Abstract

This manual 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

Contents
Abstract ................................................................................................................................................. 1
Contents .................................................................................................................................................. 2
Figures ................................................................................................................................................... 5
Tables ..................................................................................................................................................... 7
Terms and Definitions ............................................................................................................................ 8
References ............................................................................................................................................... 9
Prerequisites ......................................................................................................................................... 10
About this manual ................................................................................................................................. 11
Introduction .......................................................................................................................................... 12
Supported Systems & Versions ............................................................................................................... 13
1 SmartSnippets™ Installation ............................................................................................................... 14
  1.1 Fresh install in Microsoft Windows ................................................................................................. 14
  1.2 Fresh install in Linux ....................................................................................................................... 17
  1.3 Reinstalling / Uninstalling under Windows ..................................................................................... 20
  1.4 Reinstalling / Uninstalling under Linux .......................................................................................... 20
2 Starting SmartSnippets™ Studio ......................................................................................................... 22
  2.1 Selecting Workspace ....................................................................................................................... 22
  2.2 Supported SDKs .............................................................................................................................. 23
  2.3 SDK Tools Installer .......................................................................................................................... 23
  2.4 Change workspace Toolchain ......................................................................................................... 27
3 Welcome page ...................................................................................................................................... 28
  3.1 Provided Configurations .................................................................................................................. 30
  3.2 Notifications ..................................................................................................................................... 32
  3.3 Listed Tools ...................................................................................................................................... 32
  3.4 Resources ....................................................................................................................................... 37
  3.5 SDKs for more than one family ....................................................................................................... 38
4 SmartBond projects ............................................................................................................................ 40
  4.1 Building a simple application in Keil .............................................................................................. 40
  4.2 Building a project in Eclipse/Gcc .................................................................................................. 46
    4.2.1 Import a project to Eclipse .......................................................................................................... 46
    4.2.2 Create a new Debug Configuration ......................................................................................... 48
    4.2.3 Build & Run with debug option enabled ................................................................................. 50
    4.2.4 Use global Debug Configurations ......................................................................................... 51
    4.2.5 Use SDK-specific Debug Configurations ............................................................................ 53
    4.2.6 Cloning a project outside the SDK folder structure ............................................................. 53
    4.2.7 Copy projects in a workspace ............................................................................................... 56
    4.2.8 Enable parallel Make compilation ..................................................................................... 58
5 Scripts .................................................................................................................................................. 60
User Manual                                                                              Revision 2.0.14                     29-May-2020
CFR0012                                                                               2 of 86                     © 2020 Dialog Semiconductor
SmartSnippets™ Studio

5.1 Importing Scripts ................................................................. 60
5.2 Executing Scripts ............................................................... 62

6 Makefile Generation ............................................................ 63

7 JTAG connection ................................................................. 64

8 Dialog’s Code Formatter ....................................................... 65

9 CMSIS file for DA1468x_00 and DA1469x registers ................. 66

10 Doxygen Documentation ..................................................... 68

11 SEGGER Terminal Configuration .......................................... 71

12 Serial Terminal View .......................................................... 73

13 Launching SmartSnippets Toolbox as an External Tool .............. 74

14 Advanced Settings ............................................................. 75

15 Troubleshooting .................................................................. 76

A.1 Project errors while builder is still active ............................. 76
A.2 Debug session launching stops without any error indication ..... 76
A.3 Linux installation errors ....................................................... 76
A.4 Debugger GDB crash ............................................................ 77
A.5 Semihosting ......................................................................... 77
A.6 Debugger silent mode issue .................................................. 77
A.7 Indexer ................................................................................ 78
A.8 JLink .................................................................................. 78
A.9 Wrong proposed workspace path ......................................... 78
A.10 Manual plugin updates ....................................................... 78
A.11 Internal Browser does not start after a plug-in addition or update 78
A.12 Unhandled event loop exception when opening a project .... 78
A.13 Couldn’t reserve space for cygwin’s heap when building a project .... 79
A.14 Application crashes or buttons and dialogs not responding on Linux with GTK3................. 79
A.15 Errors when running erase_qspi_jtag_win and suota_initial_flash_jtag_win scripts ... 80
A.16 Errors after importing a project .......................................... 81
A.17 Errors after renaming a project .......................................... 81
A.18 Errors after copying a project .......................................... 81
A.19 Error: BAD ELF interpreter: No such file or directory. Make error 126 or 127 on Linux ..... 81
A.20 Could not determine GDB version on Linux .......................... 81
A.21 Blank welcome screen when opening SmartSnippets™ Studio ........................................ 82
A.22 Error on awk: “function strtonum never defined” when running external script ............. 82
A.23 SmartSnippets™ Studio error: Cannot register Python interpreter .................................... 82
A.24 CentOS 7 and Ozone J-Link debugger .................................. 83
A.25 Python interpreter error on Linux: ImportError: libtk8.6.so: cannot open shared object file: No such file or directory ........................................ 83
A.26 Error when starting SmartSnippets™ Studio application ......... 83
A.27 Error when connecting DA1468x or DA1469x family chip with JTAG on Linux .......... 83
A.28 Warning on Linux about missing libqtgui4 ........................... 84
A.29 Build error when workspace path contains spaces ............... 84

