User Manual

DA1469x Getting Started Guide with the Development Kit

UM-B-090

Abstract

The focus of this User Manual is to easily introduce the DA1496x Getting started guide with the development kit.

Contents

Figures

Tables

1 Abstract

This guide is intended to help customers setup the hardware development environment, install required software and download and run an example application on the DA1469x development platform.

2 Prerequisites

- SmartSnippets™ Studio package
- Dialog's Semiconductor SmartSnippets™ DA1469x SDK
- Operating System (Windows or Linux)
- Pro DK DA1469x
- Serial-port terminal software (e.g. Tera Term)
- A USB connection supporting USB-Serial (FTDI)

3 Terms and Definitions

BLE	Bluetooth Low Energy
CLI	Command Line Interface
COM	Communication Port
FTDI	Future Technology Devices International
GPIO	General Purpose Input/Output
HW	Hardware
iOS	iPhone OS
JTAG	Joint Test Action Group (test interface)
LED	Light Emitting Diode
LDO	Low Drop Out
mikroBUs™	Standard defines mainboard sockets
OS	Operating System
OTP	One Time Programmable
PC	Personal Computer
PCB	Printed Circuit Board
ProDK	Pro Development Kit
RAM	Random Access Memory
SDK	Software Development Kit
SOC	System On Chip
SPI	Serial Peripheral Interface
SW	Software
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

4 Introduction

The DA14695x is a family of wireless MCU dual core ARM + M0+. It is the World's Most Advanced Wireless SOC. Comparing to DA1468x, the DA1469x has:

- 2x Processing power
- 2x Battery lifetime

- 4x More resources

The DA1469x SoCs combine :

- The newest ARM Cortex application processor with floating point unit
- Advanced power management functionality
- A cryptographic security engine
- Analog and digital peripherals
- A dedicated sensor node controller
- Software configurable protocol engine accompanied by a radio compliant to the Bluetooth® 5.0 low energy standard.

For further reading, more information can be found in the DA1496x Datasheet .

4.1 Guide Purpose

The purpose of this guide is to provide an overview of the DA14695 **Pro Development Kit (ProDK)** Board and describe the setup of the hardware and installation of the software tools to fully use the board.

The following hardware and software elements are required to use the DA14695 Pro Development Kit:

- The Pro Development Kit and Software Development Kit (SDK)
- SmartSnippets™ Studio 2.0 which can be installed on Windows or Linux hosts
- Windows users should download and install terminal software such as RealTerm, Putty or Teraterm. The rest of the document uses RealTerm. Linux users can use Putty

The rest of guide is organized as follows:

Section **5** describes the hardware components and their initial installation and setup.

Sections **6** and **7** describe the installation of the SmartSnippets™ DA14695 SDK software, along with all necessary tools.

Section **9** contains all steps for downloading and executing your first DA14695 Application.

4.2 How long should it take?

This tutorial requires **45-60 minutes** to complete. For more information the user may consult the accompanied documentation these are mentioned as **for further reading**.

4.3 Kit Content

The DA14695 ProDK Kit parts can be ordered via various distributors. The DA14695 Pro Development Kit contains the following:

- DA14695-00HQDEVKT-P main Board: **Figure 2**
- Mini USB-cable
- The DA14695-00HQDB-P daughter board **Figure 4**

5 DA1469x – The hardware

The Pro Development Kit (ProDK) consists of a main board (MB-PRO) DA14695-00HQDEVKT-P and a DA14695-00HQDB-P daughter board featuring DA14695 SoC of the desired package (BGA100). A DA14699-00HRDB-P daughter board is also available. More details about the DA1469x products family is shown in **Figure 1**.

Features	DA14691	DA14695	DA14697	DA14699
MCU	M33F, M0+, SNC	M33F, M0+, SNC	M33F, M0+, SNC	M33F, M0+, SNC
RAM size	384KB	512KB	512KB	512KB
Multi-core	yes	yes	yes	yes
USB Controller and USB pins	yes	yes	yes	yes
Charger	no	yes	yes	yes
1.8V and 3.0V power rails	yes	yes	yes	yes
Parallel LCD Controller	no	yes	yes	yes
Audio Processing Unit	yes	yes	yes	yes
LRA/ERM	no	no	yes	yes
White LEDs	no	no	yes	yes
QSPI RAM controller	no	yes	yes	yes
Analog hand driver	no	no	no	yes
Total GPIOs	44	44	55	55
Total pins	86	86	100	100
Packages	VFBGA86, 6x6mm, Pitch 0.55mm	VFBGA86, 6x6mm, Pitch 0.55mm	VFBGA100, 5x5mm, Pitch 0.475mm	VFBGA100, 5x5mm, Pitch 0.475mm

Figure 1: The DA1469x Product Family

5.1 The ProDK main board

Figure 2 illustrates the physical layout of the ProDK. The daughter board containing DA14695 device is shown in **Figure 4**.

The ProDK main board provides all necessary hardware to enable:

- Full functional verification of the DA14695 family of products with the ability to take precise power measurements by isolating the DA14695 device.
- Full digital connectivity with external hardware using UART, SPI and I2C.
- USB based debugging capabilities using the SEGGER J-Link on-board debugger for Cortex M33.
- USB based UART communication with the host PC using a Future Technology Devices International (FTDI FT2232H) chipset which converts UART to USB.

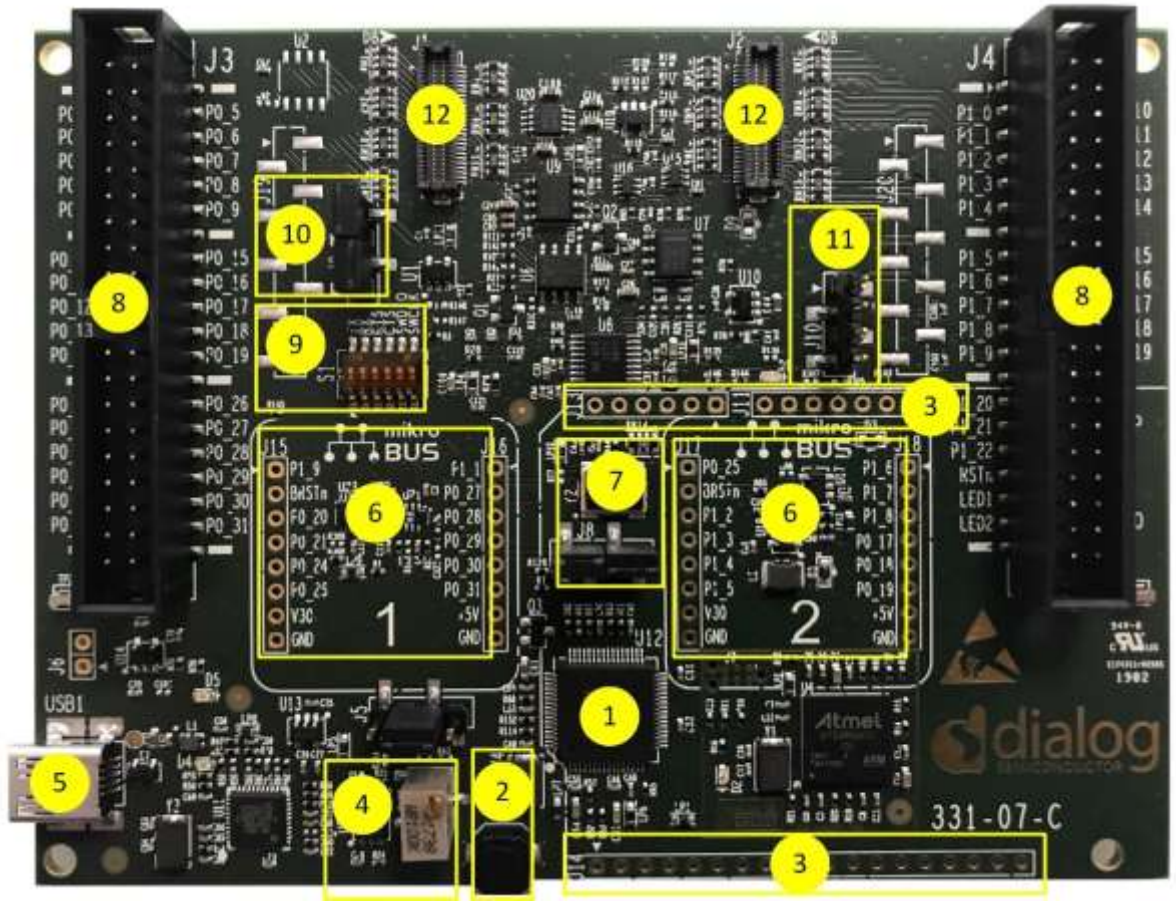

Figure 2: The DA14695 ProDK DK

Table 1: The DA14695 ProDK main elements

Reference	Description
1	U12: FTDI Chip
2	User button (K1)
3	Arduino sockets
4	Trimmer to define the adjustable LDO voltage (R127)
5	USB connector for power and communication interface
6	sockets
7	(J8.1-2) for enabling user button and (J8.3-4) for Software trigger header
8	Signal/Power breakout headers (J3, J4): Refer to **Appendix A:**
9	DIP switch (S1) connecting the debugging interface signals (JTAG/UART)
10	(J9.1-2: Input) and (J9.3-4: Output) for Header for current measurement circuit
11	LRA/ERM interface header
12	J1/J2 Daughter Board connectors

NOTE
J3 and J4 (Reference 8 in **Figure 2**) are the Breakout Headers (2 pcs 2x20pin) for monitoring GPIO and power signals, with markings of signal names on the PCB top silkscreen.

NOTE
For current measurement circuit by SmartSnippets Toolbox (driven from a DA14695 GPIO, P0_16) you should power the main board by the USB connector.

5.2 The ProDK daughter board

The ProDK main board can be combined with the daughter board shown in **Figure 6**.

Both ProDK main board and daughter board have alignment arrows printed to indicate the proper combination of the two boards, as shown in **Figure 3**.

Figure 3: Main board - daughter board alignment

Figure 4 presents the DA14695 daughter board Layout.

