User Manual

DA1468x Getting Started with the Development Kit

UM-B-047

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 DA1468x development platform.

Contents

Figures

Tables

Codes

1 Terms and definitions

CLI	Command Line Tool
COM	Communication Port
DBG	Debug
FTDI	Future Technology Devices International
FTP	File Transfer Protocol
GPIO	General Purpose Input/Output
HW	Hardware
IDE	Integrated Development Environment
IRQ	Interrupt Request
LED	Light Emitting Diode
LLD	Low-Level Driver
OPT	One Time Programmable
OS	Operating System
PC	Personal Computer
RAM	Random Access Memory
RTOS	Real Time Operating System
SDK	Software Development Kit
SoC	System on Chip
SW	Software
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

2 References

[1] DA14680, Datasheet, Dialog Semiconductor.
[2] UM-B-057-SmartSnippets_Studio_user_guide, User manual, Dialog Semiconductor.
[3] UM-B-056 DA1468x Software Developer's Guide, User manual, Dialog Semiconductor
[4] UM-B-044-DA1468x Software Platform Reference, User manual, Dialog Semiconductor
[5] UM-B-060-DA1468x_DA1510x Development kit – Pro, User manual, Dialog Semiconductor
[6] UM-B-083 SmartSnippets Toolbox, User manual, Dialog Semiconductor
[7] AN-B-046 DA1468x Booting from serial interfaces, Application Note, Dialog Semiconductor
[8] UM-B-094 DA14683 USB Kit, User manual, Dialog Semiconductor

3 Prerequisites

- SmartSnippetsTM Studio package
- Dialog's Semiconductor SmartSnippets TM DA1468x SDK
- Operating System (Windows or Linux)
- Pro DK DA1468x and accessories
- Serial-port terminal software (e.g. RealTerm)
- A USB connection supporting USB-Serial (FTDI)

4 Introduction

The purpose of this guide is to provide an overview of the DA1468x ProDK Development Board and describe the hardware and software setup needed for using the board. Finally, it guides the user through the process of building and running a basic application called `Blinky`.

The following hardware and software elements are required to use the DA1468x Development Kit:

- The ProDK Development Kit
- SmartSnippetsTM Studio 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 the 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 SmartSnippetsTM DA1468x SDK software, along with all necessary tools.

Section 8 contains all steps for downloading and executing the `Blinky` application.

Note 1 If you are using the DA14683 USB Kit, please consult [8] for usage instructions and limitations that may apply

5 DA1468x – The hardware

The Pro Development Kit (ProDK) consists of a main board (MB-PRO) and a daughterboard featuring DA1468x SoC of the desired package (AQFN60 or WLCSP).

5.1 The ProDK mainboard

Figure 1 illustrates the physical layout of the ProDK. The two (2) daughter boards containing DA1468x devices (AQFN60 as well as WLCSP packages) are shown in Figure 2.

The ProDK main board provides all necessary hardware to enable:

- Full functional verification of the DA1468x family of products with the ability to take precise power measurements by isolating the DA1468x device.
- Full digital connectivity with external hardware using UART, SPI, I2C, quadrature encoder or GPIOs.
- USB based debugging capabilities using the SEGGER J-Link on-board debugger.
- USB based UART communication with the host PC using a Future Technology Devices International (FTDI) chipset which converts UART to USB.

For additional details regarding the DA1468x, please refer to [5] or contact Dialog Bluetooth Device Forum.

