecting DA1468x or DA1469x Chip ..... 76
Figure 106: Welcome Page SystemView Icon ............................... 76
Figure 107: Missing libQtGui4 Warning ......................................... 77

Tables

Table 1: Default Tool Installation .................................................... 17
SmartSnippets™ Studio User Guide

Terms and Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>Advanced RISC Machine</td>
</tr>
<tr>
<td>CDT</td>
<td>C/C++ Development Tools</td>
</tr>
<tr>
<td>CMSIS</td>
<td>Cortex Microcontroller Software Interface Standard</td>
</tr>
<tr>
<td>DK</td>
<td>Development Kit</td>
</tr>
<tr>
<td>Eclipse</td>
<td>Open Source IDE</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>GNU</td>
<td>General Public License</td>
</tr>
<tr>
<td>GDB</td>
<td>GNU Debugger</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>RD</td>
<td>Reference Design</td>
</tr>
<tr>
<td>SVD</td>
<td>System View Description</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>HDK</td>
<td>Hardware Development Kit</td>
</tr>
</tbody>
</table>

References

[2] SEGGER manual (online)
Prerequisites

- SmartSnippets™ Studio package (contact Dialog Semiconductor customer support)
- Dialog’s Semiconductor SmartSnippets™ SDKs (supported families DA14580/581/583, DA1453x, DA14585/586, DA1468x, DA1469x) or a Dialog Reference Design (contact Dialog Semiconductor customer support)
- Minimum hardware requirements:
  - Windows or Linux Operating System – 1 GHz, 32-bit or 64-bit processor
  - 1 GB of system memory (RAM)
  - 2 GB of available disk space
- Development Kit (Basic or Pro) for one of the supported families and accessories
- Serial-port terminal software (for example, RealTerm)
- A USB connection supporting USB-Serial (FTDI)
- Unzip program
- For Linux only: netstat utility is required for SmartSnippets™ Toolbox

About User Guide

This guide describes installation process and features of the SmartSnippets™ Studio software along the different components that platform supports.

SmartSnippets™ Studio is used with the SmartSnippets™ SDKs for DA14580/581/583, DA1453x, DA1585/586, DA1468x, DA1469x families and Reference Designs for SmartBond devices.

When the term <family> SDK is used in this manual, all devices from this family are meant.

It also applies to DA1468x-based RDs (Reference Designs)

Note: Novice users can also read one of the Getting Started Guides provided for different development kits, [DA14531 Pro kit, DA14531 Usb kit, DA14585/586 Basic kit, DA14585/586 Pro kit, DA1468x Pro kit, DA1469x Pro kit, DA14695 Usb kit]

Introduction

This document presents Dialog Semiconductor software package, SmartSnippets™ Studio.

SmartSnippets™ Studio is a royalty-free software development platform for SmartBond™ devices. It fully supports the DA14531, DA14580/581/583, DA1585/586, DA1468x and DA1469x family of devices.

Note: Some of the tools (such as KEIL, GCC, and so on) might need to get installed separately since they are not included by default.

SmartSnippets™ Studio is a framework of tools comes with the popular open source IDE Eclipse CDT.

The SmartSnippets™ IDE is enabled by an on-board J-Link debugger from SEGGER. This offers standard debug capabilities such as single stepping, setting breakpoints, software download and many more. For more details on the debugger capabilities, visit https://www.segger.com/.

All components, tools and plug-ins are taken from the internet without changes and modifications. A subset of useful tools/plug-ins is selected by Dialog, and these are bundled with Eclipse CDT. Users can interact with all different tools from within the Studio environment.

Availability in tools and features is directly dependent on the selected Dialog Device Family.

For the latest version of SmartSnippets™ Studio, contact Dialog Semiconductor customer support.
Supported Systems and Versions
SmartSnippets™ Studio is getting released with support for the following operating systems and versions:

- Windows
  - Win 10 64bit
- Linux
  - Ubuntu 20.04.1 LTS
  - Ubuntu 18.04.5 LTS
1 **SmartSnippets™ Installation**

1.1 Fresh install in Microsoft Windows

This section describes installation on Microsoft Windows systems for the first time.

Supported variants:

- Windows 10 64-bit

Disk space requirements:

- At least 2 GBytes of free disk space are needed to complete this installation

To proceed with installation, go through the following steps.

1. Get access to the SmartSnippets™ SDKs. Please visit product page in [https://support.dialog-semiconductor.com](https://support.dialog-semiconductor.com), download and unzip SDK zip-file, preferably under a subfolder of the home user’s directory (for example, under a subdirectory of `C:\Users\<user>`).

   It is recommended that the root directory of the selected SDK or RD is selected for creating SmartSnippets™ Studio workspaces, so that the C projects can easily reference the source code and libraries of the SmartSnippets SDK. Also, see the SDK release notes.

   **Note:** It is recommended that the pathnames of the folders used for unzipping the SDK zip files, to be as short as possible (less than 20 characters). This is important to prevent problems under Windows due to the maximum path length restriction (260 characters for the entire path).

2. Download the latest version of SmartSnippets™ Studio from product website, as in Figure 1.

3. Run SmartSnippets™ Studio installer (.msi). Several of the required tools are automatically installed while others need to be manually downloaded and installed.

4. If a previous version of SmartSnippets™ Studio is found, user can uninstall it first before proceeding to installation of current version. This may take some minutes.
5. Select to install recommended and latest version of SEGGER J-Link GDB server and click Next.

The installer will search for already installed version and will allow easily install of the recommended version. In case that there is an already bundled version with the installer, the prompt will be shown as in Figure 4. It is recommended to install the version which is bundled with the installer (if not already installed) by clicking the Next button.
6. Select the destination folder for the SmartSnippets™ Studio and click Next.
   Destination Folder is the installation folder for SmartSnippets™ Studio. Default is C:\DiaSemi\ for windows installation.

7. The SmartSnippets™ Studio is installed. To run the application, double-click the desktop shortcut or run from:
   Start > All programs > Dialog Semiconductor > SmartSnippets™ Studio
   {v<Studio_version>}

Note:
- The default Windows installation folder for SmartSnippets™ Studio is C:\DiaSemi.
- When Windows Defender is turned ON, it will prevent the file SmartSnippets™ Studio installer (.msi) from automatically running, click Run Anyway when prompted
- Administrator access is required to install SmartSnippets Studio software components
- During the installation, the user has to click the OK button several times
- Window hosting the OK button may be hidden behind the active window
1.2 Fresh install in Linux

This section describes installation on Linux systems for first time.

SmartSnippets™ Studio has been tested under:

- Ubuntu 20.04.1 LTS (64-bit x86_64)
- Ubuntu 18.04.5 LTS (64-bit x86_64)

**Note:** SmartSnippets™ Studio should work with the most modern 64-bit Linux distributions.

Disk space requirements:
- At least 2 GB of free disk space are needed to complete this installation

To proceed with installation, go through the following steps:

1. Families DA1468x and DA1469x need the `netstat` utility for communication with JTAG interface.
   - On Ubuntu install with:
     ```
     sudo apt-get install net-tools
     ```

2. To be able to compile native projects from the SDK on Linux, install the `gcc` compiler and the `make` utility. On Ubuntu install with:
   ```
   sudo apt-get install build-essential
   ```

3. SDK 1 requires GNU cross arm compiler 4.9-2015 which is a 32 bit package. To be able to run a 32 bit program on Ubuntu 64 the following packages need to be installed:
   ```
   sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386
   ```

4. Segger SystemView 2.xx referred by current SDK require `libqtgui4`. On Ubuntu 18.04 install with:
   ```
   sudo apt-get install libqtgui4
   ```
   SystemView 2.xx is not supported on Ubuntu 20.04. User is advisable to install its own version 3.xx from [https://www.segger.com/downloads/systemview/](https://www.segger.com/downloads/systemview/). See also <Troubleshooting> section.

5. The SmartSnippets™ SDK sources should be installed in a directory that gives read and write permissions to the user working with SmartSnippets™ Studio so that the user can work with the source files and the temporary files created during compilation/linking.