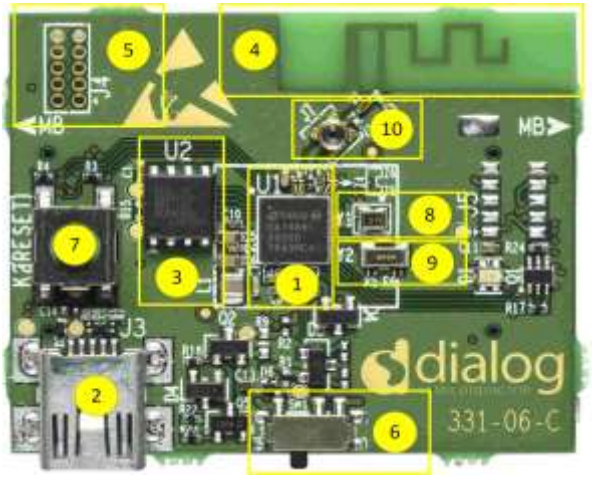

Figure 4: The DA14695 daughter board Layout

Table 2: DA14695 Daughter Board main elements

Reference	Description

1	U1: DA14695 in micro-BGA package
2	USB connector, alternative power option and connecting to DA14695 USB pins
3	U2: QSPI Flash (MX25U3235F :32Mbit)
4	Printed RF antenna
5	J4: Debug
6	Power Select
7	Reset
8	Y1 : Cristal 32 Mhz
9	Y2 : Cristal 32.768 KHz
10	J7: Coaxial switch for conducted RF measurements

NOTE

The daughter board has a switch (reference 6 in The DA14695 daughter board Layout) to select the power supply for the device:

- Power switch pos1 (left-default): VBAT LDO (default, this is the only option where the current is monitored by Studio - Toolbox)
- Power switch pos2 (right): Battery, Li-Ion (default) or Coin Cell

NOTE

The daughter board has the possibility to operate stand-alone (without the main board), powered from one of these options:

- Li-Ion/LiPo/coin Battery
- USB connector
- CIB interface (providing also JTAG/UART debugging functions)

5.3 Configuring the Pro Kit main Board by Jumper Settings

The jumper settings are displayed below:

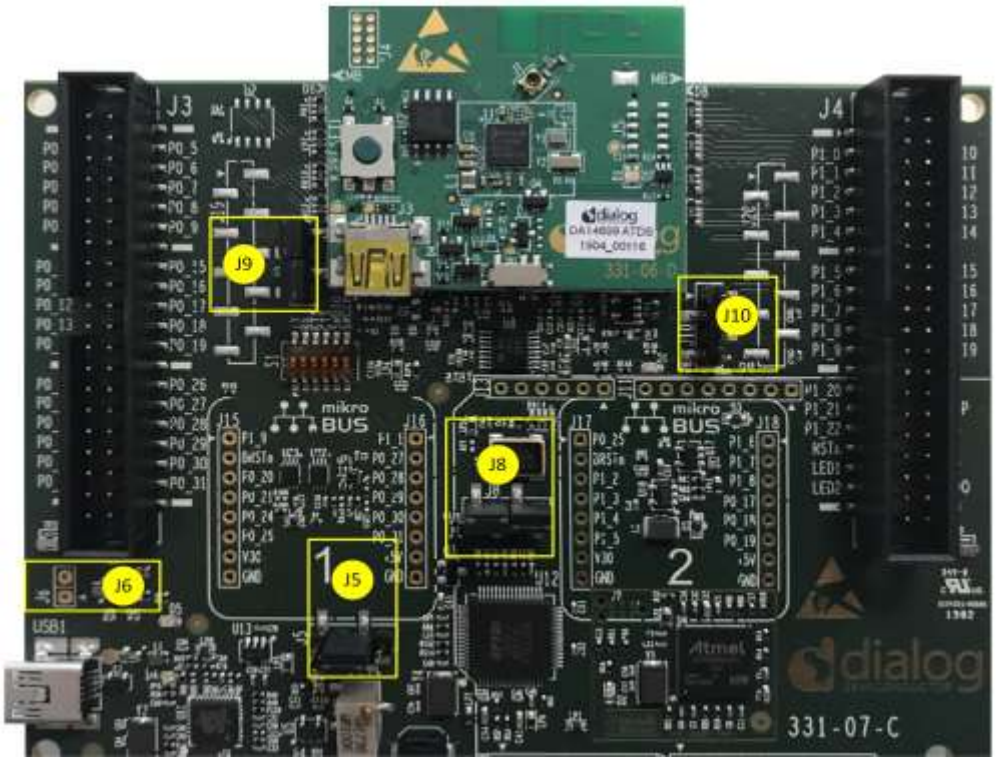

Figure 5: ProDK Development Kit Jumper setting

Table 3: Default jumper settings

Jumper	Default Position	Comment
J5	2-3	Selects 3.0V as default VBAT
J6	Not Placed	2-pin connector for Li-Ion/LiPo Battery/
J8	1-2 & 3-4	button (K1) and C_TRIG
J9	1-2 & 3-4	Current measurement input and output
J10	no jumper caps	Haptic driver outputs and ground

5.4 Connecting the ProDK to the host PC

The ProDK Development Kit allows functional verification of the DA14695 family of devices. It supports connecting external hardware by exporting DA14695 pins to standard headers and enables the user to do precise power measurements through the integrated power measurements circuitry. Additionally.

The ProDK motherboard also includes:

- An embedded J-Link debugger **U4** SAM3U2CA which ensures the USB to JTAG function by loading the software from Segger to the ROM of U4. The chip is supplied with a 3.3V from **U14** which is enabled by **PWR_ENABLE** signal.
- FTDI chipset **U12** FT2232HL which allow easy communication with the development host over USB. The FT2232HL implements the USB to UART function and ensure the connectivity of PC to the DA14695 SoC UART port and to current sense circuitry through SPI connection with ADC. The chip is supplied with a 3.3V from **U14**. A 12MHz crystal **Y4** is required for the chip operation.

The ProDK Development Kit is connected to the host PC over the connector marked as USB1, as shown in reference 6 in **Figure 2** and **Figure 6** using a standard mini-USB cable. For further reading

you can refer to the Application Note: AN-B-061 DA1469x Application Hardware Design Guidelines and UM-B-093 The hardware of DA1469x Development kit.

NOTE
Before connecting the ProDK Development Kit to the host PC make sure that: • The main board and the desired daughter board module are properly connected. • The power switch on the daughter board is in the left position (VBAT) to select USB from motherboard.

Figure 6: ProDK Development Kit connections

6 DA14695 Connecting the Board

6.1 Introduction

This Section describes the installation procedure for the drivers, the configuration of the serial port, and all necessary steps to verify the connection with the PC as well as solutions to any problems that may occur.

6.2 Requirements of the Development PC

For proper evaluation and application development using the DA14695 SoC and the ProDK an external host is required. This external host must have an operating system already installed (Windows or Linux) and USB ports as described in **2**.

6.3 Driver installation

6.3.1 Microsoft Windows

On first connection to a host PC running Microsoft Windows, the system will detect several devices and will automatically install all necessary drivers. If the system is configured to use Microsoft Windows Update, this may take several minutes to complete.

When the driver installation is complete, the system displays a Microsoft windows similar to the one presented in **Figure 7**.

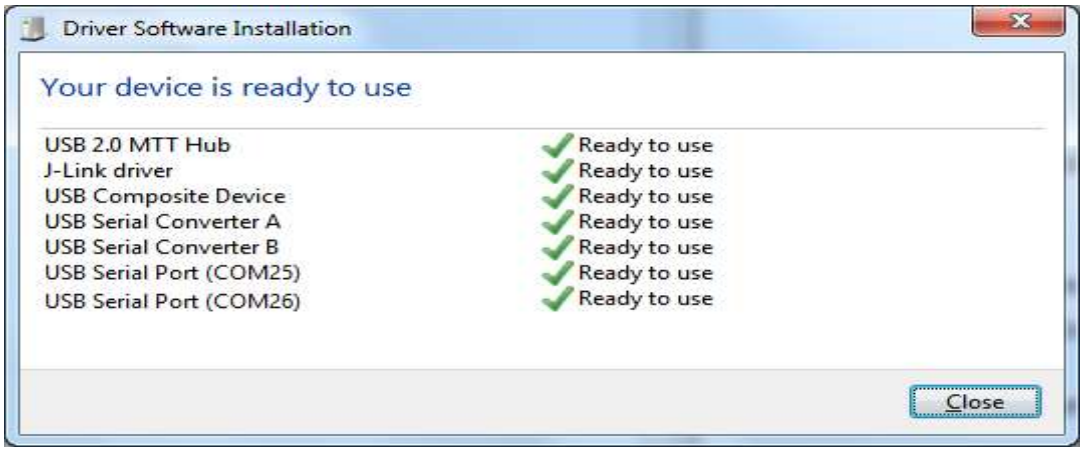

Figure 7: Windows driver installation

There are two virtual COM ports created by the Windows driver. The first COM port (lower number, COM25 in this example) provides a UART interface between the PC and the DA1469x device. The second (higher number, COM26 in this example) is used to export measurement data from the current sense circuitry on the ProDK Kit to the Power Profiler tool. For more information on the Power Profiler, see the SmartSnippets Toolbox User Manual (UM-B-083).

NOTE
The COM port numbers assigned to the ProDK Kit motherboard might be different to the ones shown in **Figure 7**

NOTE
If Your PC has a serial port on it, then that's the **COM1** you're seeing. Hence you can't make a communication between the ProDK Kit board and the Windows on that port. But you can do it only by using the other lowest COM port, **COM25** in this case.

The COM port numbers can be found in the Windows Device Manager **(Control Panel > Device Manager > Ports (COM & LPT))** as shown in **Figure 8**.

Figure 8: Device Manager Ports

6.3.2 Linux

When ProDK is connected to a host PC running a Linux distribution (such as Ubuntu or CentOS) and has Internet connectivity, the system will detect several devices and all necessary drivers will be silently installed. Provided that the process has properly finished, two additional devices will appear in the /dev directory under the names **`ttyUSB0`** and **`ttyUSB1`**, as shown in **Figure 9**. These names might be different in case other serial converters are connected to the system beforehand. If no other serial port converters are connected, the device that should be used with the terminal or programmer utility will be called **`/dev/ttyUSB0`**. If there are more devices with the name **`ttyUSBx`**, note which ones showed up when the ProDK was connected and use the lower number of the two devices.

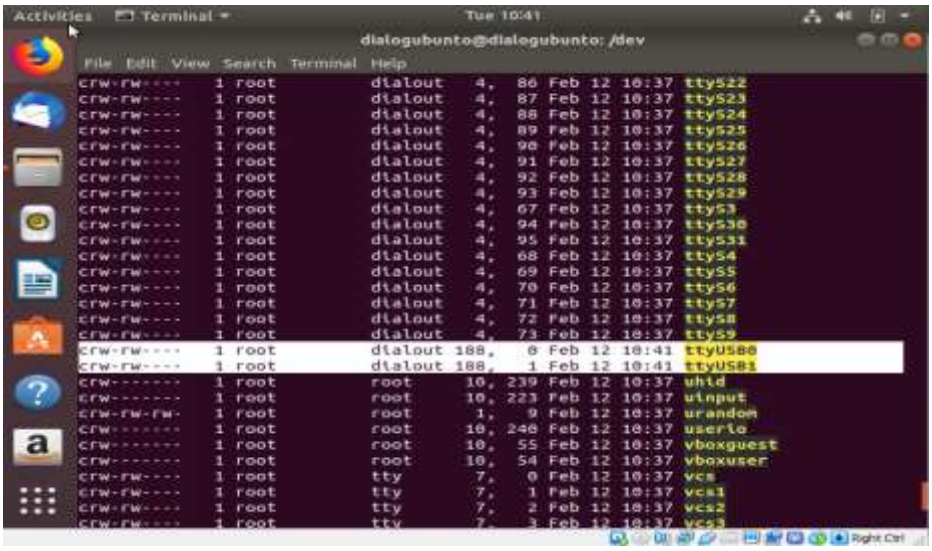

Figure 9: Ports assigned to ProDK

6.3.3 COM port usage

There are two virtual COM ports created by the driver with either Windows or Linux. The first (lower number) is used to export a UART from the DA14695 device. In the previous sections this was either COM5 or **`/dev/ttyUSB0`**. The second (higher number) is used to export measurement data from the current sense circuitry on the ProDK to the Power Profiler tool.

6.4 Configuring the serial port for UART2

Several development tools require UART2 to be routed to the FTDI serial port. Please refer to *Development Kit Pro User Manual* for details on how to properly configure the specific port. ProDK board connection verification can be made using the pre-existing Terminal application.

6.4.1 Windows Host

On a Windows Host the utility Tera Term can be used to fully validate the connection to the ProDK.

Tera Term is a free software terminal emulator (communication program) which supports multiple communication including Serial port connections. Download Teraterm from https://ttssh2.osdn.jp. Run the teraterm-x.yy.exe and follow the installation wizard.