User Manual Revision 2.0.14 29-May-2020

© 2020 Dialog Semiconductor

CFR0012 3 of 86
Figures

Figure 1: SmartSnippets™ Studio Install Link ................................................................. 14
Figure 2: Optionally uninstall previous version ........................................................... 15
Figure 3: Segger JLink GDB server installation directory .............................................. 15
Figure 4: Segger JLink GDB server installation directory with bundled GDB server .......... 16
Figure 5: SmartSnippets™ Studio installation directory ................................................ 16
Figure 6: Automatically install the J-Link ..................................................................... 18
Figure 7: Select SmartSnippets™ Studio install directory ............................................. 19
Figure 8: Allow launching on Ubuntu 20.04 .................................................................. 19
Figure 9: Uninstall GNU ARM GCC .......................................................................... 20
Figure 10: Uninstall shortcut on Linux ......................................................................... 20
Figure 11: The SDK Tools Installer Wizard Intro Page .................................................. 24
Figure 12: Installation of Segger Ozone by the SDK Tools Installer ............................... 25
Figure 13: Installation of Segger J-Link software components by SDK Tools Installer .... 25
Figure 14: Skipping Segger SystemView installation in SDK Tools installer ................. 26
Figure 15: SDK Tools Installer wizard Summary Page ................................................... 26
Figure 16: SDK Tools Installer Wizard preference page ................................................ 27
Figure 17: Change global or workspace toolchain ....................................................... 27
Figure 18: Change project toolchain .......................................................................... 28
Figure 19: No SDK selected error in Welcome page ..................................................... 29
Figure 20: SDK welcome page .................................................................................. 29
Figure 21: More options on SDK welcome page ............................................................ 30
Figure 22: Switch Workspace verification ..................................................................... 30
Figure 23: SDK Finder select root directory to search .................................................. 31
Figure 24: SDK Finder active search indicator .............................................................. 31
Figure 25: SDK Finder results ................................................................................... 31
Figure 26: Part number info popup dialog ................................................................... 32
Figure 27: SDK Inspector icon .................................................................................. 33
Figure 28: SDK inspector scan process ..................................................................... 34
Figure 29: SDK inspector project table ....................................................................... 34
Figure 30: SDK inspector project details of Eclipse project ......................................... 36
Figure 31: Select build configuration to open .................................................................. 36
Figure 32: SDK Inspector. Confirm open project in IDE ............................................... 37
Figure 33: SDK Inspector selected project and build configuration ............................... 37
Figure 34: Import a project ....................................................................................... 38
Figure 35: No family selected ...................................................................................... 39
Figure 36: Blinky Project directory ............................................................................ 40
Figure 37: Blinky Project Keil Workspace ................................................................... 41
Figure 38: Blinky Project Options ............................................................................. 41
Figure 39: Blinky Project Scatter File .......................................................................... 42
Figure 40: Blinky Project: Debug Option .................................................................... 42
Figure 41: Blinky Project: JLink setup ........................................................................ 43
Figure 42: Blinky Project: Project Building ................................................................. 43
Figure 43: Device Manager Ports ............................................................................. 44
Figure 44: Blinky Project: Start Debug Session ............................................................ 44
Figure 45: Keil Lite Pop Up Window ........................................................................... 45
Figure 46: Blinky Project: Code Execution ................................................................ 45
Figure 47: Blinky Project: Blinky message on terminal ............................................... 45
Figure 48: Importing a project into the Eclipse workspace ............................................ 46
Figure 49: Browse in root directory to import a project ............................................... 46
Figure 50: Browse and select pxp_reporter .................................................................. 47
Figure 51: Pxp_reporter has been imported ................................................................. 47
Figure 52: Create New Debug Configuration ............................................................... 48
Figure 53: New SmartBond Debug Configuration ....................................................... 48
Figure 54: Debug Configuration Setup ....................................................................... 49
SmartSnippets™ Studio