6. If a python project uses tkinter module, needs tcl/tk to be installed or else a missing library error occurs (see Troubleshooting section). There are two ways to resolve this issue:
   a. Install Tcl/Tk on the system:
      i. For Ubuntu
         ```
         sudo apt-get install tk tcl
         ```
   b. Download and install ActiveState ActiveTcl® ([https://www.activestate.com/activetcl/downloads](https://www.activestate.com/activetcl/downloads)) and update LD_LIBRARY_PATH to point to the lib directory (for example,
      ```
      LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/ActiveTcl-8.6/lib
      ```

7. To get access to the SmartSnippets™ SDKs, contact your Dialog sales representative. Download and unzip the SDK zip file, preferably under a subfolder of the home user's directory (for example, under a subdirectory of `/home/<user>/`).
   It is recommended that this folder also becomes the default folder for creating SmartSnippets™ Studio workspaces, so that the C projects can easily reference the source code and libraries of the SmartSnippets™ SDKs. Please also refer to the SDK release notes.

8. Run SmartSnippets™ Studio Installer that comes in the form of an executable file, having a '.run' extension. Modify its properties to make it executable:
   ```
   chmod +x SmartSnippets_Studio-linux.gtk.x86_64-X.X.X.XXX.run
   ```
   User may run installer as root with `sudo` (for example, `sudo ./SmartSnippets_Studio-linux.gtk.x86_64-X.X.X.XXX.run`) or as simple user.
   It is recommended to install as the user who owns the SmartSnippets™ SDK to make sure that user has all necessary permissions to automatically build SDK projects.

9. Select to install the latest version of SEGGER J-Link GDB server and click Next.
10. Select the destination folder for the SmartSnippets™ Studio and click **Next**.

Destination Folder on root installation default folder depends on Linux distribution (usually /usr/local) else default folder is user’s home directory.

11. The application can be executed from terminal by running:

```
<inst_dir>/Diasemi/SmartSnippetsStudio2.0.6/CDT/SmartSnippets_Studio.startup.sh
```

Also, .desktop files (shortcut files) are created in the following cases:

a. For installation as root user:

   `/usr/share/applications/smartsnippets_studio.desktop`

b. For installation as normal user:

   `/home/<user>/.local/share/applications/smartsnippets_studio.desktop`

These shortcuts should be visible from the window manager and the Desktop. Enable launching if needed. For example, on Ubuntu 20.04 right click on desktop shortcut and select “Allow launching” (Figure 8).
1.3 Reinstall/Uninstall under Windows

It is recommended to uninstall the previous version of SmartSnippets™ Studio before installing a new one. Also remove any remaining files and folders if installing under the same directory path. If not, the new version of SmartSnippets™ Studio will still try to uninstall the previous version before installing itself.

SmartSnippets™ Studio workspace will not be removed as long as it is not located inside the installation folder. So, workspaces can be used even after installing a newer version of the tool.

In cases there is some major change in how external paths are being treated internally, user may have to reconfigure project paths (for example, the JLink or SDK paths) again.

In general, Eclipse workspace preferences are maintained if user prefer working under the same workspace, even after upgrading to a new version of SmartSnippets™ Studio take place.

To uninstall SmartSnippets™ Studio from Windows, you can do it by executing the Uninstall SmartSnippets™ Studio located under Start > All programs > Dialog Semiconductor folder.

Note that after the uninstallation completes, the .metadata file is not deleted so that the workspaces that have been created in SmartSnippets™ Studio are still available.

During uninstallation process, user can select which of the currently installed GNU ARM GCC toolchains would like to remove.

Figure 8: Allow Launching on Ubuntu 20.04

Figure 9: Uninstall GNU ARM GCC
1.4 Reinstall/Uninstall under Linux

It is recommended to uninstall the previous version before reinstalling a more recent one under the same directory. Linux installer will not try to uninstall the previous version by itself.

To uninstall SmartSnippets™ Studio from Linux, user can search for the uninstaller shortcut (see Figure 10) or run:

```
/home/someuser/DiaSemi/SmartSnippetsStudio2.0.14/CDT/scripts/uninstall/uninstall_start up.sh
```

This should:

- Remove the installation files and folders (will ask first for folder)
- Remove file `/etc/profile.d/smartsnippets_studio.sh` if you installed as root or lines between `DIALOGECLIPSECONF_START` and `DIALOGECLIPSECONF_END` on your PATH settings file (`~/.bash_profile` or `~/.profile`) if you installed as user
- Remove the `SmartSnippets_desktop` file under `/usr/share/applications` for root installation or `/home/<user>/local/share/applications/` for normal user installation
- Remove file “`99-smartsnippets_studio_common_ftdi_devices.rules`” (if any) located under `/etc/udev/rules.d`
- Uninstaller will ask which GNU ARM GCC toolchains to remove (see Figure 9)

SmartSnippets™ Studio workspace will not be removed as long as it is not inside the installation folder. So, workspaces can be used even after installing a newer version of the tool.

If there is some major change in how external paths are being treated internally, user may have to reconfigure project paths (for example, the JLink or SDK paths) again.

In general, Eclipse workspace preferences are maintained if user prefer working under the same workspace, even after upgrading to a newer version of SmartSnippets™ Studio.
2 Starting SmartSnippets™ Studio

By starting the SmartSnippets™ Studio and after selecting a workspace, first thing that user will see is the welcome page. Nevertheless, the exact first time that IDE starts this might not be the actual case since some configuration needs to take place first.

No Toolchain included

Note that no toolchain is included by default with the provided installers. This is applicable for all platform installers. Furthermore, once a workspace gets connected with an SDK which requires a toolchain as a dependency, then a tool called “SDK Wizard” will come forward to guide the user in order to install the required dependencies. See Section 2.3 for more details.

2.1 Select Workspace

When SmartSnippets™ Studio starts for the first time, it prompts for a workspace selection under user’s profile folder (for example, under C:\Users\<user>\workspace_without_SDK for windows).

User must download and unzip an SDK to work with SmartSnippets Studio. Extract to a path that do not contain spaces. See also Section 15.29.

The workspace must be the root directory of the downloaded and unzipped SDK. The root directory contains folders named “sdk” and “projects”, and a folder named “config”, that includes a file like “config.xml”.

SmartSnippets™ Studio stores your projects in a folder called workspace. User can always change the targeted workspace from the top menu (File > Switch workspace) or through the SmartSnippets™ Studio Welcome Page (see Figure 22 for point 3).

**Important Note:** When IDE loads the workspace, it tries to locate also an SDK under the specified path. For that reason, users **should always export** the root directory of the SDK they want to work with under the selected workspace path. This ensures that when SmartSnippets™ Studio try to open the workspace will be able to read necessary configuration settings and will also prepare appropriately the environment for this SDK.

In different case, SmartSnippets™ Studio will not be able to read the configuration files and it **will not set up the necessary tools** and configuration parameters for working under the context of an SDK. Please also note that the ‘API documentation’ button will not be functional.

In case workspace folder does not correspond to a valid SDK root folder or corresponds to one of the older SDKs supported by SmartSnippets™ Studio (namely DA14580/581/583 5.0.3 SDK or DA1468x 1.0.4 SDK), SmartSnippets™ Studio welcome page will show a message like in Figure 21.

This means that a predefined set of tools, URLs and buttons will get by default enabled, but the overall installation/configuration will not be necessarily compatible with what is needed for this SDK to work appropriately under the specified workspace.

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is better that path to workspace should not contain spaces or else some projects may have build errors. To ensure that when the SmartSnippets™ Studio launches for an SDK all appropriate tools and settings are in place, it reads the SDK configuration files and sets up the appropriate environment. This means that if the user modifies some of the preferences (for example, Windows &gt; Preferences &gt; MCU &gt; Global SEGGER JLink Path) that reside in the configuration files too, next time the workspace starts these user preferences are read again from the SDK configuration file and will get overwritten. If for any reason (highly not recommended) the user wants to override the default configuration parameters for an SDK, the following steps should be followed: 1. Rename the <code>&lt;sdk_root&gt;\config\SDK_config.xml</code> file by adding a prefix to the filename and placing the new file under the same folder (<code>&lt;sdk_root&gt;\config</code>). 2. Modify the copied <code>SDK_config.xml</code> file. 3. Restart SmartSnippets™ Studio. The following preferences are reset every time Studio launches:</td>
</tr>
</tbody>
</table>
2.2 Supported SDKs

From version 2.0.10 the following SDKs are supported:

- SDK5: version 5.0.4 or newer
- SDK6: all versions
- SDK1: version 1.0.6 or newer
- SDK10: all versions

2.3 SDK Tools Installer

When the user runs the SmartSnippets™ Studio installer, a set of tools required for all SDKs are getting automatically installed.