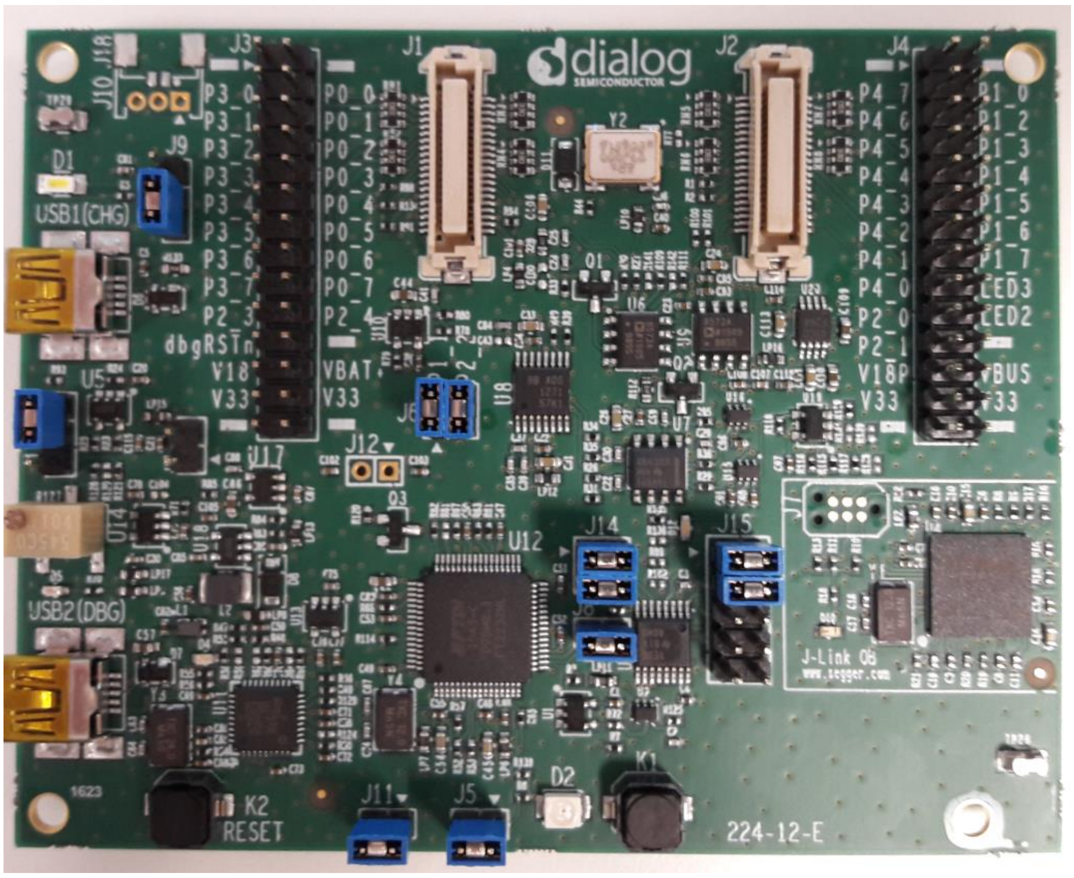


Figure 1: The DA1468x ProDK

5.2 The ProDK daughter-boards

The ProDK mainboard can be combined with one of the two daughter-boards shown in Figure 2. These daughter boards allow evaluation of the two different package options for the DA1468x devices. The daughter-boards each have a switch to select the power supply for the device:

- USB port/charger input (USB1/CHG or USB2/DBG) which is marked VBAT or
- Coin cell battery which is marked VBAT. The coin cell battery can be either rechargeable or non-rechargeable. To use a non-rechargeable coin cell battery the user must make sure that the battery charger code is disabled in the project.

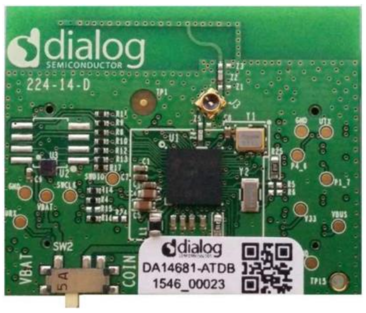
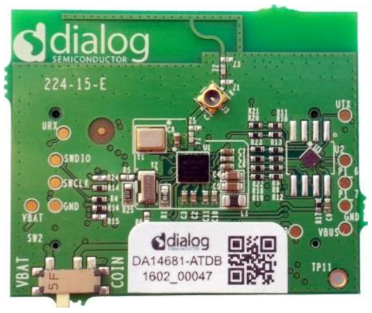
	
DA1468x – AQFN60 package	**DA1468x – WLCSP package**

Figure 2: The daughter-boards a) QFN60 and b) WLCSP to combine with the DA1468x ProDK Development Kit

Note 3 Both daughter-boards are supported in the SmartSnippetsTM DA1468x SDK. More details on how to set the build configuration for each board can be found in [3].

5.3 Connecting the ProDK to the host PC

The ProDK Development Kit allows functional verification of the DA1468x family of devices. It supports connecting external hardware by exporting DA1468x pins to standard headers and enables the user to do precise power measurements through the integrated power measurements circuitry. Additionally, the ProDK mainboard includes an embedded J-Link debugger and an FTDI chipset which allow easy communication with the development host over USB.

The ProDK Development Kit is connected to the host PC over the connector marked as `USB2(DBG)`, as shown in Figure 3 using a standard mini-USB cable. Before connecting the ProDK Development Kit to the host PC, make sure that the mainboard and the desired daughterboard module are properly connected and that the power switch on the daughterboard is in the VBAT position to select USB from motherboard.