Figure 55: Confirm Perspective Switch screen .......................................................... 49
Figure 56: Debug Perspective ................................................................................. 49
Figure 57: Building with debug option ................................................................. 50
Figure 58: xxx.elf file in Debug folder .................................................................. 50
Figure 59: Global debug configurations when no project is currently selected .... 51
Figure 60: Available build configurations .............................................................. 52
Figure 61: Example of hidden RAM debug configuration for a project .......... 52
Figure 62: Import my_project project in SmartSnippets™ Studio ...................... 54
Figure 63: Change to appropriate name ................................................................. 54
Figure 64: Match the “SDKROOT” to the root file of the SDK ......................... 55
Figure 65: Edit the right path for “SDKROOT” ..................................................... 55
Figure 66: Fix the invalid variables ....................................................................... 55
Figure 67: Setup the path for the Libraries ............................................................ 56
Figure 68: Copy a project. Step 1 .......................................................................... 56
Figure 69: Copy a project. Step 2 .......................................................................... 57
Figure 70: Copy project dialog .............................................................................. 57
Figure 71: Folder for copied project should be at the same location as the original 57
Figure 72: Correct Pre-build steps Command for enabling Parallel Build ........ 58
Figure 73: Behavior Tab to enable Make’s parallel Build ................................... 59
Figure 74: Importing a project into the Eclipse workspace ................................. 60
Figure 75: Browse in root directory to import a project ...................................... 60
Figure 76: Browse and select scripts .................................................................... 61
Figure 77: Scripts .................................................................................................. 61
Figure 78: Configure QSPI .................................................................................... 62
Figure 79: Selecting the JTAG connection ............................................................ 64
Figure 80: Selecting CMSIS file for DA1468x_00 registers .............................. 66
Figure 81: The EmbSys Registers screen .............................................................. 67
Figure 82: Doxygen wizard ................................................................................... 68
Figure 83: Building Doxygen documentation ...................................................... 68
Figure 84: Choosing which doxygen file to build ............................................... 69
Figure 85: Project Explorer with Doxygen generated documentation ................. 69
Figure 86: Editing file with doxylfe editor .............................................................. 70
Figure 87: Adding the Terminals view ................................................................. 71
Figure 88: The layout with the Terminal view included ...................................... 71
Figure 89: The Terminal settings screen ............................................................... 72
Figure 90: The RxTx Terminal View .................................................................... 73
Figure 91: The RxTx Launch Terminal Window .................................................. 73
Figure 92: An example of the RxTx Terminal Window ....................................... 73
Figure 93: Create New Configuration ................................................................. 74
Figure 94: New Configuration under External Tools .......................................... 74
Figure 95: Errors/Warnings while indexer is still running ................................. 76
Figure 96: Selecting Semimhosting console ........................................................ 77
Figure 97: Error when trying to open a project ..................................................... 78
Figure 98: Change virtual memory settings .......................................................... 79
Figure 99: Switch theme to classic ....................................................................... 80
Figure 100: Could not determine GDB version ................................................... 82
Figure 101: Unable to register python due to system error .................................. 82
Figure 102: Error when starting application ........................................................ 83
Figure 103: Error when connecting DA1468x or DA1469x chip ....................... 84
Figure 104: Welcome page SystemView icon ....................................................... 84
Figure 105: Missing libQtGui4 warning ............................................................... 84
Tables

Table 1: Default tool installation ........................................................................................................... 23
## Terms and Definitions

<table>
<thead>
<tr>
<th>Abbreviation</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 (e.g. RealTerm)
- A USB connection supporting USB-Serial (FTDI)
- Unzip program
- For Linux only: `netstat` utility is required for SmartSnippets™ Toolbox.
About this manual

This manual 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]
SmartSnippets™ Studio

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, etc.) 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 and contains as well:

- SmartSnippets™ Toolbox which covers all software development requirements, including:
  - Programming and loading of firmware into SRAM, OTP, and Flash
  - Power profiling
- SmartSnippets™ documentation

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/](https://www.segger.com/).

All components, tools and plugins are taken from the internet without changes and modifications. A subset of useful tools/plugins 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 please contact Dialog Semiconductor customer support.
SmartSnippets™ Studio

Supported Systems & Versions
SmartSnippets™ Studio is getting released with support for the following operating systems and versions:

- **Windows**
  - Win 10 64bit.

- **Linux**
  - Ubuntu 20.04 LTS.
  - Ubuntu 18.04.4 LTS.
SmartSnippets™ Studio

1 SmartSnippets™ Installation

1.1 Fresh install in Microsoft Windows

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

Supported variants:

- Windows 10 64-bit

Disk space requirements:

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

In order to proceed with installation, please go through the following steps.

1. Get access to the SmartSnippets™ SDKs. Please visit product page in https://support.dialog-semiconductor.com, download and unzip SDK zip-file, preferably under a subfolder of the home user’s directory (e.g. 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. Please also refer to 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 (e.g. less than 20 characters). This is important in order 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 shown 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 found, user can check to uninstall it first before proceeding to installation of current version. This may take some minutes.

![Figure 2: Optionally uninstall previous version](image)

5. Select to install recommended and latest version of SEGGER J-Link GDB server and click **Next**.

![Figure 3: Segger JLink GDB server installation directory](image)

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 pressing the **Next** button.
1. Select the destination folder for the SmartSnippets™ Studio and click Next.

   Destination Folder: this is the installation folder for SmartSnippets™ Studio. Default is C:\DiaSemi\ for windows installation.