Table 1: Default Tool Installation

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Version for Studio v2.0.18</th>
<th>Installation Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doxygen</td>
<td>1.8.16</td>
<td>DiaSemi/SmartSnippetsStudio2.0.18/Tools/doxygen</td>
</tr>
<tr>
<td>mingw-w64 (windows only)</td>
<td>4.3.5</td>
<td>DiaSemi/SmartSnippetsStudio2.0.18/Tools/mingw32</td>
</tr>
<tr>
<td>Msys2 (windows only)</td>
<td>20190524</td>
<td>DiaSemi/SmartSnippetsStudio2.0.18/Tools/msys2</td>
</tr>
<tr>
<td>Python</td>
<td>3.5</td>
<td>DiaSemi/SmartSnippetsStudio2.0.18/Python35</td>
</tr>
</tbody>
</table>

Depending on the selected SDK, there are some SDK dependencies that need to be installed. The set of SDK specific tools or the required versions may differ from SDK to SDK. Also, the selected SDK may have been tested with tool versions that are different than the ones bundled by default with the Studio installer.

To ensure that all required tools and their appropriate versions are installed, the SDK Tools Installer wizard is launched when Studio starts to guide the user through the installation process. A tool may be mandatory or optional. An optional tool can be specified in the configuration xml of the SDK with the attribute can_skip="true".

Installation of optional tools can be skipped in the wizard. The user’s choice to skip an optional tool is remembered by SmartSnippets™ Studio so that the wizard does not ask again for the installation of this tool the next time the wizard is launched. However, if the user decides to install an optional tool that has been skipped, a preference can be modified in order to trigger again the wizard which will also ask again for the skipped tools.

Figure 11 shows the first page of the wizard, which presents a table of the selected tools that this SDK requires along with the required version of the tool(s) and optionally the currently installed one for those that they are already installed. If the required version of a tool is not matching or not found at all, the wizard will prompt for tool installation.

**Important Note**: It is mandatory go through this wizard since otherwise SDK tools will not become functional or get installed at all. For that reason, users should never exit this dialog box by clicking the **Abort all** button. Instead, they should follow the steps till the **Finish** button appears (see Figure 15).
Figure 11: The SDK Tools Installer Wizard Intro Page

For the selected SDK, installer will guide the user to install Segger JLink, Segger Ozone, and Segger SystemView.

For tools that have been specified as optional, user can always skip those by clicking the Skip button and eventually continue with the rest of the installation process.

User should normally install all optional tools and then click the Next button to continue with the installation dialog, as in Figure 12.

Figure 12: Installation of Segger Ozone by the SDK Tools Installer

For mandatory tools, the user has to download and install such tools by clicking the Download button or can always specify an installation folder if the tool is already installed externally. The Next button is enabled only after specifying a valid installation path or successfully installing the tool.

Figure 13 shows the installation of Segger J-Link software components, which are mandatory.
The last page of the wizard shows a summary of installed and skipped tools (Figure 15).

If the option **Do not ask again for skipped tools** is selected, the SDK Tools Installer wizard will not ask again for the installation of tools that the user has selected to skip. Note that user preferences for skipped tools are SDK specific, meaning that user’s choice to skip a tool, or to select **Do not ask again for skipped tools**, has no effect on another SDK.
As mentioned in the last page of the wizard, the user can modify the choice for the wizard to ask or not for tools that have been skipped by the user through the preferences page.

Figure 16 shows SDK Tools Installer Wizard preferences page.

Selecting the Prompt for the installation of skipped tools checkbox and clicking Apply or OK will trigger SDK Tools Installer wizard, which will guide the user to install tools that have been skipped in the past.

Note: For tools provided by SEGGER (for example, the Ozone debugger, the J-Link software package and the System Viewer), older versions are delivered on as-is basis and come without support from SEGGER.

SEGGER Support can only be provided if the current version available on the SEGGER website shows the same misbehavior as the older one that is required for running an SDK.
2.3.1 SmartSnippets™ Toolbox Installation via SDK Tools Installer

SmartSnippets™ Toolbox is no longer installed by SmartSnippets™ Studio installer. To be able to launch the tool from the SmartSnippets™ Studio welcome page, it can be added as a dependency in the configuration xml file of the SDK. The SDK Tools installer wizard will guide the user on installing it and/or making its installation path known to Studio. The following lines can be added under `<tools>` node for windows platform to make SmartSnippets™ Toolbox available via SmartSnippets™ Studio:

```xml
<tool id="toolbox" can_skip="true">
  <name>SmartSnippets Toolbox</name>
  <official_url>https://sp-proptimality.s3.eu-west-3.amazonaws.com/ApplicationUpdates/ToolBox/SmartSnippets_Toolbox_v5.0.19.4046_windows.msi</official_url>
  <version>5.0.19</version>
  <official_download_url direct_download="no">https://sp-proptimality.s3.eu-west-3.amazonaws.com/ApplicationUpdates/ToolBox/SmartSnippets_Toolbox_v5.0.19.4046_windows.msi</official_download_url>
</tool>
```

The wizard will ask from the user to configure the installation path of Toolbox. The user can download and install SmartSnippets™ Toolbox installer from the URL provided by the SDK tools installer wizard page. If SmartSnippets™ Toolbox is already installed, the user can directly browse and select the path where SmartSnippets™ Toolbox is installed. Note that in this example Toolbox has been added as an optional dependency in the configuration xml file (can_skip attribute is true). If the tool installation is skipped or the SDK Tools Installer wizard dialog box is closed before installing SmartSnippets™ Toolbox, SmartSnippets™ Toolbox and Rf Master buttons will not be available in the welcome page, even if they are enabled in the configuration xml file.

![Figure 17: SmartSnippets™ Toolbox Page in SDK Tools Installer](image-url)
2.4 Change Workspace Toolchain

By default, toolchain is being set up by the SDK Tools Installer Wizard as described in Section 2.3. User may change the global or workspace toolchain from menu item Window > Preferences > MCU as shown in Figure 19.

Note that after a Studio restart or a workspace change, the toolchain location path reverts to its default version as defined in SDK configuration and setup by SDK Tools Installer Wizard.

User may also change the toolchain for a specific project by right-clicking on project and selecting Properties > MCU > Arm Toolchains Paths as shown in Figure 20.
In such a case user's selection remains after Studio restarts.
3 Welcome Page

The welcome page provides the user with the ability for easily launching applications, configure the SmartSnippets SDKs and HDKs (Hardware Development Kits), browse documentation and examples among other useful links.

The welcome page displayed in Figure 22 and Figure 23 can be reopened at any time via different accessing points, see points 1 and 2 recognizable as similarly.

Additionally, the welcome page presents a dynamic set of tools and URLs directly associated with the linked SDK. Depending on SDK selected slightly different options may be available on the welcome page.

**Important Note:** In case the selected workspace does not correspond to a valid SDK, a warning message is displayed on the welcome page prompting user to take some actions. User can either run the SDK finder tool to search for a valid SDK in local computer. (Figure 21). The user can dismiss it using the [x] button.

![Figure 21: No SDK Selected Error in Welcome Page](image)
Figure 22: SDK Welcome Page
The functionality of the welcome page is described with more details in the following sections.

3.1 Provided Configurations

**SDK Root directory selection** (Figure 22 point 3): User can configure from here the root path for the SDK/Workspace that would like to use.

When change in the above path introduced, SmartSnippets™ Studio will ask user if it is ok to go for a restart so it can switch to the new workspace. If user selects to cancel from the Switch Workspace dialog box, the SDK root path will not change.

In case that user is not using the latest SDK per chip family, an appropriate message is getting prompted, so to let the user know that he can go and download the latest one.

User can download the latest SDK by clicking the Download link (Figure 22 point 4).
SDK finder (Figure 23 point 19): SDK finder is a tool that searches on the local computer for valid SDKs. User select the startup directory (Figure 25) and a background process starts searching (Figure 26). When finished, results are presented in a table (Figure 27). If the Use this SDK button is clicked, the switch workspace verification is shown (Figure 24) and the new SDK is used after restart.