To make sure that the communication between the ProDK board and the development host is properly established, it is necessary to verify the UART connection between the two nodes. To do so, execute the following steps:

- **Step 1** Connect the ProDK board to the PC board via USB cable to USB2(DBG) as described in **Figure 6**.
- **Step 2** Verify that the host discovered two serial ports – the first is connected to UART2 (see **6.3.1**).
- **Step 3** Open Tera Term from the Windows Start menu.
- **Step 4** In the **Tera Term: New connection** dialog, select **Serial**, then select the **COM Port** to use, and click **OK**.

- **Step 5** Select Setup > Serial Port a as shown in Figure 10 and configure your UART port using the parameters as shown in **Table 4**.

Table 4: Parameters for connecting to UART2

Settings	Values
Baud rate	**115200**
Data bits	**8**
Parity	**None**
Stop bits	**1**
Handshaking	**None**

Warning
To get the control of the COM port You need to be an administrator on your local machine.

- **Step 6** Open the the Lowest COM port number assigned to the ProDK, refer to **Figure 9** to figure out which port number is used by Windows by running the Windows Device Manager. Make sure that the UART is configured as shown in **Table 4**. Once you have a connection, you should start to see something as shown in **Figure 10**. The DA1496X device use random BLE addresses, new address is generated on every reset button push.

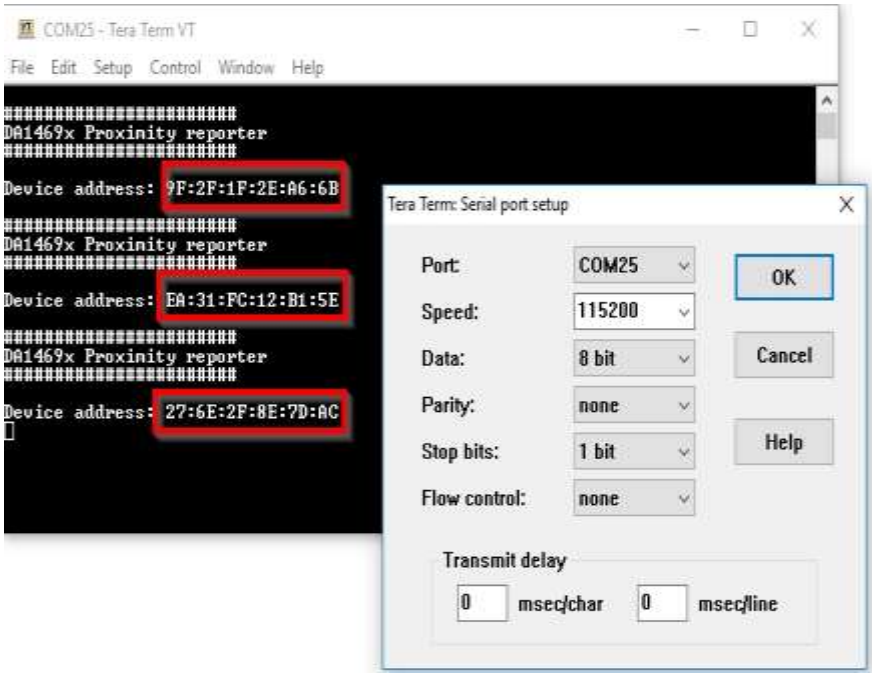

Figure 10: Terminal output via Tera Term (Windows)

6.4.2 Linux Host

Under Linux there is a simpler approach to validate the connection using a basic terminal such as putty. Connect putty to **`/dev/ttyUSB0`** at 115200 baud using this linux command sudo **`putty /dev/ttyUSB0 -serial -sercfg 115200,8,n,1,N`** or run **`putty`** as in shown in **Figure 11.**

Figure 11: Setting port and testing connectivity in Linux

Once you have a connection, you should start to see something as shown in **Figure 12**. This is the output under Linux with putty.

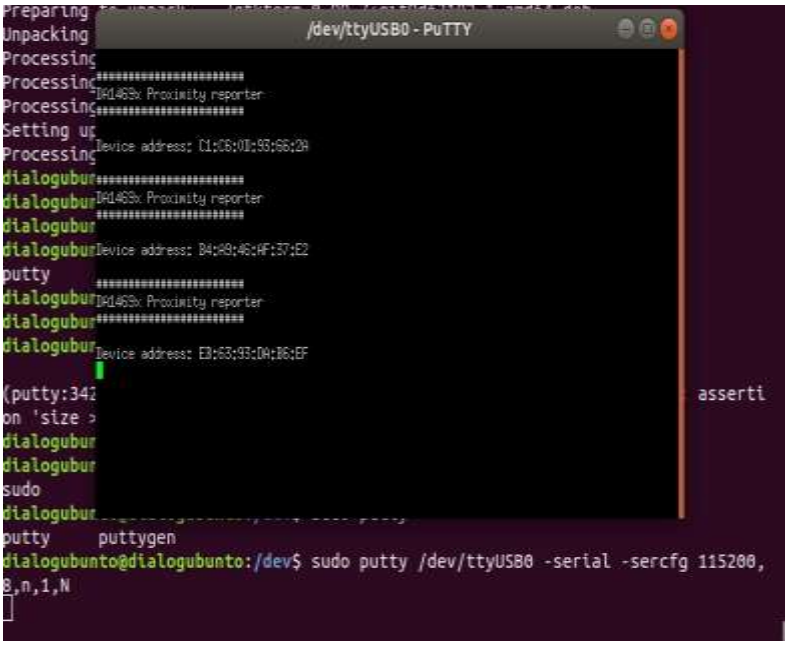

Figure 12: Terminal output via putty (Linux

6.5 Troubleshooting

If there any problems with the ProDK connection to PC some possible solutions might be:

- Make sure that the Host PC is connected to Internet
- Make sure that no old FTDI drivers are installed. Drivers are available from the FTDI website.
- Check for possible cabling issue by using a different USB cable
- Connect the two elements using a different USB port on the host PC

Note:

NOTE
If none of these actions resolved the issue, please contact Dialog Software Forum.

7 Software Development Tools

7.1 SmartSnippets™ Introduction

Dialog SmartSnippets™ Studio is a royalty-free software development platform for Smartbond™ devices. It fully supports the DA1469X family of devices.

SmartSnippets™ Studio contains:

- SmartSnippets™ IDE: Eclipse CDT based IDE with pre-configured plugins to provide the build/debug environment
- SmartSnippets™ DA1469X SDK
- SmartSnippets™ Toolbox: A tool suite covering all software development requirements, including:
- Programming and loading of firmware into integrated RAM, 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/.

7.2 SmartSnippets™ Studio Installation and Starting

This section describes the installation of SmartSnippets™ Studio. **For further reading** the installation procedure is described in detail in UM-B-057 SmartSnippets Studio User Manual.

NOTE
• The SmartSnippets™version should be 2.0.6 and above if you wish to use Eclipse/GCC • Registration is required in order to download the SmartSnippets™.

Warning
Please be aware that if you have an Antivirus software is installed in your machine, it could slow down the SmartSnippets™ installation due to the scan.

7.2.1 Windows

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

- Select the recommended version or a newer one from of SEGGER J-Link GDB server and Click **Next**.

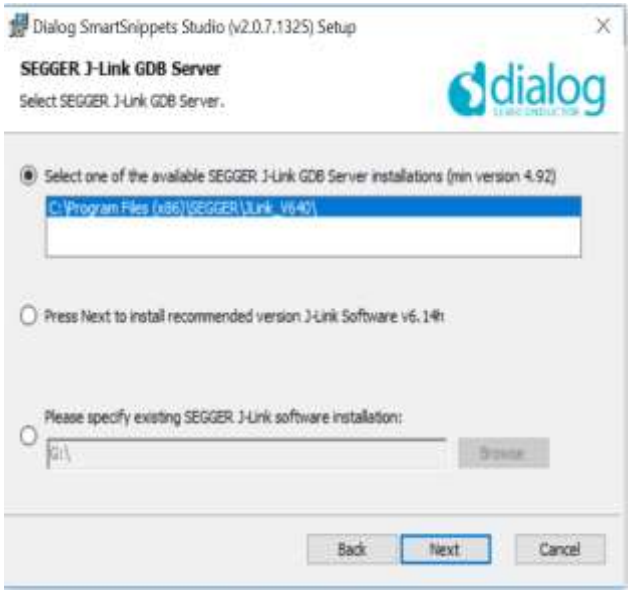

Figure 13: Automatically install J-Link

- Select the destination folder for the SmartSnippets™ Studio and click "Next". The default installation location `C:/DiaSemi` is recommended

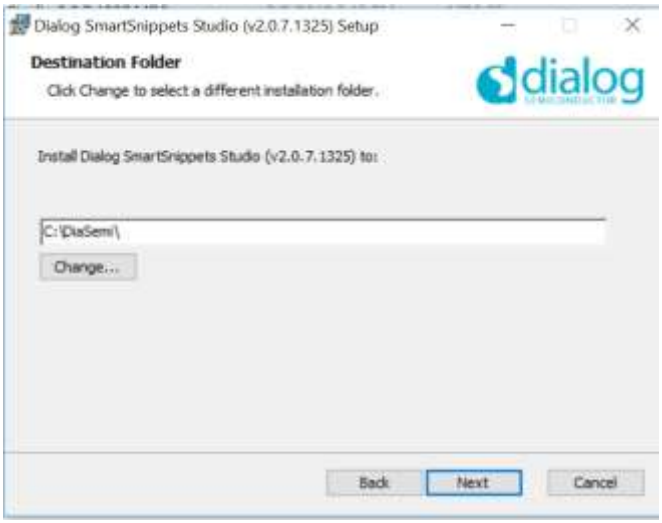

Figure 14: Select Smart Snippets Studio install directory

- The installation of the SmartSnippets™ Studio will then start.
- After the successful installation of the SmartSnippets™ Studio next screen will appear.

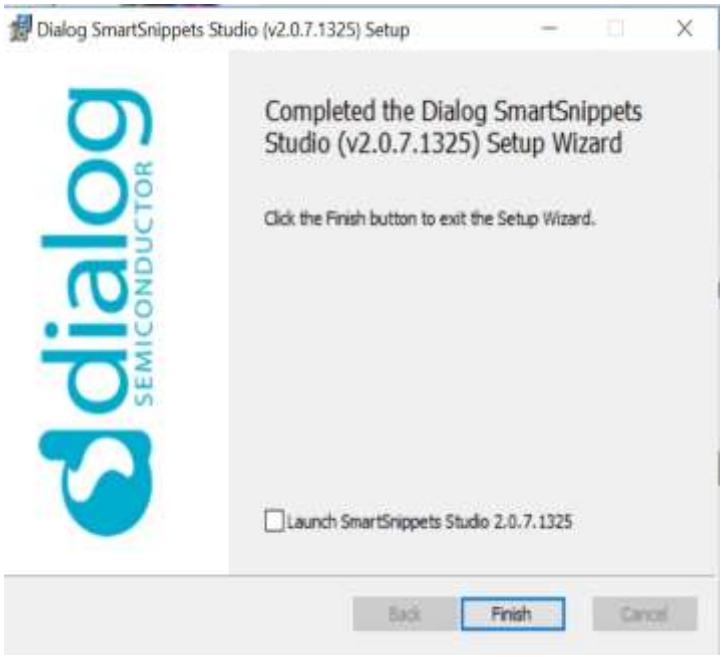

Figure 15: Smart Snippets Studio installation completed

- Click Finish and wait for the SmartSnippets™ Studio to start.

When SmartSnippets™ Studio starts for the first time, the user must configure it. The necessary configurations are the following:

- Select the workspace folder for SmartSnippets™ Studio. This should be the created directory inside **`workspace_SmartSnippets_Studio`**.