2. The SmartSnippets™ Studio is installed. To run the application just double-click on 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.
SmartSnippets™ Studio

- 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 press 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 LTS (64-bit x86_64)
- Ubuntu 18.04.4 LTS (64-bit x86_64)

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

Disk space requirements:

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

In order to proceed with installation, please 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
   ```

   SmartSnippets™ for Linux come with a cross compiler for windows under <installation_folder>/Tools/mingw-w64. To be able to compile native projects on Linux please install a gcc compiler and make. On Ubuntu install with:
   ```
   sudo apt-get install build-essential
   ```

2. 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/. See also <troubleshooting> section.

3. 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.

4. 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) and update LD_LIBRARY_PATH to point to the lib directory (e.g. LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/ActiveTcl-8.6/lib)
5. To get access to the SmartSnippets™ SDKs, please contact your Dialog sales representative. Download and unzip the SDK zip file, preferably under a subfolder of the home user’s directory (e.g. 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.

6. 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.x84_64-X.X.X.XXX.run
```

User may run installer as root with sudo (e.g. sudo ./SmartSnippets_Studio-linux.gtk.x84_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.

7. Select to install the latest version of SEGGER J-Link GDB server and click Next.

![Installation of SmartSnippets Studio](image)

Figure 6: Automatically install the J-Link

8. 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.
9. 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)

![Figure 8: Allow launching on Ubuntu 20.04](image)
1.3 Reinstalling / Uninstalling 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.

Any workspaces (SmartSnippets™ Studio workspace and Toolbox workspace) will not be removed as long as they were 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 (e.g. 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.

In order 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.

![Uninstall GNU ARM GCC](image)

**Figure 9: Uninstall GNU ARM GCC**

1.4 Reinstalling / Uninstalling under Linux

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

To uninstall SmartSnippets™ Studio from Linux, user can search for the uninstaller shortcut (Figure 10).

![Uninstall shortcut on Linux](image)

**Figure 10: Uninstall shortcut on Linux**
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).

Any workspaces (SmartSnippets™ Studio workspace and Toolbox workspace) will not be removed as long as they were not 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 (e.g. 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**

Please 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 SDK Tools Installer section for more details.

2.1 Selecting Workspace

When SmartSnippets™ Studio starts for the first time, it prompts for a workspace selection under user’s profile folder (e.g. 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 Build error when workspace path contains spaces. In troubleshoot section.

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 (check Figure 20 for Ref. 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 setup 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 this Figure 19

This means that a predefined set of tools, URLs and buttons (e.g. Toolbox) 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.

**Note 1** It is better that path to workspace should not contain spaces or else some projects may have build errors.

**Note 2** If the folder selected for the workspace is part of an SVN/Git/etc. repository, it is highly recommended that there is an exclusion rule for the ‘.metadata’ folder (which is automatically created by the Studio environment). Otherwise, as different users submit/update their workspaces, an invalid state is likely to be reached.

**Note 3** 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 (e.g. Windows > Preferences > MCU > 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.
Note 4 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 `<sdk_root>\config\*SDK_config.xml` file by adding a prefix to the filename and placing the new file under the same folder `<sdk_root>\config>.
2. Modify the copied `*SDK_config.xml` file.
3. Restart SmartSnippets™ Studio.

Note 5 The following preferences are reset every time Studio launches:

10. C/C++ Indexer: "index source files not included in the build" and "index unused headers" are always set to false when Studio starts
11. C/C++ Build: "Build configuration only when there are Eclipse resource changes..." is always set to false when Studio starts

2.2 Supported SDKs

As of 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. The table below lists some of these tools:

Table 1: Default tool installation

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Version for Studio v2.0.14</th>
<th>Installation directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SmartSnippets™ Toolbox</td>
<td>5.0.14</td>
<td>DiaSemi/SmartSnippetsStudio2.0.14/Toolbox</td>
</tr>
<tr>
<td>Doxygen</td>
<td>1.8.16</td>
<td>DiaSemi/SmartSnippetsStudio2.0.14/Tools/doxygen</td>
</tr>
<tr>
<td>mingw-w64</td>
<td>4.3.5</td>
<td>DiaSemi/SmartSnippetsStudio2.0.14/Tools/mingw32</td>
</tr>
<tr>
<td>Msys2 (windows only)</td>
<td>20190524</td>
<td>DiaSemi/SmartSnippetsStudio2.0.14/Tools/msys2</td>
</tr>
<tr>
<td>Python</td>
<td>3.5</td>
<td>DiaSemi/SmartSnippetsStudio2.0.14/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 in order 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 by pressing the Abort all button. Instead they should follow the steps till the Finish button appears (see Figure 15).

![SDK Tools Installer Wizard Intro Page](image)

**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 selecting the “Skip” button and eventually continue with the rest of the installation process.