![Figure 25: SDK Finder Select Root Directory to Search](image)

![Figure 26: SDK Finder Active Search Indicator](image)

![Figure 27: SDK Finder Results](image)

**Note:** Tool process time depends on filesystem tree of the selected root search folder.

**Family selection** (Figure 23 point 20): If SKD supports more than one product family, user can select the family to work with. This option filter project’s build configuration, welcome page’s resources section, device selection as well as the results on some tools such as SDK Inspector.

**Device selection** (Figure 22 point 5): User can select the connected device from a list of supported devices.

For DA1468x SDKs, user can see a list of part numbers and matching devices displayed in (Figure 28) by clicking the “here” link (Figure 22 point 7).
In cases that user is not aware about the selected hardware, there is always the option to detect the connected devices by clicking the Detect connected devices(s) button (Figure 22 point 6).

A notification message will appear on the top right side of the welcome page, informing the user about the currently detected devices.

### 3.2 Notifications

In Figure 22, point 8 presents a notification that indicates a wrong configuration in Segger tools. User is advised to manually correct the error according to the notification's instructions.

The notification is automatically dismissed when user corrects the error or can be closed by clicking the \(^{\text{X}}\) button.

### 3.3 Listed Tools

Welcome screen contains a list of shortcuts to Tools so that the user can easily launch other applications from within SmartSnippets™ Studio.

**IDE:** (Figure 22 point 9): Switches to the IDE workbench in the C/C++ perspective.

**SmartSnippets Toolbox:** (Figure 22 point 10): Launches SmartSnippets Toolbox as an external application.

**Ozone Debugger:** (Figure 22 point 11): Opens SEGGER Ozone (JLink Debugger) as an external application. The path where user installed Ozone is set through Window > Preferences > Run/Debug > SEGGER Ozone.

**System View:** (Figure 22 point 12): Opens System View tool as an external application.

**Keil IDE:** (Figure 22 point 13): Opens Keil IDE as an external application instead of C/C++ perspective inside the IDE. Note that for DA14580/581/583 SDKs, C/C++ perspective is not available.
IAR Embedded workbench (Figure 23 point 22): Opens IAR Embedded workbench as an external application.

SDK inspector (Figure 23 point 21): Scans SDK root folder for projects. The following platforms are currently supported: Eclipse CDT projects and uVision Keil. Also shows new user projects created with Studio.

If tool is enabled by the SDK the icon is shown on the welcome page as in Figure 29.

![SDK Inspector Icon](image)

**Figure 29: SDK Inspector Icon**

Scan process starts when opening application. A spinning icon and empty table indicate that process is still running (Figure 30).

![SDK Inspector Scan Process](image)

**Figure 30: SDK Inspector Scan Process**

When done, projects are shown on the table (Figure 31). Projects are filtered based on Product Family selection of the welcome screen. This filter is shown on top of the table.
Project table has the following columns:

- **Project name**
- **Product families.** SDK inspector tries to find the supported families for a project using search patterns in SDK's config.xml file in combination with some code embedded patterns. An example of such pattern is for example:

  ```xml
  <studio_config>
    <sdk_search>
      <Eclipse id=".cproject">
        <family id="DA1468x">name="(.*)DA1468[0123]-0[01](.*)"</family>
        <family id="DA14585-586">name="(.*)DA1458[56](.*)"</family>
        <family id="DA1468x">file="startup_DA1468x.s"</family>
      </Eclipse>
      <uVision id=".uvprojx">
        <family id="DA14585-586">name="(.*)_58[56]"</family>
        <family id="DA14585-586">file="startup_DA14585_586.s"</family>
      </uVision>
    </sdk_search>
  </studio_config>
```

  Where `name=""` means to search for project's build configuration names that match this pattern. If match assign this project to corresponding family id. A project may support more than one product families.

  Pattern `file=""` means to search in project’s dependency files for files that match the given pattern.

  This is a best effort process. If no product family found, the table field is blank.

- **Type.** Applies only for Eclipse project. Two values are possible. *Native* and *embedded*.
- **IDE.** Either Eclipse or uVision.

The **Export data** button exports table contents in comma-separated text .csv format.
User can have access to project details with one of the following ways:

- Select project from the table and press the View project details button
- Right click on project row and click details on the context menu
- Double click on project row

Figure 32 shows an example of a project details view for an Eclipse type project.

The new information here is:

- Full path. User may open full path in system’s file explorer
- Product Families: Product families found to be supported by this project
- Devices. Just present the devices supported by the product families found. If no product families are found, this field is blank
- Toolchain. Applies for Eclipse projects only (Figure 32). If project have already imported into the workspace the toolchain name and path (only for Cross ARM GCC) is shown here. Toolchain is searched with respect to the hierarchy project settings-workspace settings-global settings
- If project is not imported into the workspace then, workspace default values or values taken from project’s configuration is shown
- Build configurations. Here shown all project’s build configurations, filtered by the selected family, plus those configurations that could not match to a product family
- Open in IDE button. When clicked, a table with all available build configurations are shown (Figure 33)

User selects the preferred build configuration. When the confirmation message is accepted (Figure 34) SDK inspector closes and project with the selected build configuration is opened at IDE (Figure 35).
3.4 Resources

API Documentation: (Figure 22 point 14): Opens the Dialog Semiconductor’s SDK Documentation located inside doc/html folder of the SDK. To work correctly, the location of a proper SDK should be given in point 3.

Software Resources: (Figure 22 point 15): Opens a dialog to the SDK root directory structure. After selecting a directory, the available projects are shown. The selected projects are imported in the workspace and SmartSnippets™ Studio switches to C/C++ perspective. Figure 36 shows an example of using Software Examples of welcome page to import projects from the SDK into the workspace. Import of software resources is not available for DA14580/581/583 SDKs.
3.5 SDKs for More than One Family

When SDK which supports more than one family is loaded for first time, configuration and information for all supported families is presented on the welcome page. The user can select one of the available families from the respective drop-down box. Then devices, tools, and resources of the selected family are presented only. The family selected by the user is saved in the workspace and the next time the same SDK is used, the configuration of the last selected family will be presented. Available build and debug configurations (if any), are also filtered by family and by device. When no specific device has been selected, all the build and debug configurations of the selected family are available.
Figure 37: No Family Selected
4 SmartBond Projects

Dialog’s SmartBond™ family is the simplest route to delivering the most power-friendly and flexible Bluetooth® Smart connected products to the market.


In the following chapters, we provide project examples based on SDK’s supported family members and tools.

4.1 Build a Simple Application in Keil

This section explains how the user can select, build, and run a simple software application on a development board using KEIL IDE.

Furthermore, it provides step-by-step instructions for setting up one of the example projects, build and eventually execute it via the debug mode in any of the supported devices. However, for the needs of this demonstration we will use a DA14585/586 device and for that reason we need to choose the appropriate SDK as well.

For more information regarding KEIL Environment, visit Keil Website.

Blinky is a simple application which demonstrates basic device initialization and LED blinking functionality.

Once SDK is successfully unzipped, blinky-example source code can be found under the directory ‘peripheral_examples’.

To get started, go through the following steps:

1. Open the folder containing the SDK files. This is the folder where you extracted the SDK zip file.
2. In `<sdk_root_directory>\projects\target_apps\peripheral_examples\blinky\Keil\_5, double-click blinky.uvproj to open the project in Keil.

![Figure 38: Blinky Project Directory](image)

The development environment should look like in Figure 39 when the project is opened with Keil.
3. Click the **Target Options** button, then click the **Device** tab. The dialog box should look like Figure 40.

4. Then click the **Linker** tab. Scatter files (.sct) are used for selecting memory areas.
5. Click the Debug tab and ensure J-LINK/J-TRACE Cortex is selected and that the Initialization File is set correctly to .sysram.ini.

6. Next, click the Settings button and check that the SW Device has been detected correctly.
7. You can click **OK** to save the settings in both windows. All settings have now been saved and you can continue to build the example.