- All the other tools required by the SDK and SmartSnippets™ Studio will be automatically installed, such as GNU ARM GCC, SmartSnippets Toolbox, SEGGER drivers etc. On a clean PC they will probably not be there and so the next steps are to download and install them.

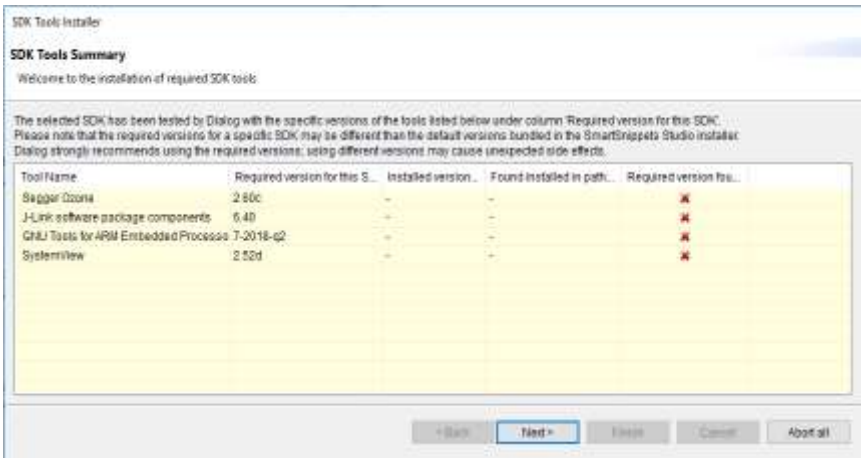

Figure 16: SDK Tools Summary

- Tick the **download and install** radio button and then press **Download** button for SEGGER Ozone tool
- This will install the tool to **`C:\Program Files (x86)\SEGGER\Ozone V2.60c`**. Then press **Next**.

NOTE
• Users are required to download and install tools marked as mandatory by clicking the download button or specifying an installation folder if the tool is already installed externally. Some tools can be optional, in which case Skip button will show • The **Next** button is enabled only after the mandatory tools are successfully installed or a valid installation path is specified.

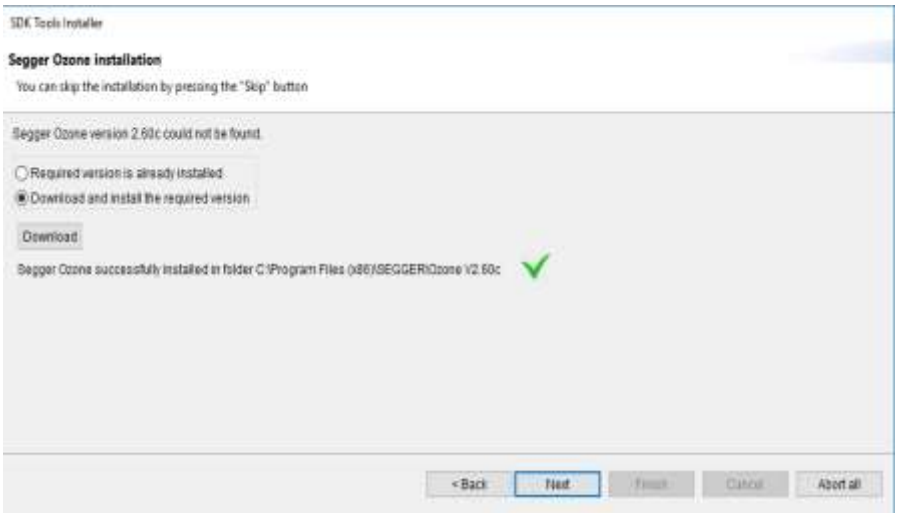

Figure 17: Start Ozone download

- Then, the installer will guide user to install SEGGER JLink version 6.42b or later. They are available from the SEGGER website Refer to **Figure 18.**

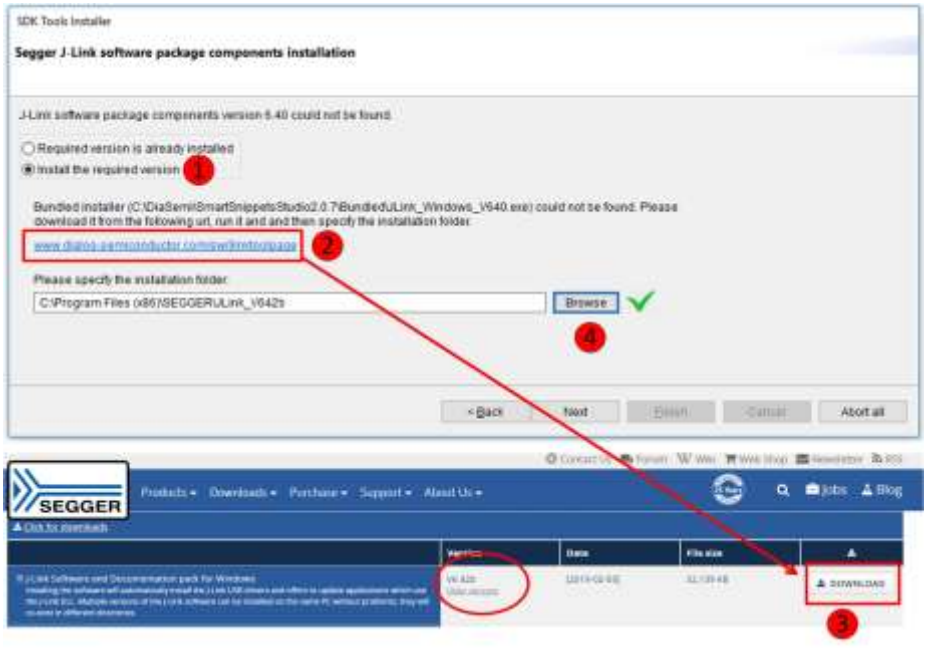

Figure 18: Set SEGGER J-Link Installation Folder

- The SDK Tools Installer will prompt for the installation of GCC. Select **Download and install** the required version and click **Download**. After the download is complete, the green completion mark appears. Then press the **Next** button to continue as shown in **Figure 19**.

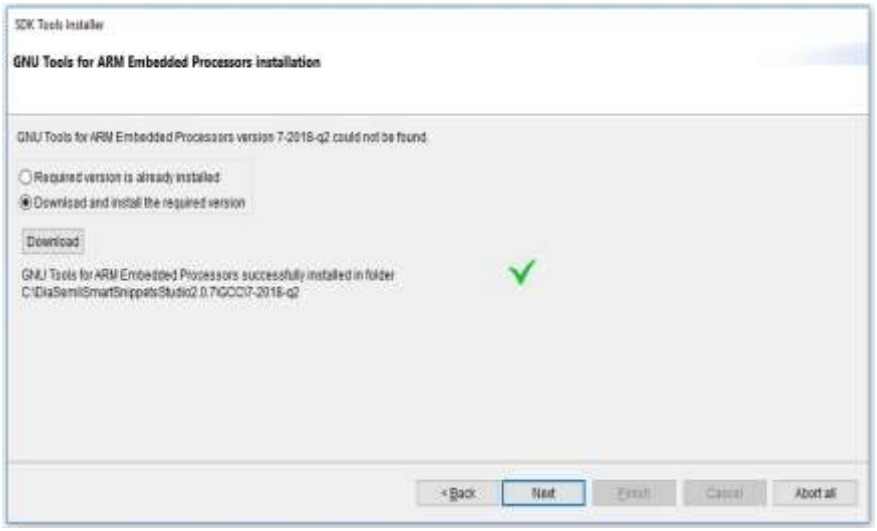

Figure 19: GCC Tools Installation

- The SDK Tools Installer will prompt for the installation of SEGGER SystemView, Refer to **Figure 20.**

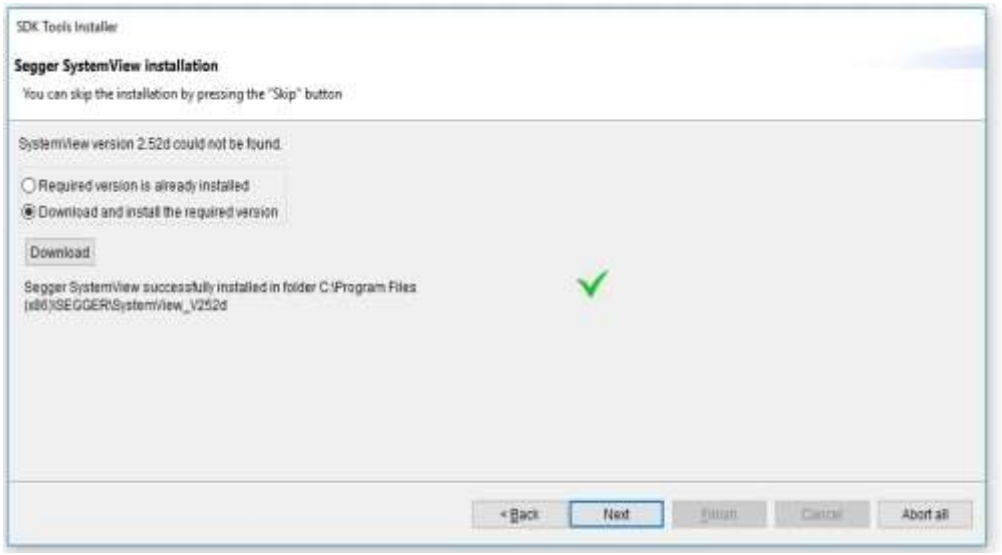

Figure 20: SEGGER SystemView Tool Installation

- The last page of the wizard shows a summary of all installed tools. 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.

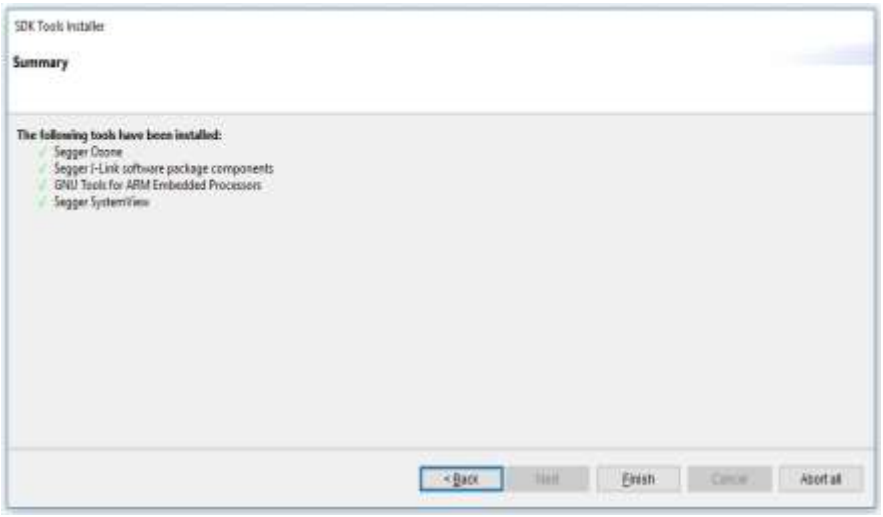

Figure 21: Installed tools summary

7.2.2 Linux

Before running the installer, it is necessary to install some Linux packages that are mandatory for the execution of SmartSnippets™ Studio. Without them SmartSnippets™ Studio will not run correctly but will fail with no reported error.