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

In Figure 13 the installation of Segger J-Link software components, which are mandatory, is illustrated.
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 checked, 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 check “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. Checking “Prompt for the installation of skipped tools” and pressing “Apply” or “OK” will trigger SDK Tools Installer wizard, which will guide the user to install tools that have been skipped in the past.
2.4 Change workspace Toolchain

By default, toolchain is being setup by the SDK Tools Installer Wizard as described on section SDK Tools Installer. User may change the global or workspace toolchain from menu item Window -> Preferences -> MCU as shown on Figure 17.

Figure 16: SDK Tools Installer Wizard preference page

Note: for tools provided by SEGGER (e.g. 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.

Figure 17: Change global or workspace toolchain
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 on Figure 18.

![Figure 18: Change project toolchain](image)

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 below [Figure 20] [Figure 21] can be reopened at any time via different accessing points, check for Ref. points [1] and [2] recognizable as respectfully.

Additionally, the welcome page presents a dynamic set of tools and URLs directly associated with the linked SDK. Depending on SDK selected slight different options may 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 19). The user can dismiss it using the [x] button.
Warning: This workspace does not contain a valid SDK. Please do the following:

a) Press the "SDK Finder" button, that will display a list of valid SDKs, found under the selected directory. Watch the progress on the lower right corner.

Search for SDKs in C:\tmp: (0%) or

b) Press SmartSnippets Toolbox, to use the toolbox without any SDK.

Figure 19: No SDK selected error in Welcome page

Figure 20: SDK welcome page
The functionality of the welcome page is described with more details in the following chapters.

3.1 Provided Configurations

**SDK Root directory selection (Figure 20 Ref. 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, the SDK root path will not change.

![Switch Workspace](image)

Figure 22: Switch Workspace verification
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 pressing the “Download” link (Figure 20 Ref. point [4]).

**SDK finder** (Figure 21 Ref point [19]): SDK finder is a tool that searches on the local computer for valid SDKs. User select the startup directory (Figure 23) and a background process starts searching (Figure 24). When finished, results are presented in a table (Figure 25). If pressed the “Use this SDK” button the “switch workspace verification” is shown (Figure 22) and the new SDK is used after restart.

![Figure 23: SDK Finder select root directory to search](image)

![Figure 24: SDK Finder active search indicator](image)

![Figure 25: SDK Finder results](image)

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

Family selection (Figure 21 Ref. point [20]): If SKD supports more than one product family, user can select the family to work with. This option filters 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 20 Ref. 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 26) by pressing the “here” link (Figure 20 Ref. point [7]).

![Part number info popup dialog](image)

**Figure 26: Part number info popup dialog**

In cases that user is not aware about the selected hardware, there is always the option to detect the connected devices by pressing the Detect connected devices(s) button (Figure 20 Ref. 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 20 the Ref. 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 pressing the 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.
SmartSnippets™ Studio

IDE: (Figure 20 Ref. point [9]): Switches to the IDE workbench in the C/C++ perspective

SmartSnippets Toolbox: (Figure 20 Ref. point [10]): Launches SmartSnippets Toolbox as an external application.

Ozone Debugger: (Figure 20 Ref. 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 20 Ref. point [12]): Opens System View tool as an external application.

Keil IDE: (Figure 20 Ref. 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 21 Ref. point [22]): Open IAR Embedded workbench as an external application.

SDK inspector (Figure 21 Ref. 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 following icon is shown on the welcome page (Figure 27):

![SDK Inspector icon](image)

Figure 27: SDK Inspector icon

Scan process starts when opening application. A spinning icon and empty table indicate that process is still running (Figure 28).
When done, projects are shown on the table (Figure 29). Projects are filtered based on Product Family selection of the welcome screen. This filter is shown on top of the table.

**Figure 29: SDK inspector project table**

Project table has the following columns:

1) Project name

2) 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 e.g.:
<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>

  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.

3) Type. Applies only for Eclipse project. Two values are possible. Native and embedded.

4) IDE. Either Eclipse or uVision.

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:

1) Select project from the table and press the View project details button
2) Right click on project row and click details on the context menu
3) Double click on project row.

Figure 30 shows an example of a project details view for an Eclipse type project.
The new information here is:

1) Full path. User may open full path in system’s file explorer.
2) Product Families: Product families found to be supported by this project.
3) Devices. Just present the devices supported by the product families found. If no product families found this field is blank.
4) Toolchain. Applies for Eclipse projects only (Figure 30). 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.
5) 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.
6) Open in IDE button. When pressed a table with all available build configurations are shown (Figure 31).

User select the preferred build configuration. When the confirmation message is accepted (Figure 32) SDK inspector closes and project with the selected build configuration is opened at IDE (Figure 33).
3.4 Resources

**API Documentation:** (Figure 20 Ref. 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 Ref. point [3].

**Software Resources:** (Figure 20 Ref. 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 34 illustrates 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 in 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, 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 35: 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. See http://www.dialog-semiconductor.com/bluetooth-smart for more details.

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

4.1 Building 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 please 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 please 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.