8. You can build the project by pressing **F7** key or clicking the **Build** button.

9. Make sure that you have a UART connection between your PC and the motherboard. Check the COM number on your PC. The COM port numbers can be found in the Windows Device Manager (Control Panel > Device Manager > Ports (COM & LPT)).
10. Select **Setup > Serial Port** to configure the port with the following values:
   - **Baud Rate**: 115200
   - **Data bits**: 8
   - **Parity**: None
   - **Stop bits**: 1
   - **Flow control**: None

11. In Keil, select **Debug > Start/Stop Debug Session**.

12. If a non-licensed version of Keil is used, the following dialog box is displayed. Click **OK**.
13. Press **F5** or click the **Run** button to start code execution.

User can verify that the procedure got completed successfully once the blinky message is displayed on the UART terminal screen Figure 49 and the green LED is blinking on DA14585/586 Demo board. This indicates that the program downloaded successfully and is running on target device.

### 4.2 Build a Project in Eclipse/Gcc

This section explains how the user can select, build, and run/debug among other options a software application on a development board using GCC & Eclipse IDE.

Furthermore, it provides step-by-step instructions for importing one of the example projects, build and eventually execute it via the debug mode to any of the supported devices.
4.2.1 Import a Project to Eclipse

1. Select File/Import and select Existing Projects into Workspace.

   ![Figure 50: Importing a Project into the Eclipse Workspace](image)

   **Important Note**: Importing a project to Eclipse IDE should only happen if first the proper workspace is pre-selected, and the right SDK exported to it. Otherwise, no tools installed, no valid configuration is done, and so on. If this is the case, see Section 2.1.

2. A new dialog box will appear when Next is selected.

   ![Figure 51: Browse in Root Directory to Import a Project](image)

3. The user can now select the root directory of the project by clicking the Browse button.
4. Once the project has been selected, user can click OK to confirm and finally click Finish to import the selected project into the Eclipse workspace. The imported project appears in the Project Explorer.

4.2.2 Create a New Debug Configuration

1. Connect the Development Kit (Basic or Pro) to the PC.
2. Select the project on Project Explorer window and select Debug Configurations from the toolbar.
3. Right-click the **SmartBond** category of Debug Configurations.

![Figure 54: Create New Debug Configuration](image)

**Figure 54: Create New Debug Configuration**

**Note:** You can see all your debug configurations by pressing **Run > Debug Configurations**. Every change applied at a Debug Configuration will be saved at the respective launch file found inside the project.

Clicking **Debug** at a debug configuration also launches a Debug Session.
To open the Debug perspective, if not opened automatically, user can click one of the available perspectives shown in the upper right corner of Eclipse or click Window > Perspective > Open Perspective > Debug.

4.2.3 Build and Run with Debug Option Enabled

1. To build a project with the debug option enabled, first select the project name. Immediately the build icon will highlight. Click the down arrow of the build icon to select a debug build mode.
2. Check console for any errors.

3. If build get completed with success, a new folder is created with the name of the selected build configuration. In the above example a folder named DA14683-00-Debug_QSPI_SUOTA is created with a file pxp_reporter.elf inside.

After that point you can execute in run/debug mode by selecting on the drop-down menu in the icon the appropriate option to execute/debug on target. In this case we can select the QSPI option and the Debug Perspective should come forward.

4.2.4 Use Global Debug Configurations

SmartSnippets™ Studio has three built-in debug configurations for SmartBond devices.
These configurations are global, meaning that instead of belonging to a specific project they are
dynamic and are getting applied to the project that is currently selected. The debug configurations
are the following:

- **RAM**: Executes from RAM
- **QSPI**: Resets and executes from QSPI (requires an image flashed in QSPI)
- **ATTACH**: Attaches to a running target

Figure 61 shows Debug Configurations Manager with RAM, QSPI and ATTACH debug
configurations.

Notice that when no project has been selected yet, C/C++ Application and Project fields are empty.

To use RAM debug configuration, the user has to build a project with a Debug_RAM or
Release_RAM build configuration.

When a RAM active build configuration has been selected, QSPI debug configuration is not
available.

When a build configuration suitable for debugging with RAM debug configuration is not present in the
selected project, the respective RAM debug configuration is hidden from the list of available
SmartBond debug configurations. For example, project “ancs” (Figure 62) has only QSPI suitable
build configurations, so RAM debug configuration will not be available.
Similarly, when user wants to debug with a QSPI debug configuration, a QSPI build configuration should be selected.

ATTACH debug configuration option can be used in combination with any build configuration. If user selects the project in the Project Explorer and opens the Debug Configurations Manager, fields 
C/C++ Application and Project will be prefilled with project-specific settings.

To control which build configurations are required to enable a debug configuration, the user can specify a set of valid substrings that should be part of the name of the RAM build configurations and QSPI build configurations respectively. By default, a build configuration containing the string “QSPI”
is suitable for debugging with the QSPI debug configuration and a build configuration containing the string “RAM” is suitable for the RAM debug configuration.

To change these default values, the user can enter a set of comma-separated values in RAM_BUILD_CONFIGURATION_NAMES and QSPI_BUILD_CONFIGURATION_NAMES variables in C:\DiaSem\SmartSnippetsStudio2.0\CDT\configuration\config.ini file. If these variable names are not existing in the config.ini file mentioned above, user needs to provide a value for them.

For example, entering RAM_BUILD_CONFIGURATION_NAMES = Debug_RAM, R_RAM, Debug_Memory in the config.ini file means that any project that contains build configurations with names Debug_RAM or R_RAM or Debug_Memory, can be launched with the RAM debug configuration.

4.2.5 Use SDK-Specific Debug Configurations

Users can place their own debug configurations under <sdk_root>\config folder. The three global launch configurations (RAM, QSPI, ATTACH) will be created by default even if “config” folder does not exist or being empty. Nevertheless, if user places .launch files with names RAM.launch or QSPI.launch or ATTACH.launch under <sdk_root>\config folder, the respective global launch configurations will be initialized from these files.

Note that the global launch configurations are automatically saved inside the workspace under the .metadata\plugins\org.eclipse.debug.core\launches subfolder.

If the user would like to update the global launch configurations with any recent modifications made on the .launch files located under the <sdk_root>\config folder, it is required to delete the respective files in .metadata\plugins\org.eclipse.debug.core\launches subfolder first so that the modified .launch files are copied to the workspace.

4.2.6 Clone a Project outside the SDK Folder Structure

Cloning a project is needed in case you want to use an existing SmartBond project provided inside the SmartSnippets SDK and use it as a basis for developing your own application. In case the new project folder needs to be created outside the SDK folder structure, this section gives a small how-to guide (see Section 4.2.7 for a simpler way in case the new project folder can be created inside the SDK folder structure):

1. Navigate to <sdk_root>\projects\dk_apps\demos.
2. Create a folder with the desired project name (for example, “my_project”).
3. Choose a project. For example, “pxp_reporter” project, located inside “projects/dk_apps/demos” folder, can be used. Copy the contents of the “pxp_reporter” folder to the “my_project” folder.
4. Browse in “my_project” and find the following files:
   a. .cproject
   b. .project
5. Open them with a text editor. Find and replace every “pxp_reporter” string with the string “my_project”. Save them and quit.
6. Start SmartSnippets™ Studio and import the project “my_project” to the workspace.
7. Rename (right-click) the files containing the old project’s name to the new project’s name, for example, “pxp_reporter_task.c” > “my_project_task.c”. This can be either done from eclipse’s project explorer or windows project folder.

8. Right-click to the project and navigate to properties:
   a. Select Linked Resources under Resource.
b. If required, edit **SDKROOT** to match the root folder of the SDK.

![Edit Variable](image)

**Figure 67: Edit the Right Path for “SDKROOT”**

- Then, click the **Linked Resources** tab and fix the invalid variable relative locations if any.

![Linked Resources](image)

**Figure 68: Fix the Invalid Variables**
d. Click **Settings** under **C/C++ Build**. In the **Configuration** menu, select **All configurations**, then click **Libraries** under **Cross ARM C Linker**, and edit Library search path (-L) if required, to match the SDK root directory. Click **Apply**, then click **OK** and exit.

![Properties for my_project](image)

**Figure 69: Setup the Path for the Libraries**

9. Right-click “my_project” and click **Refresh**. Then you can build the project.

### 4.2.7 Copy Projects in a Workspace

Cloning is needed when user wants to copy an existing SDK project to another location outside the SDK folder structure and work from there (see Section 4.2.6 for a detailed guide). If the user has the option to copy an existing project under the same location as the original one, the following guide requires less actions, since no references or relative paths need to change:

1. Select the project to be copied. Right-click and then click **Copy**.

![Copy a Project. Step 1](image)

**Figure 70: Copy a Project. Step 1**

2. Right-click into the workspace and click **Paste**.
3. Copy project dialog box appears.

4. Clear the Use default location checkbox and browse to the same location as the original project. Create a new folder, select it, and click OK.