Additional information on the exact capabilities of each daughterboard can be found in [5].

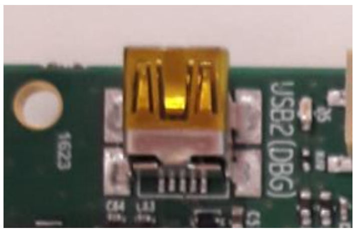


Figure 3: USB2 (DBG) connector

6 DA1468x – Software Installation

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

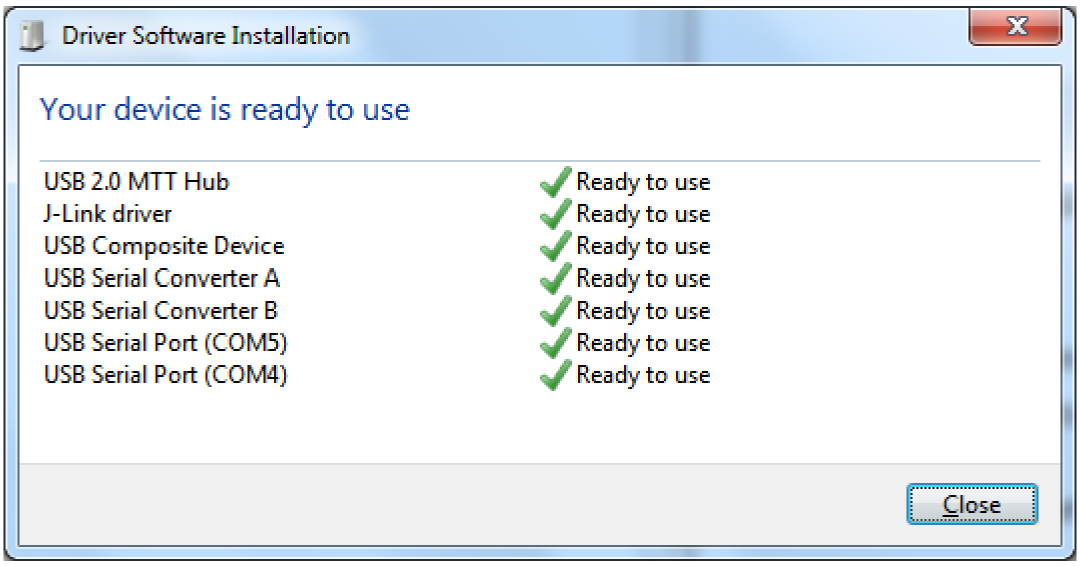
For proper evaluation and application development using the DA1468x SoC and the ProDK an external host is required. This external host must have an operating system already installed (Windows or Linux) and USB ports as described in section 3.

Internet connectivity is also highly recommended for proper driver and software installation.

6.3 Driver installation

6.3.1 Microsoft Windows

On the first connection to a host PC which is running Microsoft Windows and has Internet connectivity, the system will detect several devices and will automatically install all necessary drivers. If the system is configured to use Windows Update, this process may need several minutes to be finished.

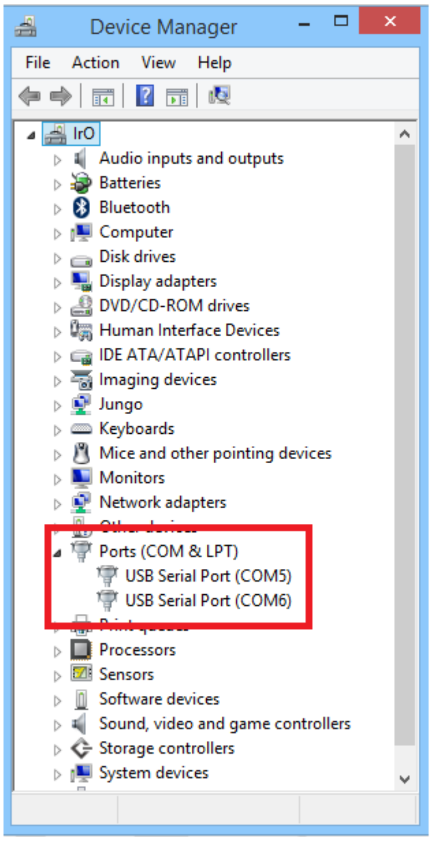

Driver Software Installation

Your device is ready to use

USB 2.0 MTT Hub	✓ Ready to use
J-Link driver	✓ Ready to use
USB Composite Device	✓ Ready to use
USB Serial Converter A	✓ Ready to use
USB Serial Converter B	✓ Ready to use
USB Serial Port (COM5)	✓ Ready to use
USB Serial Port (COM4)	✓ Ready to use

Figure 4: Windows driver installation

After driver installation is successfully completed, the system will pop-up a window similar to the one presented in Figure 4. Please note that the COM port number assigned to the newly attached ProDK mainboard might be different than the one shown in Figure 4, where the assigned COM port numbers are COM4 and COM5. The COM port number can be found in Device Manager (**Control Panel > Device Manager > Ports (COM & LPT)**) as shown in Figure 5.