- For **CentOS 7:**

```
sudo yum install install epel-release
sudo yum install webkitgtk.x86_64
sudo yum install glibc.i696 ncurses-libs.i696
sudo yum install qt-x11 (required for SystemView tool)
```

- For Fedora 25:

```
sudo yum install webkitgtk.x86_64
sudo yum install glibc.i696 ncurses-libs.i696
sudo yum install gcc-c++
sudo yum install libncurses.so.5
```

- For Ubuntu 16.04.1 install:

```
sudo apt-get install libz1:i386 libncurses5:i386 libbz2-1.0:i386
sudo apt-get install libwebkitgtk-1.0-0 libwebkitgtk-3.0-0
sudo apt-get install gawk
```

- The first step is to make the SmartSnippets™ Studio installer executable.
- `$chmod a+x SmartSnippets_Studio-linux.gtk.x86_64-1.6.3.run`
- And then run it.
- `$./SmartSnippets_Studio-linux.gtk.x86_64-1.6.3.run`
- Several of the required tools will automatically install and others need to be manually downloaded and installed.

- Install the recommended version of SEGGER J-Link GDB server. The V6.14h version will be installed by default. If you have recent SEGGER J-Link version you have just to select the path where it is already installed, as shown in **Figure 26.**

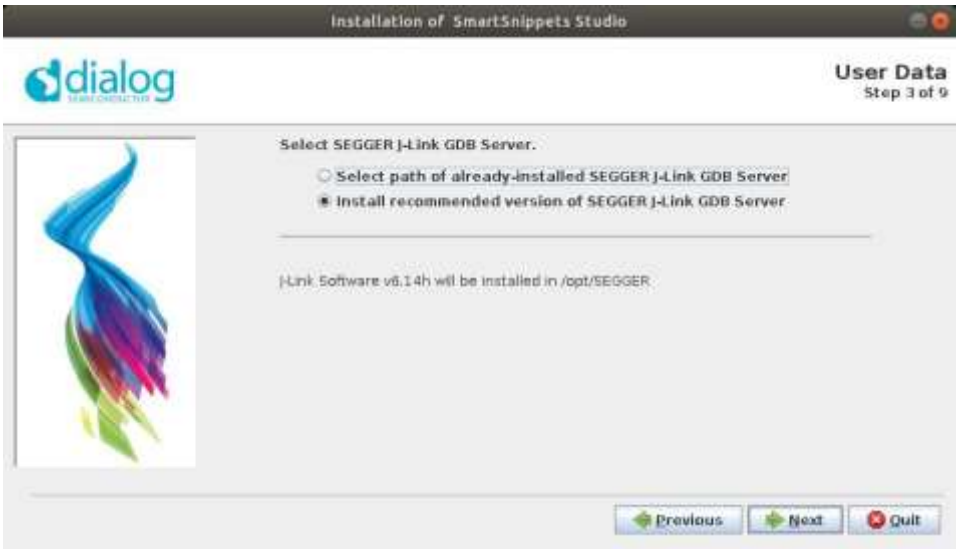

Figure 22: Automatically install the J-Link

- Select the destination folder for the SmartSnippets™ Studio

Figure 23: Select SmartSnippets™ Studio install directory

- The installation of the SmartSnippets™ Studio will then start.
- After the successful installation of the SmartSnippets™ Studio click Finish on the next screen that will appear and wait for the SmartSnippets™ Studio to start.

- When SmartSnippets™ Studio starts it will check which SEGGER tools are installed. On a clean PC they will probably not be there and so the next steps are to download and install them

Figure 24: SDK Tools Summary (Linux)

- Tick the **download and install** radio button and then press **Download** button for SEGGER Ozone tool. The download will then start and the SEGGER Ozone tool will be installed automatically. This will install the tool to `/opt/SEGGER/ozone/2.60c`. Use the **Browse** button to find this folder and then press **Next**.

Figure 25: Set the Ozone installation directory

- Then, the installer will guide user to install SEGGER JLink.

Figure 26: Set SEGGER J-Link Installation directory

- Then, the installer will guide user to install GCC. **Browse** button to find this folder and then press **Next**.

Figure 27: GCC Installation directory

- The final stage is to install SEGGER SystemView.

Figure 28: SEGGER SystemView Installation directory

The last page of the wizard shows a summary of all installed tools. 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.

Figure 29: Installed tools summary under Linux

7.3 Package structure and Folders

The directory structure of the SDK is shown in **Figure 30.**

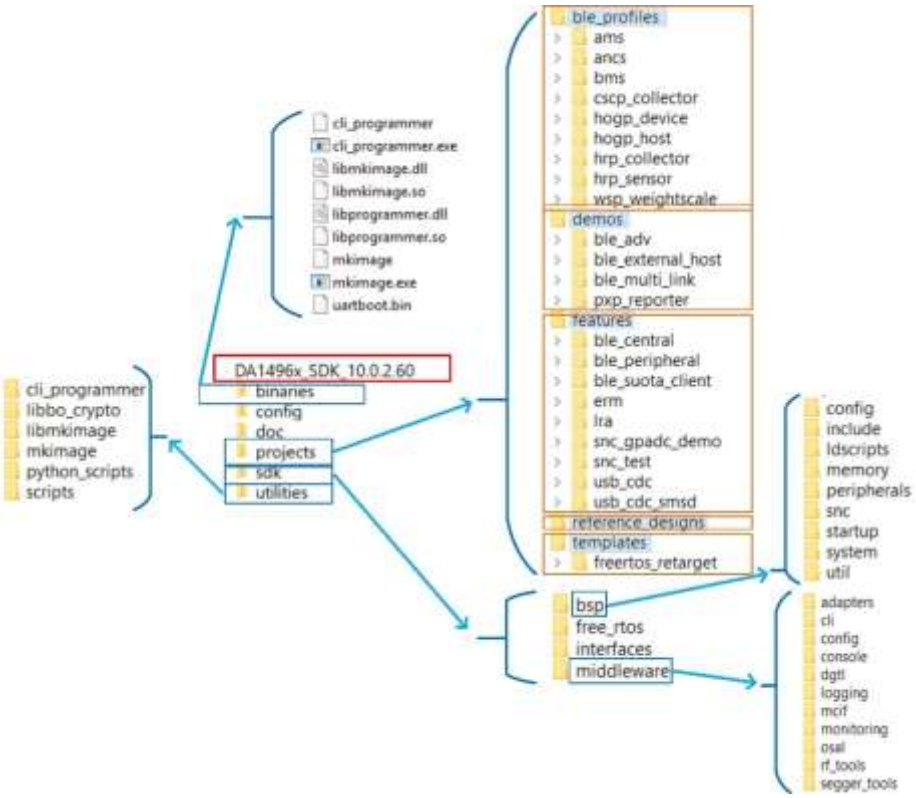

Figure 30: Software SDK Directory Structure (on Windows)

7.4 Extracting and using the SDK

As SmartSnippets™ Studio is an Eclipse based Integrated Development Environment (IDE) all the source files are contained within a workspace folder which contains all the project sources, build configurations etc.

In the user directory (Windows or Linux) create a directory called:

`workspace_SmartSnippets_Studio`.

Extract the contents of the SDK zip file to the workspace folder.

Warning
When selecting the SDK directory in the SmartSnippets™, please make sure to select the level with the name (i.e. DA14695_SDK_10.x.y.z) otherwise the projects will not build. It must contain the contents of the directory as shown in **Figure 30**. This shows a Windows file system; the principle is the same for a Linux filesystem.

As the workspace is the same directory as the SDK it means that all edits to files in the workspace change the contents of the SDK. This means that all projects should be developed in different workspaces and so each workspace needs to be created with an extraction from the original SDK zip file as there is no longer a clean copy of the SDK on which to base new projects.

NOTE
In case the selected workspace does not correspond to a valid SDK and the user selected the “Take no action” option in the dialog asking how to treat the selected workspace Figure 31. Different level needs to be selected.

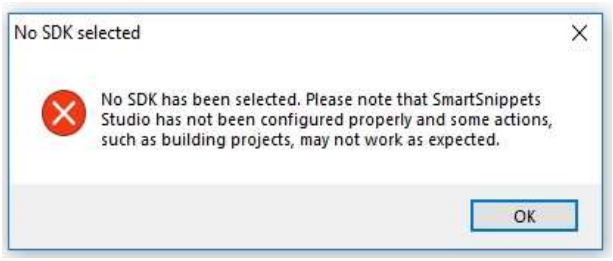

Figure 31: No SDK selected error popup message

7.5 Additional Software

The SmartSnippets™ Toolbox, Figure 32, is focused on enabling the process of programming flash and optimizing code for optimal power performance by allowing:

- The re-programming of the internal QSPI with the actual application compiled image.
- An accurate examination of the power profile and the effects of any executed application software.
- The seamless download and execution of a certain software image to RAM over UART.

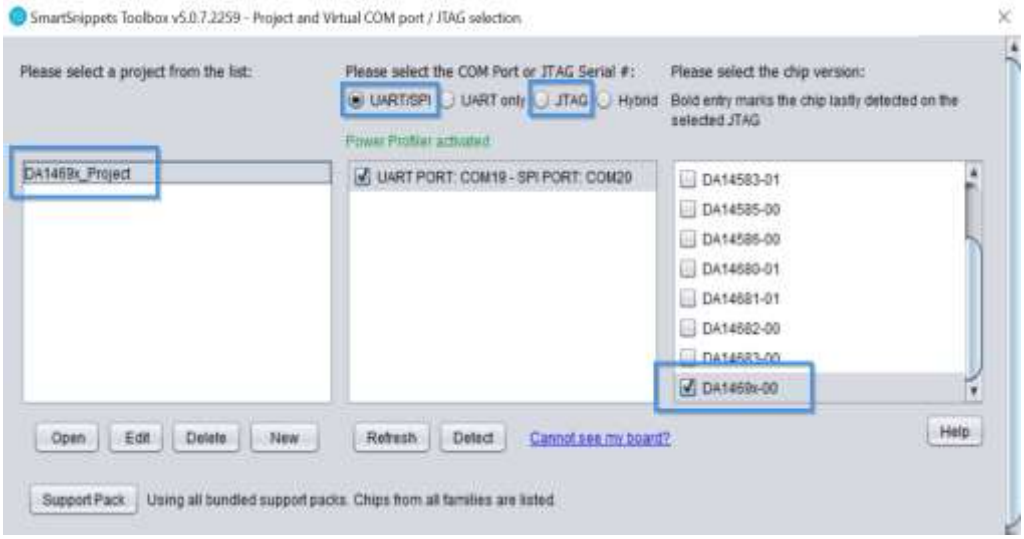

Figure 32: SmartSnippets™Toolbox : Mode to download the application

SmartSnippets™ Toolbox is also supported by other utilities such as the Command Line Interface (CLI) Programmer. The CLI Programmer is a command line tool for programming the DA14695 family of devices. It allows erasing and programming the device Flash or OTP memory. This tool may be used both in development and on the production line. The CLI Programmer will be installed as an integrated part of the SmartSnippets™ DA14695 SDK.

The SmartSnippets™ framework makes maximum use of the available features on the motherboard like the on board current sensing circuitry to allow developers of Bluetooth applications to work without expensive and bulky equipment such as a Digital Multi Meter (DMM). The tool provides full visibility on the chip activity, which is crucial in the developing of ultra-low power wireless applications.

8 Run the Pre-Loaded Demo

This section describes how to quickly get started with the DA1469x ProDK and run the preloaded demo.

- **Step 1:** Take the daughter board and Plug it to the main board through the dedicated connector **Figure 3**. Take care to match the orientation.
- **Step 2:** Connect a USB type A to mini-B USB cable on the micro USB header on the ProDK to power On. The other side of the USB cable can be connected to a PC/Laptop. **Figure 6**
- **Step 3:** After the ProDK is powered, and If everything works fine then the following LEDs status can be seen as shown **Figure 33.**
  - On the daughter board, Red LED 1 is blinking
  - On the main board Blue LED D1, Green LED 4 and Green LED 5 are ON.