The development environment should look like in Figure 37 when the project is opened with Keil.
3. Click on the **Target Options** button, then click on the **Device** tab. The dialog should look like this.

4. Then click on **Linker** tab. Scatter files (*.sct) are used for selecting memory areas.
5. Then click on **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 on **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**

![Device Manager Ports](image)

*Figure 43: Device Manager Ports*

![Blinky Project: Start Debug Session](image)

*Figure 44: Blinky Project: Start Debug Session*
12. If a non-licensed version of Keil is used, the following dialog is displayed. Click OK.

![Figure 45: Keil Lite Pop Up Window](image)

13. Press **F5** or click the **Run** button to start code execution.

![Figure 46: Blinky Project: Code Execution](image)

User can verify that the procedure got completed successfully once the blinky message is displayed on the UART terminal screen Figure 47 and the green LED is blinking on DA14585/586 Demo board. This indicates that the program downloaded successfully and row running on target device.

![Figure 47: Blinky Project: Blinky message on terminal](image)
4.2 Building 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 48: 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, etc. If this is the case, you are advised to see the Selecting Workspace chapter.

2. A new pop up window will open once ‘Next >’ is selected.

   ![Figure 49: Browse in root directory to import a project](image)
3. The user can now select the root directory of the project by pressing the 'Browse...' button.

4. Once the project has been selected, user can press ‘OK’ to confirm and finally press ‘Finish’ to import the selected project into the Eclipse workspace. The imported project shows up 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.

![Figure 52: Create New Debug Configuration]

3. Right click on “SmartBond” category of Debug Configurations

![Figure 53: New SmartBond 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.

Pressing Debug at a debug configuration also launches a Debug Session.
To open the Debug perspective, if not opened automatically, user can press one of the available perspectives shown on the upper right corner of the Eclipse or click on Window > Perspective > Open Perspective > Debug.
4.2.3 Build & 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. Press the down arrow of the build icon to select a debug build mode.

   ![Image](figure57.jpg)

   Figure 57: Building with debug option

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.

   ![Image](figure58.jpg)

   Figure 58: xxx.elf file in Debug folder
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 59 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.

![Debug Configurations Manager](image)

Figure 59: Global debug configurations when no project is currently selected

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. E.g. project ‘ancs’ (Figure 60) 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:

![Figure 60: Available build configurations](image)

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:

![Figure 61: Example of hidden RAM debug configuration for a project](image)
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 Cloning 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 (please refer to paragraph Copy projects in a workspace 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 (e.g. “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 & 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, e.g. “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. Choose “Linked Resources” under “Resource”.

Figure 62: Import my_project project in SmartSnippets™ Studio

Figure 63: Change to appropriate name
b. If required, edit “SDKROOT” to match the root folder of the SDK.

c. Then, choose the “Linked Resources” tab and fix the invalid variable relative locations if any.
d. Then, choose “Settings” under “C/C++ Build”. In the drop down menu “Configuration:”, choose “[ All configurations ]”, then click on the “Libraries” under “Cross ARM C Linker” and edit Library search path (-L) if required, in order to match the SDK root directory. Click Apply, then OK and exit.

![Setup the path for the Libraries](image)

Figure 67: Setup the path for the Libraries

9. Right click “my_project” and click on Refresh.
10. 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 (please refer to paragraph Cloning a project outside the SDK folder structure 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. Press right click and then copy.

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

Figure 68: Copy a project. Step 1
2. Right click into the workspace and press "Paste"

![Image of workspace with right-click menu open, showing Copy option]

Figure 69: Copy a project. Step 2

3. Copy project dialog appears:

![Image of Copy Project dialog box]

Figure 70: Copy project dialog

Uncheck "Use default location" and browse to the same location as the original project. Create a new folder, select it and press OK.

![Image of Browse for Folder dialog box with folder structure]

Figure 71: Folder for copied project should be at the same location as the original
4. At copy project dialog, enter a project name different than the original one and press 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. E.g. 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 paragraph Errors after copying a project at troubleshoot section and follow steps 8 to 10 in paragraph 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, the following configuration needs to be applied:
1. Right-click over a project found in project explorer and select “Settings”.
2. Locate the “Build Steps” tab, found under the path – [C/C++ Build -> Settings -> Build Steps].
3. Add ‘+’ symbol in the beginning of the “Pre-build Steps” command, as in the Figure 72 below.

![Figure 72: Correct Pre-build steps Command for enabling Parallel Build.](image)

4. Locate “Behavior” tab, found under the path – [C/C++ Build -> Behavior].
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 Importing Scripts

To import the scripts to SmartSnippets™ Studio the same procedure as importing a project is followed:

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

![Figure 74: Importing a project into the Eclipse workspace](image)

2. A new pop up window will open once ‘Next >’ is hit:

![Figure 75: Browse in root directory to import a project](image)
3. Press 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 76.

![Figure 76: Browse and select scripts](image)

4. Hit ‘OK’ to confirm and then ‘Finish’ to import the selected project into the Eclipse workspace.

5. 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 77 shows the supported scripts for Windows.