Device Manager

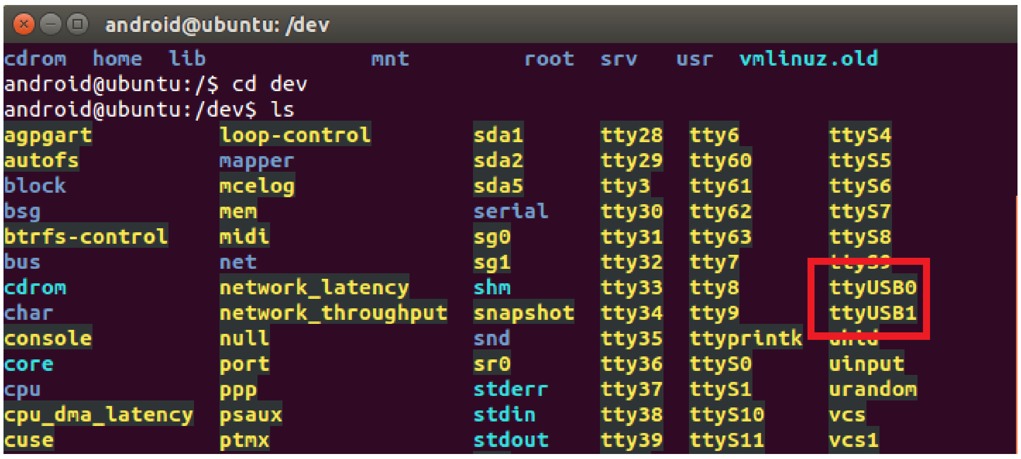
File Action View Help

- IrO
  - Audio inputs and outputs
  - Batteries
  - Bluetooth
  - Computer
  - Disk drives
  - Display adapters
  - DVD/CD-ROM drives
  - Human Interface Devices
  - IDE ATA/ATAPI controllers
  - Imaging devices
  - Jungo
  - Keyboards
  - Mice and other pointing devices
  - Monitors
  - Network adapters
  - Other devices
  - Ports (COM & LPT)
    - USB Serial Port (COM5)
    - USB Serial Port (COM6)
  - Print queues
  - Processors
  - Sensors
  - Software devices
  - Sound, video and game controllers
  - Storage controllers
  - System devices

Figure 5: 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 6. 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.



```
android@ubuntu: /dev
cdrom  home  lib             mnt         root  srv   usr  vmlinuz.old
android@ubuntu:/$ cd dev
android@ubuntu:/dev$ ls
agpgart          loop-control        sda1      tty28  tty6       ttyS4
autofs           mapper              sda2      tty29  tty60      ttyS5
block            mcelog              sda5      tty3   tty61      ttyS6
bsg              mem                 serial    tty30  tty62      ttyS7
btrfs-control    midi                sg0       tty31  tty63      ttyS8
bus              net                 sg1       tty32  tty7       ttyS9
cdrom            network_latency     shm       tty33  tty8       ttyUSB0
char             network_throughput  snapshot  tty34  tty9       ttyUSB1
console          null                snd       tty35  ttyprintk  uhid
core             port                sr0       tty36  ttyS0      uinput
cpu              ppp                 stderr    tty37  ttyS1      urandom
cpu_dma_latency  psaux               stdin     tty38  ttyS10     vcs
cuse             ptmx                stdout    tty39  ttyS11     vcs1
```

Figure 6: 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 DA1468x device. In the previous sections this was either COM4 or /dev/ttyUSB0. The second (higher number) is used to export measurement data from the current sense circuitry on the ProDK [5] to the Power Profiler tool [6].