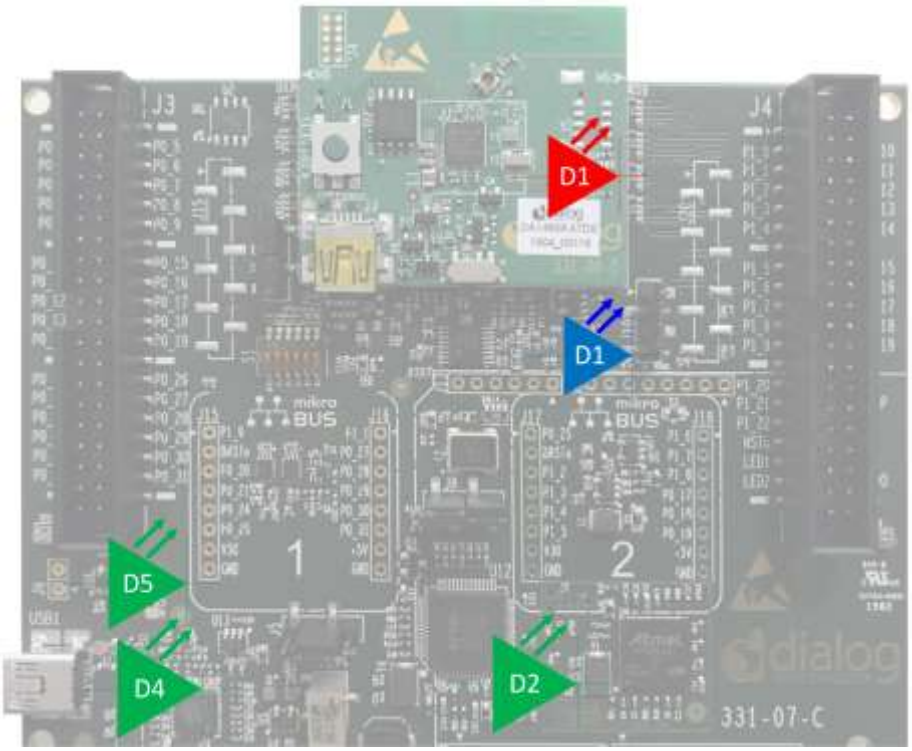

Figure 33: LEDs Status

- **Step 4:** The ProDK now starts to advertise over Bluetooth. The Pre-Loaded demo application is executed from QSPI flash. Open your BLE scanner application to scan and connect to the Dialog DA14695 device. Refer to **Figure 34.**

Figure 34: Pre-Loaded Demo: Interacting with BLE Application

NOTE
In this example, we used an Android application, but you can also use the LightBlue iOS application to connect an iPad/iPod/iPhone device to the application.

NOTE
The number shown in **Figure 34** is aligned with the number observed on the Terminal program **Figure 10** for windows.

- **Step 5** As Feature, alerting is done using the RED LED available on the daughter board, refer to **Figure 33**. You can write value to Alert Level characteristic in Immediate Alert service:
  - 0x00 (No alert) - the white LED on ProDK device does not blink
  - 0x01 (Mild alert) - the white LED on ProDK blinks slow immediately
  - 0x02 (High Alert) - the white LED on ProDK blinks fast immediately

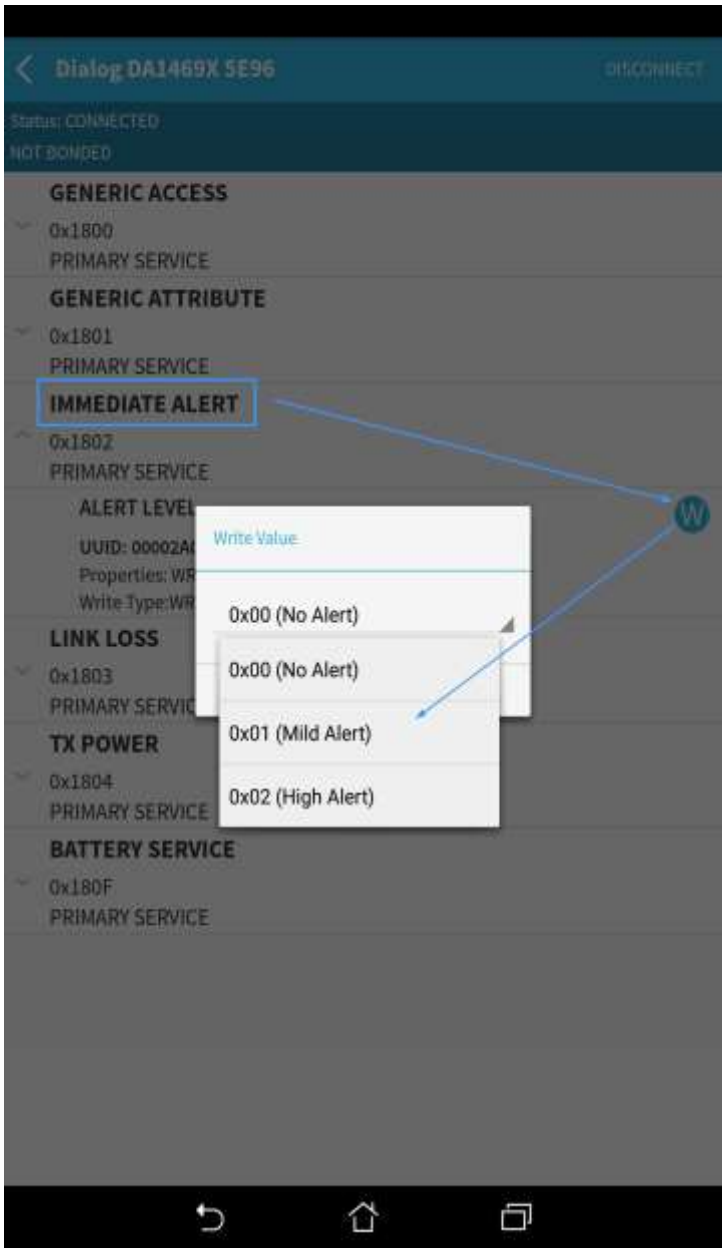

Figure 35: Pre-Loaded Demo: Alerting Manual testing

9 Building a DA14695 Application – Advertising Demo

9.1 Introduction

The following sections explain how the user can build, program and run a simple software application called Advertising Demo **`ble_adv`** on the ProDK development board using the SmartSnippets™ DA1469X SDK.

The application is first described, then step by step instructions are given to build and run it.

9.2 Software Architecture

In order to be familiar with building and executing **ble_adv** project is better to run it in debug mode first. This is the easiest setup and does not need any Flash programming prior to executing the binary.

When the application starts running the first thing that is executed is the **Reset_Handler**, which is located in **startup > startup_ARMCM0.s**. This is followed by setting **IRQ** priorities and initializing variables.

Next, code execution continues with the main subroutine in file *main.c*. Here, the main routine creates the task **SysInit** and starts the RTOS scheduler. From now on the RTOS scheduler is running and it will start the first task which is **SysInit**.

The **SysInit** first initializes the clock and the low power clock and sets the clock dividers for AHB and APB buses. Then BLE Adv is initialized which leads to function initialization of BLE manager.

Last thing done before SysInit task exits is to create another task **ble_adv_demo_task** which is the main application task running until the program gets stopped. The function code implementing this main task is as follows:

Code 1 *The main task in SysInit() the ble_adv_demo_task()*

```
static void ble_adv_demo_task(void *pvParameters)
{
 int8_t wdog_id;

 /* Just remove compiler warnings about the unused parameter */
 ( void ) pvParameters;

 /* Register ble_adv_demo_task to be monitored by watchdog */
 wdog_id = sys_watchdog_register(false);

 /* Start BLE device as a peripheral */
 ble_peripheral_start();

 /* Set device name */
 ble_gap_device_name_set("Dialog ADV Demo", ATT_PERM_READ);

 /* Set advertising data */
 ble_gap_adv_data_set(sizeof(adv_data), adv_data, 0, NULL);

 /* Start advertising */
 ble_gap_adv_start(GAP_CONN_MODE_UNDIRECTED);

 for (;;) {
         ble_evt_hdr_t *hdr;

         /* Notify watchdog on each loop */
         sys_watchdog_notify(wdog_id);

         /* Suspend watchdog while blocking on ble_get_event() */
         sys_watchdog_suspend(wdog_id);

         /*
          * Wait for a BLE event - this task will block
          * indefinitely until something is received.
          */
         hdr = ble_get_event(true);

         /* Resume watchdog */
         sys_watchdog_notify_and_resume(wdog_id);

         if (!hdr) {
                 continue;
         }

         switch (hdr->evt_code) {
         case BLE_EVT_GAP_CONNECTED:
                 handle_evt_gap_connected((ble_evt_gap_connected_t *) hdr);
                 break;
         case BLE_EVT_GAP_DISCONNECTED:
                 handle_evt_gap_disconnected((ble_evt_gap_disconnected_t *) hdr);
```



```
                break;
        case BLE_EVT_GAP_PAIR_REQ:
                handle_evt_gap_pair_req((ble_evt_gap_pair_req_t *) hdr);
                break;
        default:
                ble_handle_event_default(hdr);
                break;
        }

        /* Free event buffer (it's not needed anymore) */
        OS_FREE(hdr);
      }
}
```

9.3 Software Build

This section describes all the steps required to import, build and run this first project.

1. In the SmartSnippets™ Studio welcome page click on the IDE icon from the Tools tab as shown in **Figure 6**

Figure 36: SmartSnippets™ Studio welcome page

2. Import the template project : **`ble_adv`** from:

   `<sdk_root_directory>/projects/dk_apps/demos/ble_adv` into the selected workspace. Press the **browse** button highlighted in the Resources tab (reference 1) and navigate to the folder which contains the specific project as shown in **Figure 37**.

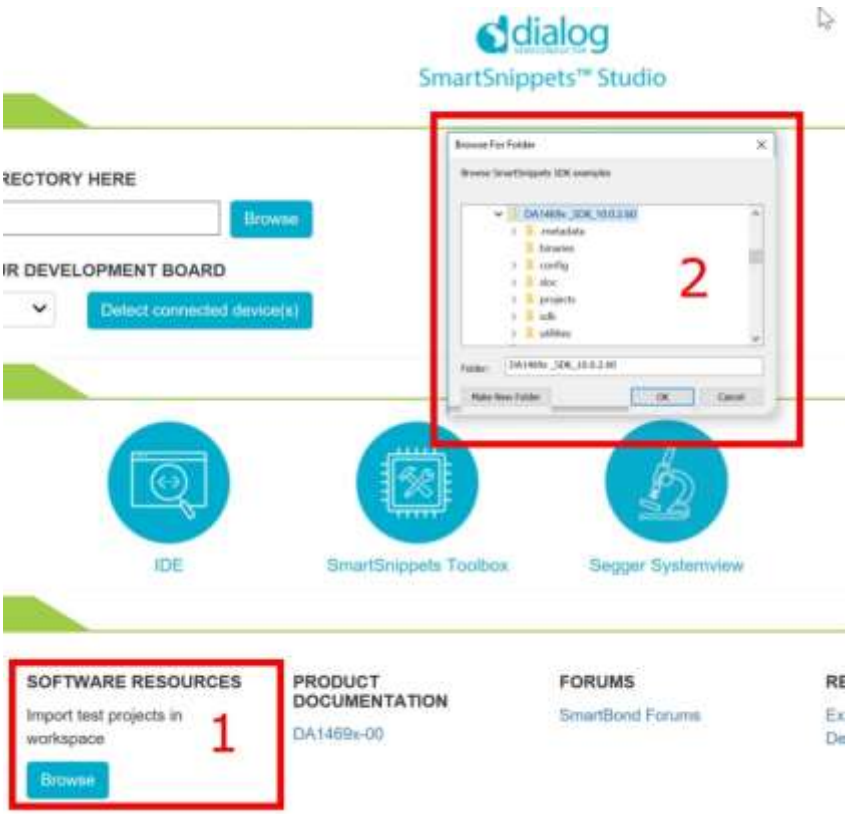

Figure 37: Project import

3. In the same way import the **python_scripts** project from:

   <sdk_root_directory>\utilities\python_scripts

9.3.1 Build the project to run from RAM

The first build to try is a RAM build. This is the simplest one as there is no need to write the code to external QSPI Flash, the debugger will load it directly into RAM from where it can be run. This is not the normal method of development.