5. In the copy project dialog box, enter a project name different than the original one and click OK. The new copied project is ready for development. All project preferences, properties, and relative paths have been copied from the original project.

If a project is copied to a directory, in another level in the directory structure than the original project, an error may occur during the link step. For example, DA1468x SDK projects in directories projects/dk_apps/ble_profiles and projects/dk_apps/demos are using the ble_stack_da1468x library...
that resides inside the SDK and is not copied together with the project. To resolve such issues, some changes are needed in the project settings. See Section 15.18 and follow steps 8 to 10 in Section 4.2.6 to fix the libraries’ paths and build the copied project without errors.

4.2.8 Enable Parallel Make Compilation
To enable GNU Make parallel compilation support for a project, apply the following configuration:

1. Right-click a project found in project explorer and select Settings.
2. Click the Build Steps tab under the path C/C++ Build > Settings > Build Steps.
3. Add “+” symbol in the beginning of the Pre-build steps command, as in Figure 74.

![Figure 74: Correct Prebuild Steps Command for Enabling Parallel Build](image)

4. Click the Behavior tab under the path C/C++ Build > Behavior.
Figure 75: Behavior Tab to Enable Make’s Parallel Build

5. Enable parallel build option by specifying the amount of used parallel jobs.
6. Select **Apply and Close** to save and exit configuration wizard.
7. Locate the “makefile.targets” file under the project explorer’s project structure.
8. Edit the file accordingly to support parallel compilation and remove the ".NOTPARALLEL" indicator.
9. Finally, clean and build your project.
5 Scripts

To use or test a project, the binary must be loaded to the ProDK. This procedure is supported by the scripts, which are included in the SmartSnippets DA1468x and DA1469x SDKs and are located in the following path:

```
<sdk_root>/utilities/scripts
```

Note that for DA1468x RDs the path is:

```
<dk_root>/sdk_680/utilities/scripts
```

5.1 Import Scripts

To import the scripts to SmartSnippets™ Studio, follow the same procedure as importing a project:

1. Select File/Import and select Existing Projects into Workspace.

![Figure 76: Importing a Project into the Eclipse Workspace](image1)

2. A new dialog box appears when Next is clicked.

![Figure 77: Browse in Root Directory to Import a Project](image2)
3. Click the **Browse** button to select the directory path where to import the project from. Select the path `<sdk_root> > utilities > scripts` as shown in **Figure 78**.

![Figure 78: Browse and Select Scripts](image)

4. Click **OK** to confirm and then **Finish** to import the selected project into the Eclipse workspace. Now the scripts are available for use. Note that there are more scripts available in the SDK, but they are filtered depending on the Operating System. **Figure 79** shows the supported scripts for Windows.

![Figure 79: Scripts](image)

5.2 **Execute Scripts**

If a project has been successfully built, you can load it to the device using `program_qspi_serial_win` (`program_qspi_serial_linux` for Linux). Alternatively, the binary can be loaded to the Pro DK with the script `program_qspi_jtag_win` which is faster. Loading a binary file is performed by the scripts. To use
a script for the first time, QSPI needs to be configured. Figure 80 shows the information needed to configure QSPI.

![Figure 80: Configure QSPI](image)

To change the configuration, choose `program_qspi_config_win`.

## 6 Makefile Generation

SmartBond and SmartSnippets SDK projects are configured in such a way so that Eclipse generates Makefiles that can be executed by GNU Make. Eclipse uses them to build the projects, but the Makefiles can also be executed from the command line, which is very helpful for users (especially Linux) who want to maintain their projects based on Makefiles.

Note that on Linux manual building of `utilities/bin2image` is required.
7 JTAG Connection

For debugging purposes, the Segger JLink adaptor is used. Typically, the JLink adaptor is a small box with a JTAG/SWD interface to the ARM processor core on the DK, and an USB or Ethernet interface to the PC. SWD is the ARM-specific interface with only two wires that supports the same functionality as the more known JTAG interface.

JLink adaptors are available in several tastes. Dialog has integrated a Segger JLink adaptor on the DK itself. It can be connected to the PC via the USB interface named USB2(DBG). See also DA1468x/DA1510x Pro DK user manual (Ref. [3]).

Typically, during a debug session, only one JTAG SWD interface is used. When more than one JTAG/SWD interfaces are connected to the PC, where the debug session takes place, the following window appears asking the user to select which one should be used. Select that appropriate serial number and click the OK button. A timeout message will be issued if user does not make a selection within a specific timeframe.

![Figure 81: Selecting the JTAG Connection](image)

8 Dialog’s Code Formatter

Dialog provides a code formatter for SmartSnippets™ Studio. When an SDK is used for the first time, Dialog’s default code formatter is selected automatically in SmartSnippets™ Studio.

User can override the default code formatter by navigating to Window > Preferences > C/C++ > Code Style > Formatter. If user wants to restore the default code formatter, he/she is advised to select Import, navigate to the Other folder (should be under folder SmartSnippetsStudio2.0.6\CDT\Other of the installation directory), and select Dialog_formatter.xml.
9 CMSIS File for DA1468x_00 and DA1469x Registers

The ARM Cortex Microcontroller Software Interface Standard (CMSIS) is a vendor-independent hardware abstraction layer for the Cortex-M processor series and specifies debugger interfaces.

The CMSIS enables consistent and simple software interfaces to the processor for interface peripherals, real-time operating systems, and middleware. It simplifies software reuse, reducing the learning curve for new microcontroller developers and cutting the time-to-market for devices.

To update the EmbSys Registers preferences page:

1. Click ` 그리스 ` from the EmbSys Registers view to open the EmbSys Registers Preferences page. If EmbSys View is not visible, you can add it to the workbench by selecting Window > Show View > Other > Debug > EmbSys Registers.

2. Select SVD(CMSIS) in the Architecture list and Dialog_Semiconductor in the Vendor list, as in Figure 82.

3. Select the Chip of your preference.

4. Click the OK button and EmbSys Registers’ view will be populated as shown in Figure 83.

Figure 82: Selecting CMSIS file for DA1468x_00 Registers
Figure 83: The EmbSys Registers Screen
10 Doxygen Documentation

Doxygen is a tool that extracts documentation from source file comments and generates output in HyperText Markup Language (HTML). To build the Doxygen documentation:

1. Check if your project contains already a `.doxyfile`. If not, you can add it by right-clicking a project and selecting **New > Other > Doxyfile**.

   ![Figure 84: Doxygen Wizard](image)

   **Figure 84: Doxygen Wizard**

2. Build the doxygen documentation by right-clicking on the doxyfile and selecting **Build Documentation**.

   ![Figure 85: Building Doxygen Documentation](image)

   **Figure 85: Building Doxygen Documentation**

Alternatively, you can build the documentation by clicking the button on top menu and selecting the `.doxyfile` of the project you want to build documentation for.
3. After building the documentation, you can find the contents in the **html** and **latex** folders.

**Figure 87: Project Explorer with Doxygen Generated Documentation**

**Note:** It is possible to copy a `.doxyfile` to another project and build documentation for it. You can open a doxygen file in doxygen editor, by right-clicking on it and selecting **Open With > Doxyfile Editor**. Now you can easily set up any project specific settings, like project name, version, input folders, and so on.
Figure 88: Editing File with Doxygen File Editor
11 SEGGER Terminal Configuration

SEGGER Real Time Terminal is an alternative for semi hosting. It enables a very fast way to redirect printf statements from the board to the Telnet Terminal in our Eclipse environment.

To establish the connection to the board, set up the Telnet terminal as follows:

1. Select **Window > Show View > Other**.
2. In the window that appears, select **Terminal** and click **OK**.

![Adding the Terminals View](image)

**Figure 89: Adding the Terminals View**

3. A terminal window is added to the layout.

![The Layout with the Terminal View Included](image)

**Figure 90: The Layout with the Terminal View Included**

4. Click the Terminal icon (circled in red above). The **Launch Terminal** window appears.
5. Select **Telnet Terminal**, fill in localhost in the **Host** field and 19021 in the Port field.