6.4 Configuring the serial port for UART1

Several development tools require UART1 to be routed to the FTDI serial port. Please refer to [5] 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 RealTerm can be used to fully validate the connection to the ProDK

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:

1. Connect the ProDK board to the PC board via USB cable to `USB2(DBG)` as described in Paragraph 5.3 above.
2. Verify that the host discovered two serial ports – the first of which connected to UART1 (see section 6.3.3).
3. Start a terminal software with HEX display capabilities (Figure 8, Reference points 2 and 3) (i.e. RealTerm)
4. Open the serial port (Figure 8, Reference points 4 and 5) which corresponds to ProDK UART1 (i.e. COM4 for Windows), using the parameters shown in Table 1:

Table 1: Parameters for connecting to UART1

Settings	**Values**
Baud rate	**57600**
Data bits	**8**
Parity	**None**
Stop bits	**1**
Handshaking	**None**

5. Click `K2` Reset button on the ProDK board as depicted in Figure 7. If terminal shows bytes (non-printable characters) similar to those presented in Reference point 1 in Figure 8 on every reset button push, the connection has been correctly setup and the programmer can be used (sometimes other characters bytes can be sent). The sequence shows in Figure 8 is a confirmation sequence corresponding to a correct setup confirmation. The sequence is generated by the Bootloader [7] and corresponds to an ASCII string with the chip version.

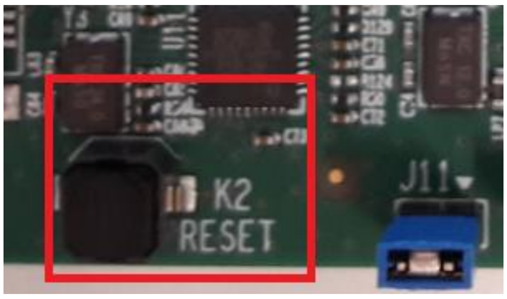
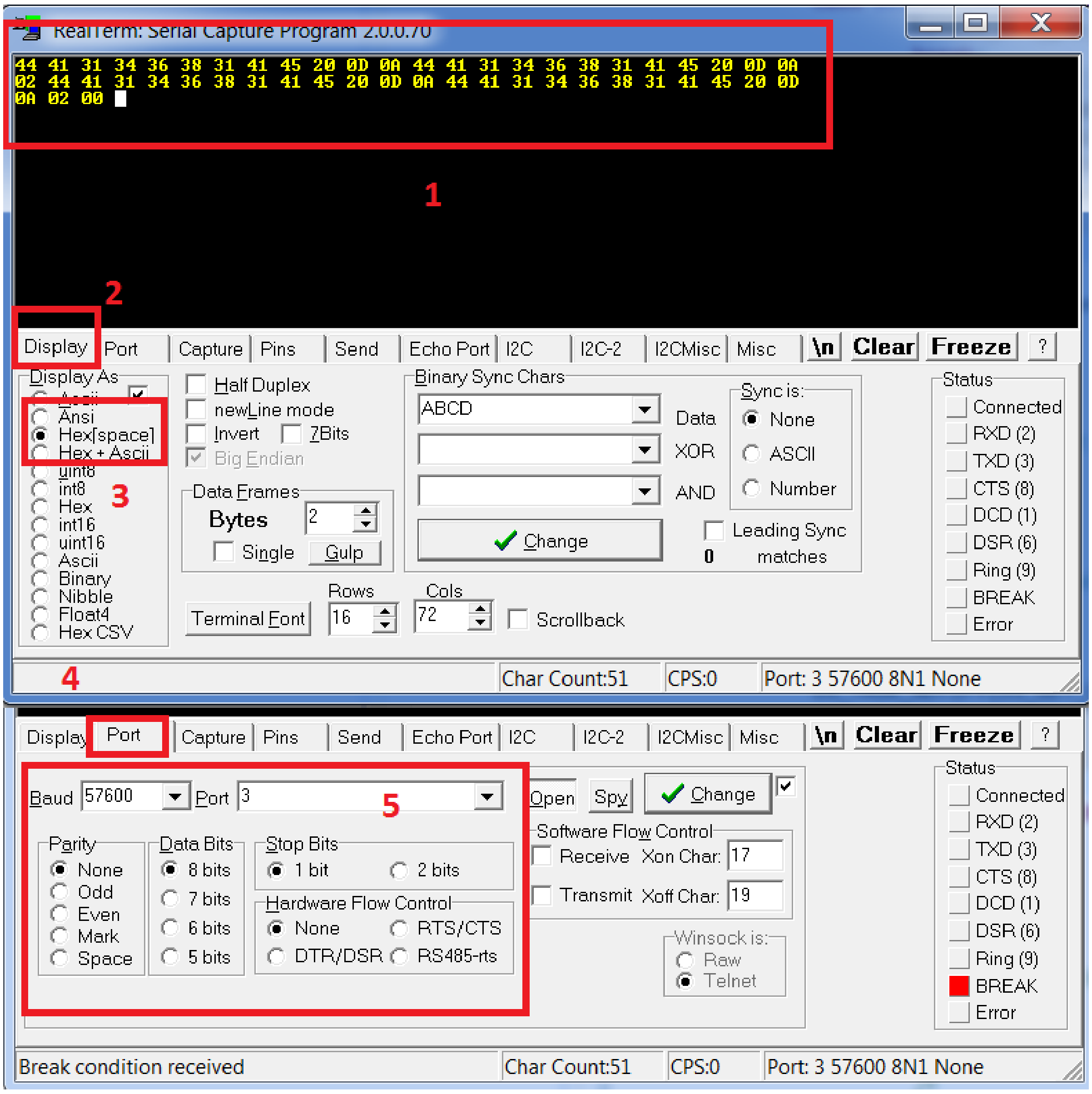