Build the project with the **Build** button and select Debug RAM configuration **DA1469X-00-Debug-RAM** as shown in Figure 38.

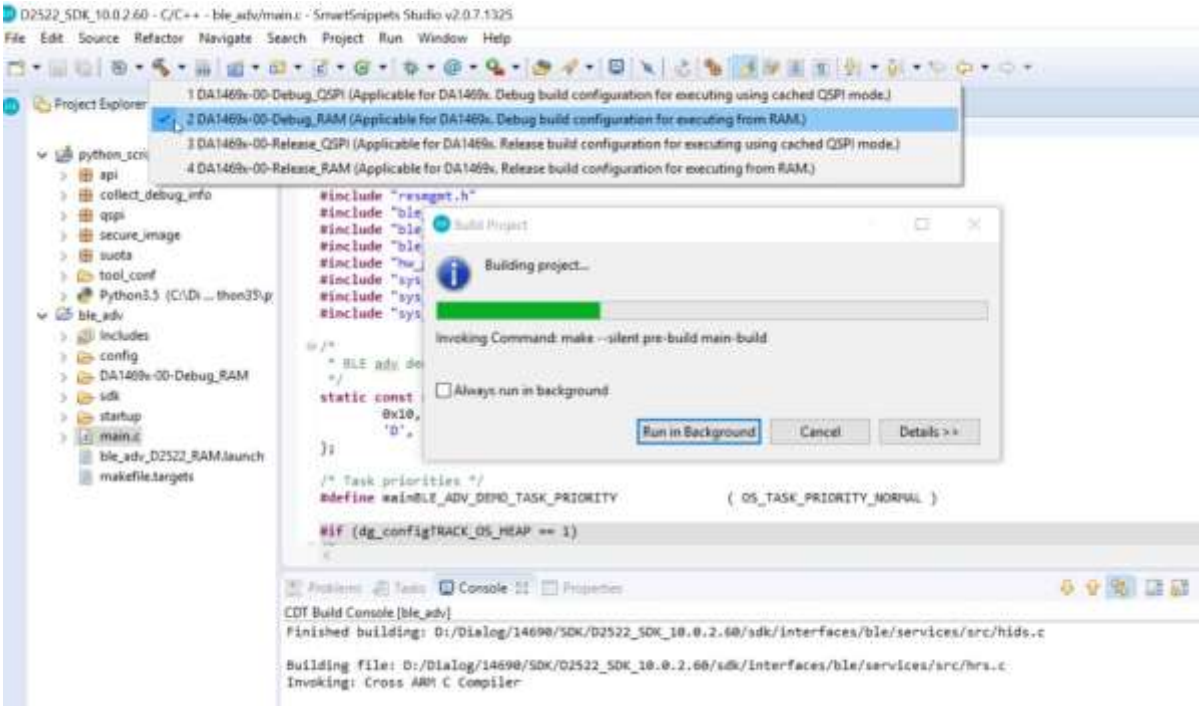

Figure 38: Build ADV BLE in Debug RAM configuration

Once the Debug RAM binary is built, the next step is to start the Debugger using as shown in **Figure 39**. As this is a RAM build the debugger will download the binary file via J-Link debugger into the system RAM. To enable this the system RAM is mapped to address 0 by the debugger.

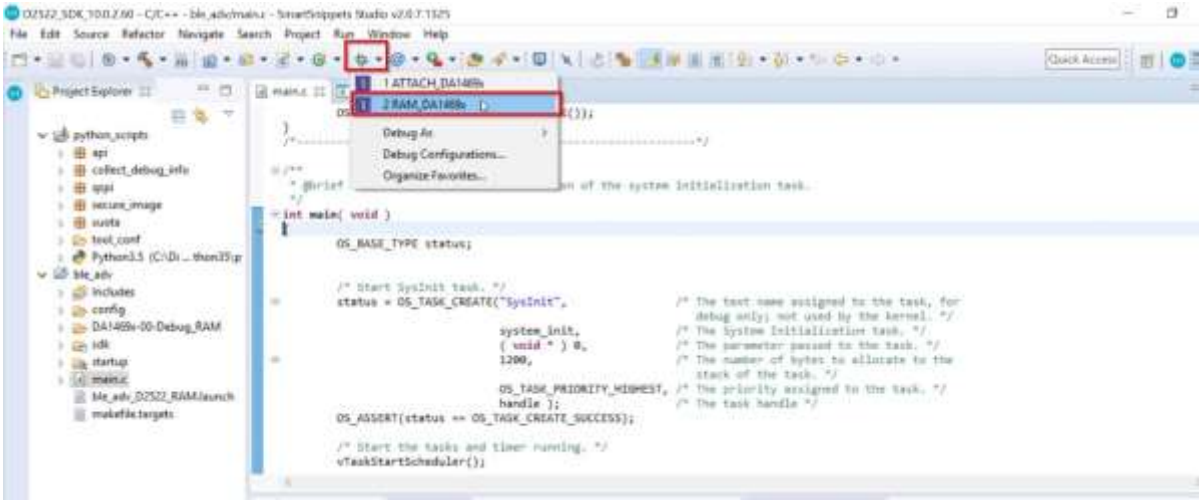

Figure 39: Start Debug in RAM mode

9.3.2 Build the project to run from QSPI Flash

1. This will be the normal development flow which has three steps: build the code, write it to QSPI Flash and then run it in the debugger. Some questions may pop-up about which Flash device etc, as explained later in the next section: **Configure SmartSnippets™ to write to Flash**
2. Build the project pressing the **build** button and select the Debug QSPI configuration **`DA1469X-00-Debug-QSPI`** as shown in Figure 40.

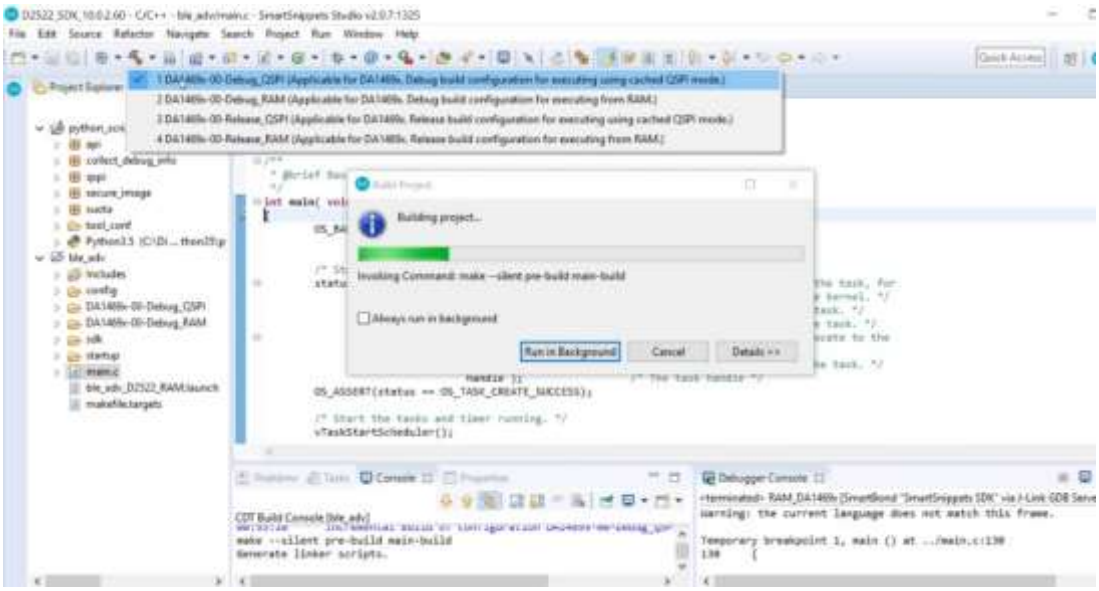

Figure 40: Build ADV BLE in Debug QSPI configuration

The next step is to write the binary file to QSPI Flash. This is done by using a script selected from the **External Tool** button. In Figure 41 select **`program_qspi_jtag`** to program the QSPI Flash memory. Alternatively, use **Run > External Tools > program_qspi_jtag**.

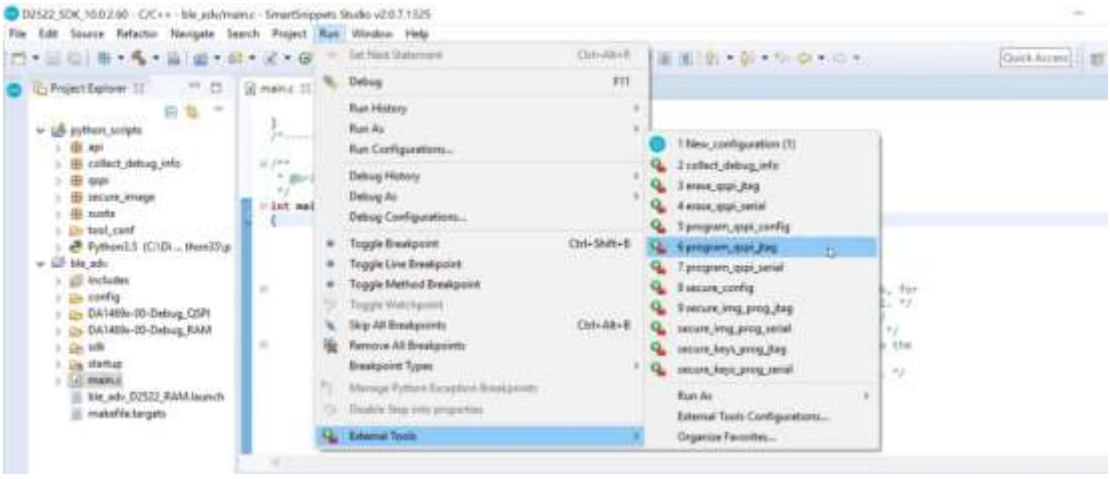

Figure 41: Write ADV BLE to QSPI Flash

Finally start the debugger as shown in Figure 42. This will start the debug perspective in SmartSnippets™ and load the symbols for the current project into the debugger.