6. Click OK. A telnet terminal session will be started.

    After completing the setup of the Terminal window, user can launch it in any perspective by pressing CTRL+ALT+t.
12 Serial Terminal View

SmartSnippets™ Studio comes installed with the RxTx plugin which enables serial terminal connection with the development board (http://rxtx.qbang.org/wiki/index.php/Main_Page)

1. Select **Window > Show View** to locate and open the terminal view.
2. Click the **Open a terminal** button.

![Figure 92: The RxTx Terminal View](image)

3. Select **Serial Terminal** from the list.

![Figure 93: The RxTx Launch Terminal Window](image)

4. Select the UART port, Baud Rate and click **OK**.

The serial output should be shown on the terminal window.

![Figure 94: An Example of the RxTx Terminal Window](image)
13 Launch SmartSnippets Toolbox as an External Tool

To launch SmartSnippets Toolbox through SmartSnippets™ Studio:

1. Select Run > External Tools > External Tools Configuration > SmartSnippets Toolbox > New. A new configuration which launches SmartSnippets Toolbox is created. If needed, the default path can be changed on the Main tab. This step is only needed to be performed once.

![Create New Configuration](image1)

**Figure 95: Create New Configuration**

2. Click the Run button. From now on, the new configuration that launches SmartSnippets Toolbox should appear directly under Run > External Tools.

![New Configuration under External Tools](image2)

**Figure 96: New Configuration under External Tools**
14 Advanced Settings

SmartSnippets™ Studio comes with a set of variables whose values are pointing to the installation paths that have been used to install the different tools or are getting updated as the user changes project or build configuration. These variables are intended to be used in scripts or any other Studio projects. They can be found under Windows > Preferences > Run/Debug > String Substitution.
15 Troubleshooting

15.1 Project Errors while Builder is Still Active

After creating a new template project, wait for the Indexer to finish before building the project. Invoking build, while indexer is still active will result in a project with build errors and should be avoided. Figure 97 shows the Indexer in action, after creating template project 680Test.

![Figure 97: Errors/Warnings while Indexer is Still Running](image)

Notice that while the Indexer is still active, 680Test project has a red marker, indicating errors. After the indexer finishes, there will be no errors at the template project.

15.2 Debug Session Launching Stops without any Error Indication

This may be an indication that the firmware used by the JTAG Debugger is older than the SEGGER J-Link GDB server version used. The user is advised to disconnect and reconnect the JTAG debugger and launch a new debug session. Now a message should pop up, informing users that they should update the debugger firmware.

15.3 Linux Installation Errors

If an error occurs while installing in Linux, check the installation log located at: `<user_home>/SmartSnippets_Studio_installation_<version>.log`

15.4 Debugger GDB Crash

In case GDB crashes randomly, remove all breakpoints and try again.
15.5 Semihosting

Semihosting is a mechanism for ARM targets to communicate input/output requests from application code to a host computer running a debugger. This mechanism could be used, for example, to enable functions in the C library, such as printf() and scanf(), to use the screen and keyboard of the host rather than having a screen and keyboard on the target system.

On some installations, semihosting (that is --specs=rdimon.specs in Properties > C/C++ build > settings > linker > Misc > Other linker flags) does not work, and the debugger fails to start. The solution is to use --specs=nosys.specs instead, in which case, of course, semihosting is not available.

15.6 Debugger Silent Mode Issue

On some installations, debugger does not work in silent mode. In this case, in the respective launch Run > Debug configurations > Debugger tab, clear the Silent checkbox. Note that all SmartSnippets DA1468x SDK projects are distributed with a debug launcher with silent mode switched off.

Application output for Semi Hosting can be seen in Semihosting and SWV console. In case another console has the focus while the application is running, the user can switch to semihosting console by selecting the  icon from the console view.

15.7 Indexer

Sometimes, the static code analysis tool may find unresolved symbols and other errors, even though compilation passes. This usually happens after code refactoring, or file renaming. Solution: Rebuild index (Project > C/C++ Index > Rebuilt)
15.8 JLink

When using two or more JLink connections to connect to multiple boards simultaneously, the connection may randomly stop on one of them. This is due to the limited bandwidth of the USB hub used. Attaching the JLink connections directly on the laptop solves the problem.

15.9 Wrong Proposed Workspace Path

While starting SmartSnippets™ Studio, if the workspace launcher proposes a folder that does not seem to have the correct format (for example, two folder paths one next to each other), try restarting the machine. Or just press the Browse button to select the workspace path.

15.10 Manual Plug-In Updates

Every SmartSnippets™ Studio release version comes together with a set of plug-ins that have been tested thoroughly for compatibility with the main applications and with each other. It is highly recommended that SmartSnippets™ Studio users do not manually update plug-ins to their latest versions. Updated plug-ins cannot be tested using the currently system-installed version of SmartSnippets™ Studio and its plug-ins, and that way compatibility cannot be reassured. Every new SmartSnippets™ Studio release will include all the latest versions for the included plug-ins too.

15.11 Internal Browser does not Start after a Plug-In Addition or Update

It may happen that an addition or update on existing plug-ins produces conflicts with the internal browser leading to the last one not able to proper start. As a workaround, go to Windows > Preferences > General > Web Browser and select Use external web browser.

15.12 Unhandled Event Loop Exception when Opening a Project

From time to time, when trying to open a closed project, the following error may appear.

![Figure 99: Error when Trying to Open a Project](image)

If this happens, restart the application. Projects will open without problems.

15.13 Couldn't Reserve Space for Cygwin's Heap when Building a Project

When trying to build an SDK project, the following error may appear:

```
D:\DiaSemi\SmartSnippetsStudio2.0.6\Tools\mingw64_targeting32\msys\1.0\bin\make.exe:
*** Couldn't reserve space for cygwin's heap, Win32 error 0
```

This happens because the external library msys-1.0.dll is not built to be position independent. In such an event, try the following:

1. Reboot the system.
2. Use the dll rebase to force the library load at a new address:
   $ rebase.exe -b 0x50000000 msys-1.0.dll

3. Change the Virtual Memory settings. As an example, for Windows 7 go to **Control Panel > System and Security > System > Advanced System Settings**. On the **Advanced** tab, click the performance **Settings** button and select **Advanced** tab, and click the **Change** button. Enter a custom size value as follows for Initial Size (MB) and Maximum Size (MB):
   - **Initial Size (MB)** = **Currently Allocated (shown at the bottom)**
   - **Maximum Size (MB)** = **Recommended (shown at the bottom)**

![Figure 100: Change Virtual Memory Settings](image)

4. Click the **Set** button, click **OK**, and reboot PC to take effect.

**15.14 Application Crashes or Buttons and Dialogs not Responding on Linux with GTK3**

There are known issues with Eclipse using the GTK3 library, especially with the KDE desktop environment. If you experience buttons not responding or certain menus that are not open, there are two potential solutions:

- First try to switch theme to classic from **Window > Preferences > Appearance**
● Restart SmartSnippets™ Studio

If it does not solve the problem force SmartSnippets™ Studio to use GTK2 instead of GTK3. Add these lines in SmartSnippets_Studio.ini before “pluginCustomization”:
```ini
--launcher.GTK_version
2
```

Restart SmartSnippets™ Studio.

15.15 Errors when Running `erase_qspi_jtag_win` and `suota_initial_flash_jtag_win` scripts

If earlier installations of SmartSnippets™ Studio and Segger tools exist under different user accounts, the new user must uninstall SmartSnippets™ Studio and the Segger tools from the user account that had been used when they were initially installed to remove them completely before she/he can install and run SmartSnippets™ Studio flawlessly. Otherwise, errors like the following ones may occur while trying to run `suota_initial_flash_jtag_win` and `erase_qspi_jtag_win` scripts:

```
Jlink.exe is not recognized as an internal or external command, operable program or batch file.
Failed to bind socket cannot open gdb interface
```

15.16 Errors after Importing a Project

Importing a project to Eclipse IDE should only happen if first the proper workspace is normally provided, and the right SDK exported to it.