![Figure 77: Scripts](image)
5.2 Executing Scripts

If a project has been successfully built, we 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 78 shows the information needed to configure QSPI.

![Figure 78: 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 a USB or Ethernet interface to the PC. SWD is the ARM-specific interface with only 2 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 (UM-B-060-DA1468x/DA1510x Development kit - Pro.pdf)

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 pops up asking the user to select which one should be used:

![Figure 79: Selecting the JTAG connection](image)

Please select that appropriate serial number and press the ‘OK’ button. A timeout message will be issued if user does not make a selection within a specific timeframe.
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 folder ‘Other’ (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 re-use, reducing the learning curve for new microcontroller developers and cutting the time-to-market for devices.

To update the EmbSys Registers preferences page:

1. Press \(<\) from the EmbSys Registers view to bring up 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

![CMSIS file for DA1468x_00 registers](image)

**Figure 80: Selecting CMSIS file for DA1468x_00 registers**

2. Select SVD(CMSIS) from Architecture and Dialog_Semiconductor from Vendor, as indicated at the image above.
3. Select the Chip of your preference
4. Press the OK button and EmbSys Registers’ view will be populated as illustrated below:

![EmbSys Registers screen](image)

**Figure 81: 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 on a project and selecting New > Other > Doxyfile

2. Now you can build the doxygen documentation by right clicking on the doxyfile and selecting Build Documentation
Alternatively, you can build the documentation by pressing the button on top menu and selecting the .doxygen file of the project you want to build documentation for:

![Figure 84: Choosing which doxygen file to build](image)

3. After building the documentation, the user will find the contents in html and latex folders.

![Figure 85: Project Explorer with Doxygen generated documentation](image)

Note: It is possible to copy a .doxygen file 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 > Doxygen Editor. Now the user can easily setup any project specific settings, like project name, version, input folders etc.
Figure 86: Editing file with doxyfile 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, setup the Telnet terminal as follows:

1. Select Window > Show View > Other
2. In the window that pops up, select Terminal and press OK:

![Figure 87: Adding the Terminals view](image)

3. A terminal window is added to the layout, as shown below:

![Figure 88: The layout with the Terminal view included](image)

4. Click on the Terminal icon (circled in red above). The 'Launch Terminal' window pops up:
5. Select ‘Telnet Terminal’, fill in localhost in the ‘Host’ field and 19021 in the Port field
6. Press ‘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 pop up the terminal view.
2. Press ‘Open a terminal button’:

![Figure 90: The RxTx Terminal View](image)

3. Select ‘Serial Terminal’ from dropdown menu:

![Figure 91: The RxTx Launch Terminal Window](image)

4. Select the UART port, Baud Rate and press OK
5. The serial output should be shown on the terminal window:

![Figure 92: An example of the RxTx Terminal Window](image)
13 Launching 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 in the Main tab. This step is only needed to be performed once.

2. Press the ‘Run’ button. From now on, the new configuration that launches SmartSnippets Toolbox should appear directly under Run > External Tools.
14 Advanced Settings

SmartSnippets™ Studio comes with a set of variables whose values are pointing to 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

A.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. The following screenshot shows the Indexer in action, after creating template project 680Test:

![Figure 95: 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.

A.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.

A.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`
A.4 Debugger GDB crash

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

A.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 (i.e. --specs=rdimon.specs in Properties > C/C++ build > settings > linker > Misc > Other linker flags) doesn't 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.

---

**Figure 96: Selecting Semihosting console**

A.6 Debugger silent mode issue

On some installations, debugger doesn't work in silent mode. In this case, in the respective launch Run > Debug configurations > Debugger tab, untick 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.
A.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)

A.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.

A.9 Wrong proposed workspace path
While starting SmartSnippets™ Studio, if the workspace launcher proposes a folder that doesn’t seem to have the correct format (e.g. two folder paths one next to each other), try restarting the machine. Or just press the ‘Browse… button’ to select the workspace path.

A.10 Manual plugin updates
Every SmartSnippets™ Studio release version comes together with a set of plugins 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 plugins to their latest versions. Updated plugins cannot be tested using the currently system-installed version of SmartSnippets™ Studio and its plugins, and that way compatibility cannot be reassured. Every new SmartSnippets™ Studio release will include all the latest versions for the included plugins too.

A.11 Internal Browser does not start after a plug-in addition or update
It may happen that an addition or update on existing plugins 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’.

A.12 Unhandled event loop exception when opening a project
From time to time, when trying to open a closed project, the following error may show up:

![Figure 97: Error when trying to open a project](image)

If this happen, restart the application. Projects will open without problems.
A.13 Couldn't reserve space for cygwin’s heap when building a project

When trying to build an SDK project, the following error may show up:
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 options:

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 on the performance Settings… button - select advanced tab, and click on 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 98: Change virtual memory settings](image)

Click on "Set" button, click "OK" and reboot PC to take affect.