Figure 7: K2 RESET button in ProDK



```
44 41 31 34 36 38 31 41 45 20 0D 0A 44 41 31 34 36 38 31 41 45 20 0D 0A
02 44 41 31 34 36 38 31 41 45 20 0D 0A 44 41 31 34 36 38 31 41 45 20 0D
0A 02 00
```

Display As: Ascii, Ansi, [✓ Hex[space]], Hex + Ascii, uint8, int8, Hex, int16, uint16, Ascii, Binary, Nibble, Float4, Hex CSV

Half Duplex, newLine mode, Invert, 7Bits, Big Endian

Data Frames: Bytes 2, Single, Gulp

Binary Sync Chars: ABCD; Data, XOR, AND; Sync is: [✓ None], ASCII, Number; Leading Sync; 0 matches; Change

Terminal Font; Rows 16; Cols 72; Scrollback

Char Count:51 | CPS:0 | Port: 3 57600 8N1 None

Baud 57600; Port 3; Open; Spy; Change

Parity: [✓ None], Odd, Even, Mark, Space

Data Bits: [✓ 8 bits], 7 bits, 6 bits, 5 bits

Stop Bits: [✓ 1 bit], 2 bits

Hardware Flow Control: [✓ None], RTS/CTS, DTR/DSR, RS485-rts

Software Flow Control: Receive Xon Char: 17; Transmit Xoff Char: 19

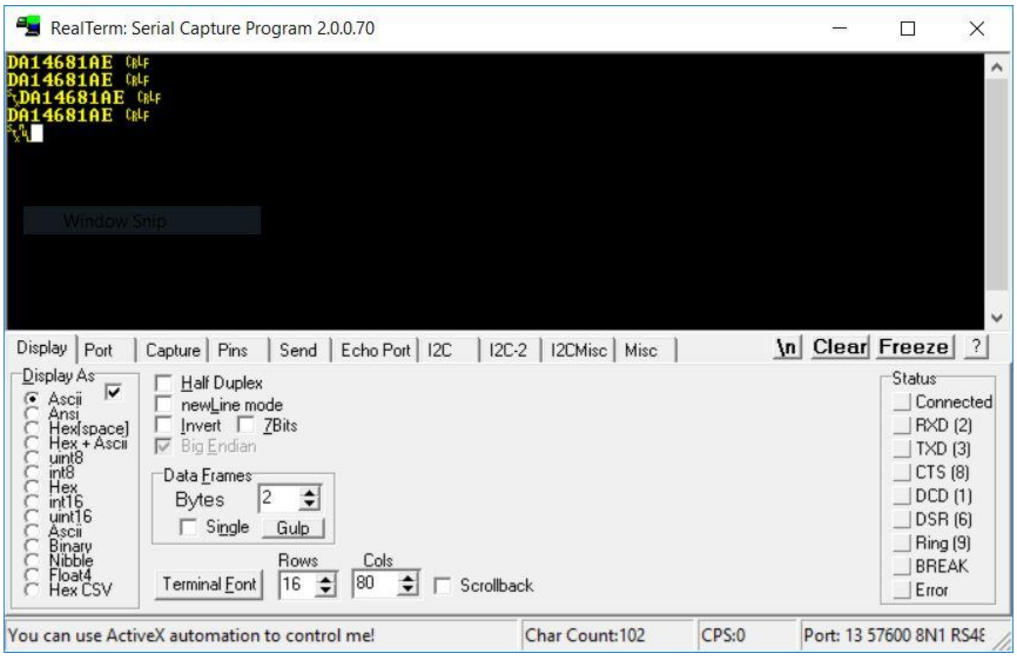
Winsock is: Raw, [✓ Telnet]