Otherwise, no tools installed, no valid configuration is done, and so on. If this is the case, you are advised to go and fix this issue by repeating the Select Workspace procedure.
15.17 Errors after Renaming a Project

In general, it is strongly recommended to avoid renaming existing projects since this does not allow the user to modify the folder name, plus not all references are updated automatically. Instead, it is recommended to see Section 4.2.6 and Section 4.2.7 for details about the steps to be following when cloning a project outside the original SDK folder structure or copying under the same SDK folder structure. After copying it, the user could delete the initial project.

15.18 Errors after Copying a Project

If a project has not been copied properly, some compile errors related to relative paths may occur, for example:

Invoking: Cross ARM C Linker

c:/diasemi/smartsnippetsstudio2.0.6/gcc/4_9-2015q3/bin/..lib/gcc/arm-none-eabi/4.9.3/../../arm-none-eabi/bin/ld.exe: cannot find -lble_stack_da14681_01
collect2.exe: error: ld returned 1 exit status
make: *** [Copy of ancs.elf] Error 1

See Section 4.2.6 and Section 4.2.7 for details about the steps to be following when cloning a project outside the original SDK folder structure or copying under the same SDK folder structure.

15.19 Error: BAD ELF Interpreter: No such File or Directory. Make Error 126 or 127 on Linux

Support for 32-bit executables should be added (see also Section 1.2).
For CentOS and Fedora install with:
sudo yum install glibc.i686
For Ubuntu install with:
sudo apt-get install libz:i386 libncurses5:i386 libbz2-1.0:i386

15.20 Could not Determine GDB Version on Linux

If an error as in Figure 102 occurs, when trying to run a project, then ncurses library needs to be installed.

![Figure 102: Could not Determine GDB Version]

For CentOS and Fedora install with:
sudo yum install ncurses-libs.i686

15.21 Blank Welcome Screen when Opening SmartSnippets™ Studio

Install libwebkitgtk-1.0-0 and libwebkit-3.0-0 for Ubuntu. See Section 1.2 for more details.
15.22 Error on awk: “function strtonum never defined” when Running External Script

Install gawk. See Section 1.2 for more details.

15.23 SmartSnippets™ Studio Error: Cannot Register Python Interpreter

SmartSnippets™ Studio cannot register the bundled Python interpreter in Windows. User cannot use the PyDev plugin to run python scripts. This may have been caused by not updated windows system or missing windows library.

Solution:

- Move into Python installation directory `<install_dir>/SmartSnippetsStudio/Python35` and run a test python command, for example, “python --version”. If the error as in Figure 103 occurs, there are two solutions:

  ![Figure 103: Unable to Register Python due to System Error](image)

  - Update windows system using “Windows update” functionality
  - Download and install Update for Universal C Runtime in Windows. Restart SmartSnippets™ Studio

15.24 CentOS 7 and Ozone J-Link Debugger

CentOS 7 comes with libstdc++.so.6.0.19 (part of GNU GCC compiler), which is not compatible with latest versions of Ozone J-Link debugger. This results in Ozone exiting with error “Could not open J-Link shared library. Exit now”.

As a workaround user can follow these steps:

1. Download and install gcc 4.9.1 or later which contains libstdc++.so.6.0.20
2. Set LD_LIBRARY_PATH to include the 64-bit library folder of GCC. E.g. in `.bash_profile` add lines:

   ```bash
   LD_LIBRARY_PATH=/home/user/gcc-4.9.4/lib64
   export LD_LIBRARY_PATH
   ```

15.25 Python Interpreter Error on Linux: `ImportError: libtk8.6.so: Cannot Open Shared Object File: No such File or Directory`

Python program uses a module, for example, tkinter that requires Tcl/Tk to run.

There are two ways to resolve this issue:

1. Install Tcl/Tk on the system:
   - For Ubuntu
     ```bash
     sudo apt-get install tk tcl
     ```
2. Download and install ActiveState ActiveTcl® ([https://www.activestate.com/activetcl/downloads](https://www.activestate.com/activetcl/downloads)) and update LD_LIBRARY_PATH to point to the lib directory (e.g. in .bash_profile add lines

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/ActiveTcl-8.6/lib)

15.26 Error when Starting SmartSnippets™ Studio Application

If the message as in Figure 104 appears when executing SmartSnippets™ Studio application, check if the file `<install_dir>/SmartSnippetsStudioX.Y/CDT/SmartSnippets_Studio.ini` contains –vm option and –pluginCustomization option with correct option values.

![Figure 104: Error when Starting Application](image)

15.27 Error when Connecting DA1468x or DA1469x Family Chip with JTAG on Linux

If an error as in Figure 105 occurs when you try to connect to a DA1468x or DA1469x chip on Linux with JTAG, check that the `netstat` utility is installed on computer.

![Figure 105: Error when Connecting DA1468x or DA1469x Chip](image)

See Section 1.2 on how to install netstat utility based on Linux distribution.

15.28 Warning on Linux about Missing libqtgui4

On Linux system, click the SystemView icon from the welcome page (Figure 106) and if the message occurs as in Figure 107, see Section 1.2 on how to install the missing library.

![Figure 106: Welcome Page SystemView Icon](image)
15.29 Build Error when Workspace Path Contains Spaces

It is possible that user get build error if the Studio workspace path contains spaces. For example:

Generate linker scripts.

make[2]: *** No rule to make target 'mem.ld', needed by 'generate_ldscripts'. Stop.

Use a workspace that does not contain spaces in path.
# Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>19-Nov-2015</td>
<td>First released version</td>
</tr>
<tr>
<td>1.1</td>
<td>15-Apr-2016</td>
<td>Updated for SmartSnippets™ Studio v1.1</td>
</tr>
<tr>
<td>1.2</td>
<td>26-Aug-2016</td>
<td>Updated for SmartSnippets™ Studio v1.2</td>
</tr>
<tr>
<td>1.3</td>
<td>18-Nov-2016</td>
<td>Updated for SmartSnippets™ Studio v1.3</td>
</tr>
<tr>
<td>1.4</td>
<td>22-Dec-2016</td>
<td>Updated for SmartSnippets™ Studio v1.4</td>
</tr>
<tr>
<td>1.5</td>
<td>06-Apr-2017</td>
<td>Updated for SmartSnippets™ Studio v1.5</td>
</tr>
<tr>
<td>1.6</td>
<td>23-Jun-2017</td>
<td>Updated for SmartSnippets™ Studio v1.6</td>
</tr>
<tr>
<td>2.0.3</td>
<td>02-Feb-2018</td>
<td>Updated for SmartSnippets™ Studio v2.0.3</td>
</tr>
<tr>
<td>2.0.5</td>
<td>18-May-2018</td>
<td>Updated for SmartSnippets™ Studio v2.0.5</td>
</tr>
<tr>
<td>2.0.6</td>
<td>07-Nov-2018</td>
<td>Updated for SmartSnippets™ Studio v2.0.6</td>
</tr>
<tr>
<td>2.0.7</td>
<td>08-Feb-2019</td>
<td>Updated for SmartSnippets™ Studio v2.0.7</td>
</tr>
<tr>
<td>2.0.8</td>
<td>03-Apr-2019</td>
<td>Updated for SmartSnippets™ Studio v2.0.8</td>
</tr>
<tr>
<td>2.0.10</td>
<td>20-Sep-2019</td>
<td>Updated for SmartSnippets™ Studio v2.0.10</td>
</tr>
<tr>
<td>2.0.14</td>
<td>29-May-2020</td>
<td>Updated for SmartSnippets™ Studio v2.0.14</td>
</tr>
<tr>
<td>2.0.16</td>
<td>24-Dec-2020</td>
<td>Updated for SmartSnippets™ Studio v2.0.16</td>
</tr>
<tr>
<td>2.0.18</td>
<td>16-Dec-2021</td>
<td>Updated for SmartSnippets™ Studio v2.0.18</td>
</tr>
<tr>
<td>2.0.20</td>
<td>13-May-2022</td>
<td>Updated for SmartSnippets™ Studio v2.0.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Editorial corrections</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rebranded to Renesas</td>
</tr>
</tbody>
</table>
## Status Definitions

<table>
<thead>
<tr>
<th>Status</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAFT</td>
<td>The content of this document is under review and subject to formal approval, which may result in modifications or additions.</td>
</tr>
<tr>
<td>APPROVED or unmarked</td>
<td>The content of this document has been approved for publication.</td>
</tr>
</tbody>
</table>