A.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: 1) First try to switch theme to classic from Window > Preferences > Appearance.

![Appearance settings](image)

**Figure 99: Switch theme to classic**

2) 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”:

```
--launcher.GTK_version 2
```

Restart SmartSnippets™ Studio.

**A.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.
A.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, etc. If this is the case, you are advised to go and fix this issue by repeating the Selecting Workspace procedure.

A.17 Errors after renaming a project
In general, it is strongly recommended to avoid renaming existing projects since this doesn’t allow the user to modify the folder name, plus not all references are updated automatically. Instead, it is recommended to check paragraphs Cloning a project outside the SDK folder structure and Copy projects in a workspace 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.

A.18 Errors after copying a project
If a project has not been copied properly, some compile errors related to relative paths may occur, e.g.:

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

Check paragraphs Cloning a project outside the SDK folder structure and Copy projects in a workspace 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.

A.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 Fresh install in Linux).
For CentOS and Fedora install with:
sudo yum install glibc.i686
For Ubuntu install with:
sudo apt-get install libz1:i386 libncurses5:i386 libbz2-1.0:i386

A.20 Could not determine GDB version on Linux
If an error occurs like this one:
when trying to run a project, then ncurses library needs to be installed.

For CentOS and Fedora install with:

```
sudo yum install ncurses
```

**A.21 Blank welcome screen when opening SmartSnippets™ Studio**

Please install `libwebkitgtk-1.0-0` and `libwebkit-3.0-0` for Ubuntu. See section `Fresh install in Linux` for more details.

**A.22 Error on awk: “function strtonum never defined” when running external script**

Please install gawk. See section `Fresh install in Linux` for more details.

**A.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 e.g. “python --version”. If the following error shows up:

![Figure 101: Unable to register python due to system error](image)

there are two solutions:

1. Update windows system using “Windows update” functionality
A.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:
   ```
   LD_LIBRARY_PATH=/home/user/gcc-4.9.4/lib64
   export LD_LIBRARY_PATH
   ```

A.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 e.g. tkinter that requires Tcl/Tk to run. There are two ways to resolve this issue:

1. Install Tcl/Tk on the system:
   a. For Ubuntu
      ```
      sudo apt-get install tk tcl
      ```
2. Download and install ActiveState ActiveTcl® (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
   ```

A.26 Error when starting SmartSnippets™ Studio application

If the following message appears when executing SmartSnippets™ Studio application:

![Figure 102: Error when starting application](image)

Please check that file

<install_dir>/SmartSnippetsStudioX.Y/CDT/SmartSnippets_Studio.ini contains –vm option and -pluginCustomization option with correct option values.

A.27 Error when connecting DA1468x or DA1469x family chip with JTAG on Linux

If the following error occur when try to connect to a DA1468x or DA1469x chip on Linux with JTAG:
Figure 103: Error when connecting DA1468x or DA1469x chip

Please check that the netstat utility is installed on computer. See section Fresh install in Linux on how to install netstat utility based on Linux distribution.

A.28 Warning on Linux about missing libqtgui4

On Linux user press the SystemView icon from the welcome page Figure 104

![SystemView icon](image)

Figure 104: Welcome page SystemView icon

and the following message occurs Figure 105

![Missing libQtGui4 warning](image)

Figure 105: Missing libQtGui4 warning

Refer to section Fresh install in Linux on how to install the missing library.

A.29 Build error when workspace path contains spaces.

It is possible that user get build error if the Studio workspace path contain spaces. E.g.

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

Please use a workspace that do 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>
</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>

Disclaimer

Information in this document is believed to be accurate and reliable. However, Dialog Semiconductor does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. Dialog Semiconductor furthermore takes no responsibility whatsoever for the content in this document if provided by any information source outside of Dialog Semiconductor.

Dialog Semiconductor reserves the right to change without notice the information published in this document, including without limitation the specification and the design of the related semiconductor products, software and applications.

Applications, software, and semiconductor products described in this document are for illustrative purposes only. Dialog Semiconductor makes no representation or warranty that such applications, software and semiconductor products will be suitable for the specified use without further testing or modification. Unless otherwise agreed in writing, such testing or modification is the sole responsibility of the customer and Dialog Semiconductor excludes all liability in this respect.

Customer notes that nothing in this document may be construed as a license for customer to use the Dialog Semiconductor products, software and applications referred to in this document. Such license must be separately sought by customer with Dialog Semiconductor.

All use of Dialog Semiconductor products, software and applications referred to in this document are subject to Dialog Semiconductor's Standard Terms and Conditions of Sale, available on the company website (www.dialog-semiconductor.com) unless otherwise stated.

Dialog and the Dialog logo are trademarks of Dialog Semiconductor plc or its subsidiaries. All other product or service names are the property of their respective owners.

© 2020 Dialog Semiconductor. All rights reserved.