Status: BREAK 🟥

Break condition received | Char Count:51 | CPS:0 | Port: 3 57600 8N1 None

Figure 8: Setting port and testing connectivity via RealTerm

Note 4 To verify the chip version user can select ASCII display option. After pushing and releasing the `K2` reset button the chip version is printed four times, as shown in Figure 9.



```
DA14681AE CRLF
DA14681AE CRLF
DA14681AE CRLF
DA14681AE CRLF
```

Display As: (● Ascii) Ansi, Hex[space], Hex + Ascii, uint8, int8, Hex, int16, uint16, Ascii, Binary, Nibble, Float4, Hex CSV ✓

☐ Half Duplex
☐ newLine mode
☐ Invert ☐ 7Bits
☑ Big Endian

Data Frames: Bytes 2, ☐ Single

Rows 16, Cols 80, ☐ Scrollback

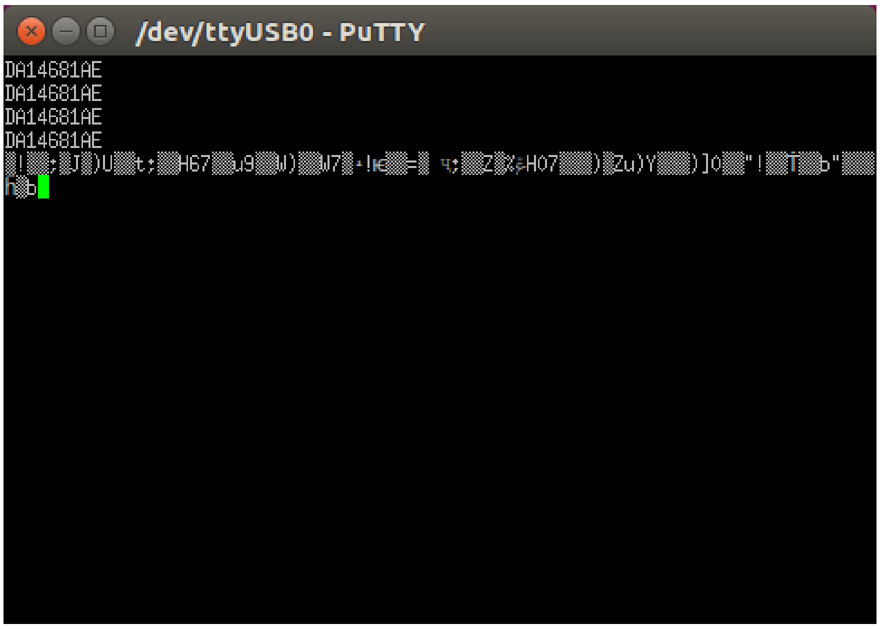
Status: Connected, RXD (2), TXD (3), CTS (8), DCD (1), DSR (6), Ring (9), BREAK, Error

You can use ActiveX automation to control me! | Char Count:102 | CPS:0 | Port: 13 57600 8N1 RS48

Figure 9: Chip version in RealTerm in 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 57600 baud and press `K2` Reset. There should be an output as shown in Figure 10 which contains ASCII characters with the device name as well as non-ASCII characters. This is enough to validate the connection to UART1.



```
DA14681AE
DA14681AE
DA14681AE
DA14681AE
▒!▒;▒J▒)U▒t;▒H67▒u9▒W)▒W7▒·!ю▒=▒ ч;▒Z▒%;H07▒)▒Zu)Y▒)]◊▒"!▒T▒b"▒
h▒b
```

Figure 10: Chip version in PuTTY in 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
- Check for possible cabling issue by using a different USB cable
- Connect the two elements using a different USB port on the host PC

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

7 Software Development Tools

7.1 SmartSnippets™ Introduction

Dialog SmartSnippetsTM Studio is a royalty-free software development platform for Smartbond™ devices. It fully supports the DA1468x family of devices.

SmartSnippetsTM Studio contains:

- SmartSnippetsTM IDE: Eclipse CDT based IDE with pre-configured plugins to provide the build/debug environment
- SmartSnippetsTM DA1468x SDK
- SmartSnippetsTM Toolbox: A tool suite covering all software development requirements, including:
  - Programming and loading of firmware into SRAM, OTP and Flash
  - Power profiling
- SmartSnippetsTM Documentation

The SmartSnippetsTM 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/ .