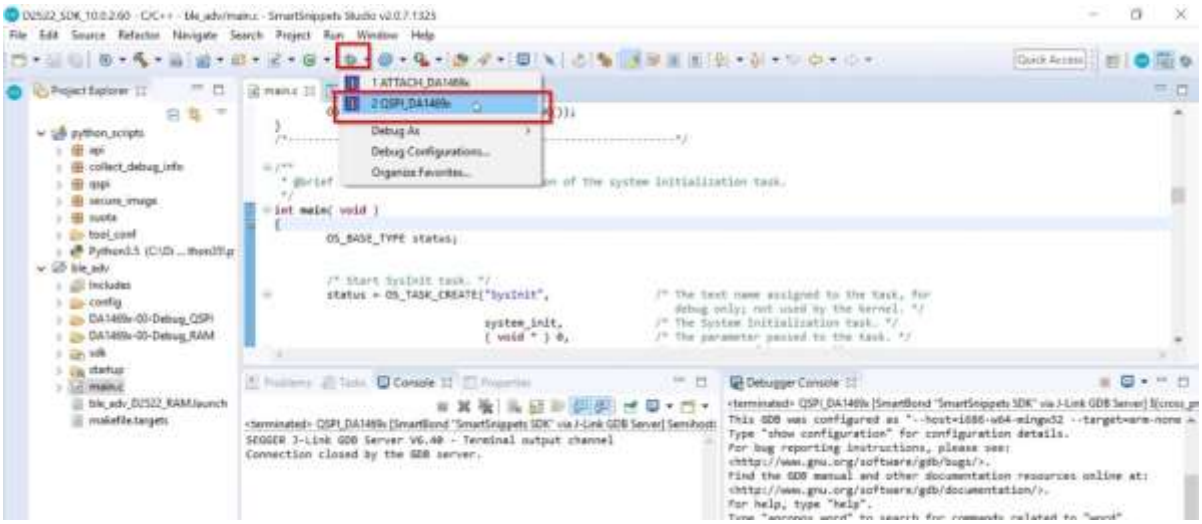

Figure 42: Start Debug in QSPI mode

9.4 Configure SmartSnippets™ to write to Flash

In order to write an image to another Flash version a configuration must be done first. To access configuration menu alternatively, use:

Run > External Tools > program_qspi_config `program_qspi_config`, refer to Figure 41. This will open the window as shown in Figure 43 with a summary of the current QSPI configuration and supported device.

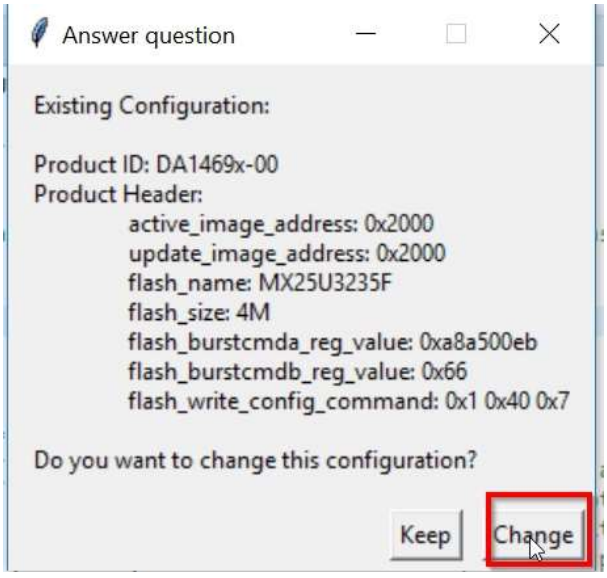

Figure 43: Configuration Summary

Press change to apply a new configuration. The first question has to do with the Product Id Select **DA1469x-00** product family

Figure 44: Select Product ID

Next you will asked about the Flash configuration. For ProDK of the DA1469x family select **MX25U3235F**.

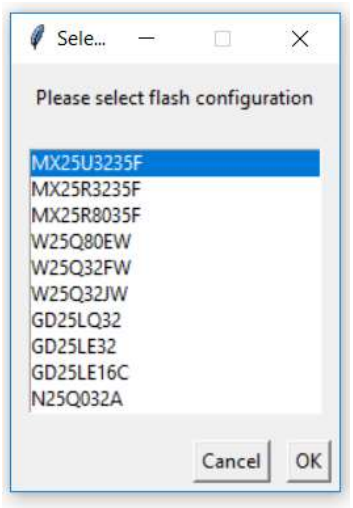

Figure 45: Select Flash configuration

Finally, you will be asked to insert an address about:

Active FW image address and **Update FW image address**. Please keep in both entries the default value **0x2000**.

Warning

Warning
Once your project is loaded and it is not working, it could that a wrong flash version was chosen. make sure to select the right flash version refer to Figure 45

9.5 Running the project in the Debugger

1. Now that the binary has been loaded to memory (either RAM by debugger or QSPI by script) and the debugger has the symbols for the project loaded it is possible to run project in the debugger.
2. Start execution of the ADV BLE project by selecting **Resume** inside the SmartSnippets™ Run menu or by hitting the **play** icon as indicated in Figure 46

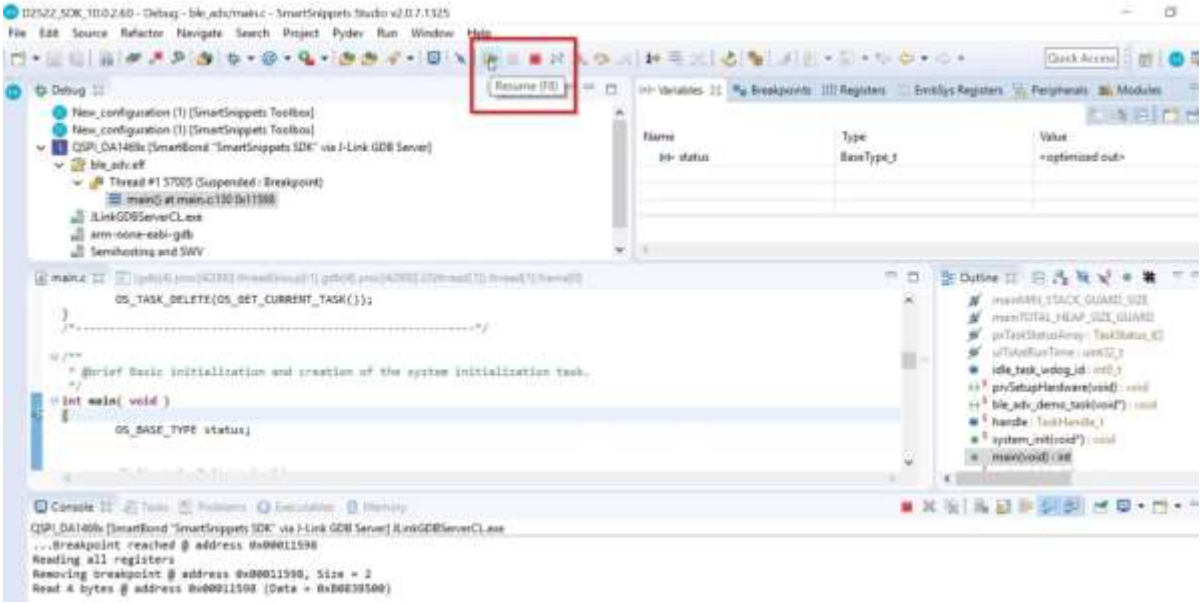

Figure 46: Executing the ADV BLE project in SmartSnippets™

The correct functionality of the ADV BLE project can be checked by noticing in the BLE scanner application.

Figure 47: ADV BLE: Interacting with BLE Application

9.6 How to Program the original FW back into the QSPI Flash

In this section we will describes how you can Program the original FW back into the QSPI Flash In This Small SW example : DA1469x_dev_kit_demo , You can built and write to the QSPI flash as it shown in: Build the project to run from QSPI Flash or you can compile it and write the built **`.bin`** to the Flash using **`<sdk_root_directory>/binaries/cli_programmer.exe`** as shown in **Figure 48.**

Code 2 *Command to Write binary to the QSPI Flash using cli_programmer*

```
./cli_programmer.exe gdbserver write_qspi 0x0 pro_kit_demo.bin
```

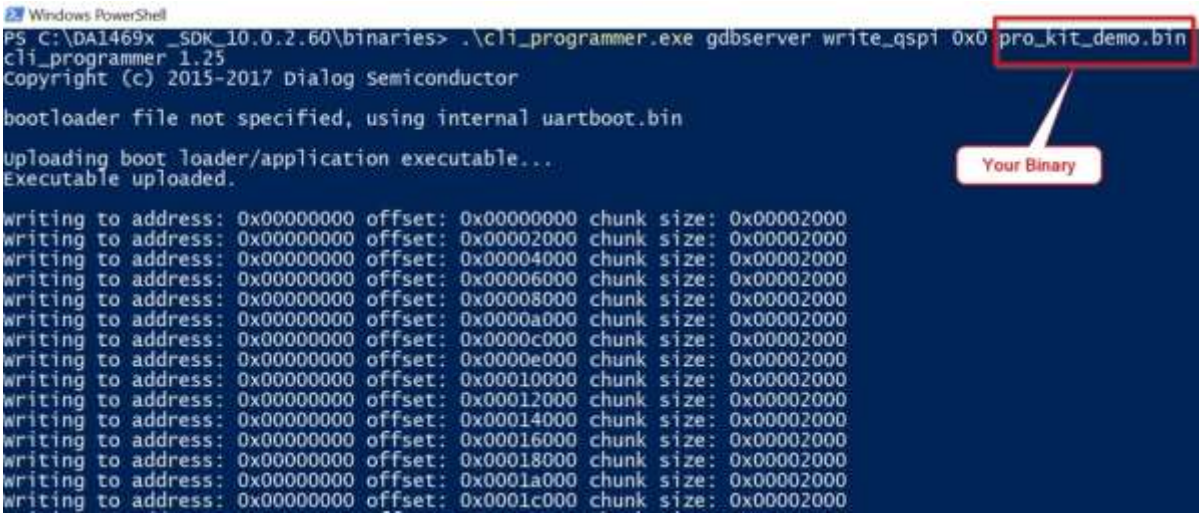

Figure 48: Write binary to the QSPI Flash using cli_programmer

9.7 What is Next ?

This tutorial does not cover all the topics relevant to software development environments, it describes the first steps necessary to get started with the Pro Development Kit. **For further reading** the following links provide more information on DA1496X:

- DA1469x Product Brief : To know more about the SmartBond™ DA1469x SoC.
- UM-B-092: DA1469x Software Platform Reference : To know more about software architecture.

10 Appendices

10.1 Appendix A:

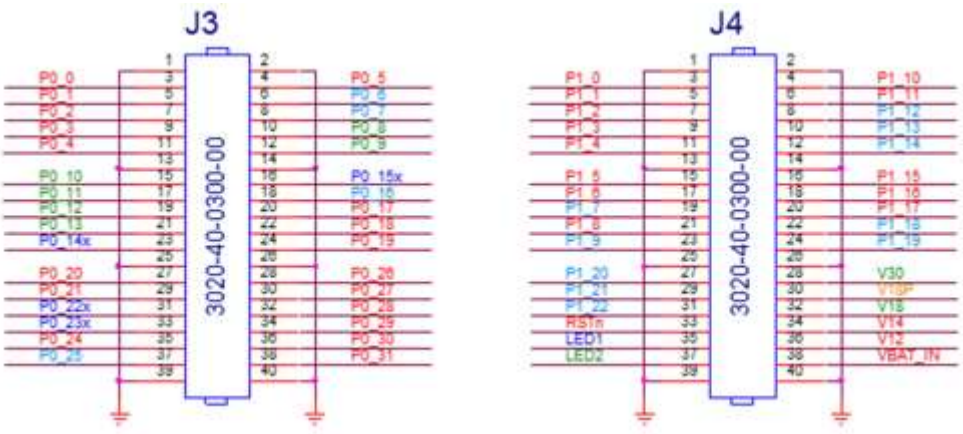

Figure 49: PRO-Mainboard breakout headers

Revision history

Revision	Date	Description
1.1	18-Jan-2022	Updated logo, disclaimer, copyright.
1.0	24-Feb-2019	First released version

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.