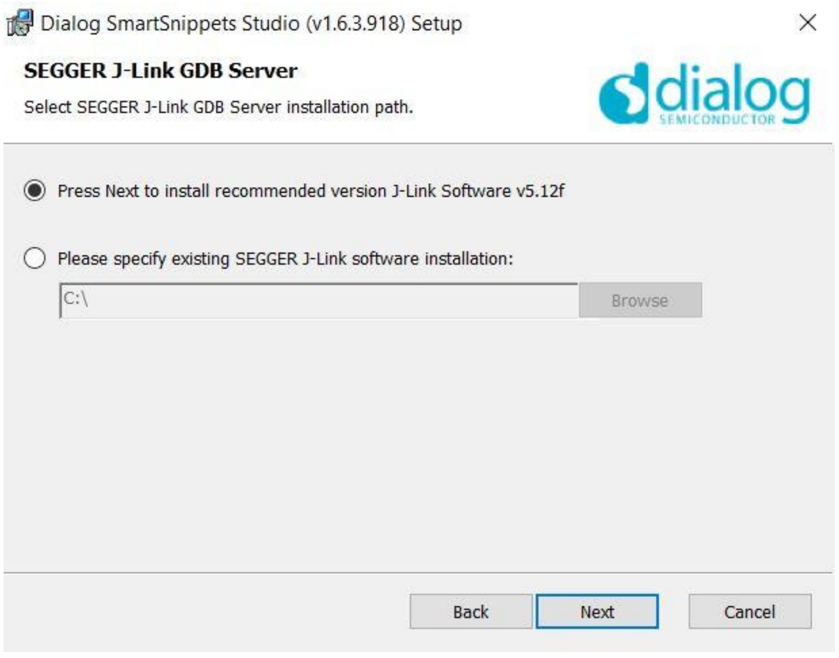
7.2 Installation

The installation procedure for SmartSnippetsTM Studio is described in detail in [2]. A summary of the steps is given here. Download the Windows or Linux version of SmartSnippetsTM Studio from the Software and Tools section of the DA14681 Product Dialog Support Website: https://support.dialog-semiconductor.com/connectivity/product/da14681

7.2.1 Windows

SmartSnippetsTM Studio installer (.msi). 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.


Dialog SmartSnippets Studio (v1.6.3.918) Setup

SEGGER J-Link GDB Server

Select SEGGER J-Link GDB Server installation path.

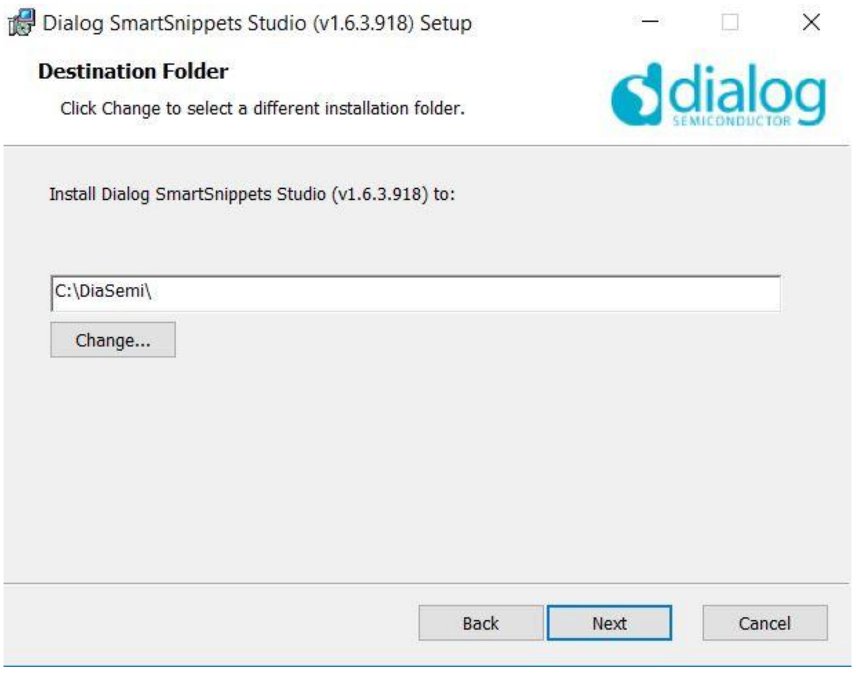
(●) Press Next to install recommended version J-Link Software v5.12f

( ) Please specify existing SEGGER J-Link software installation:

C:\

Figure 11: Automatically install J-Link

- Select the destination folder for the SmartSnippetsTM Studio.



Dialog SmartSnippets Studio (v1.6.3.918) Setup

Destination Folder

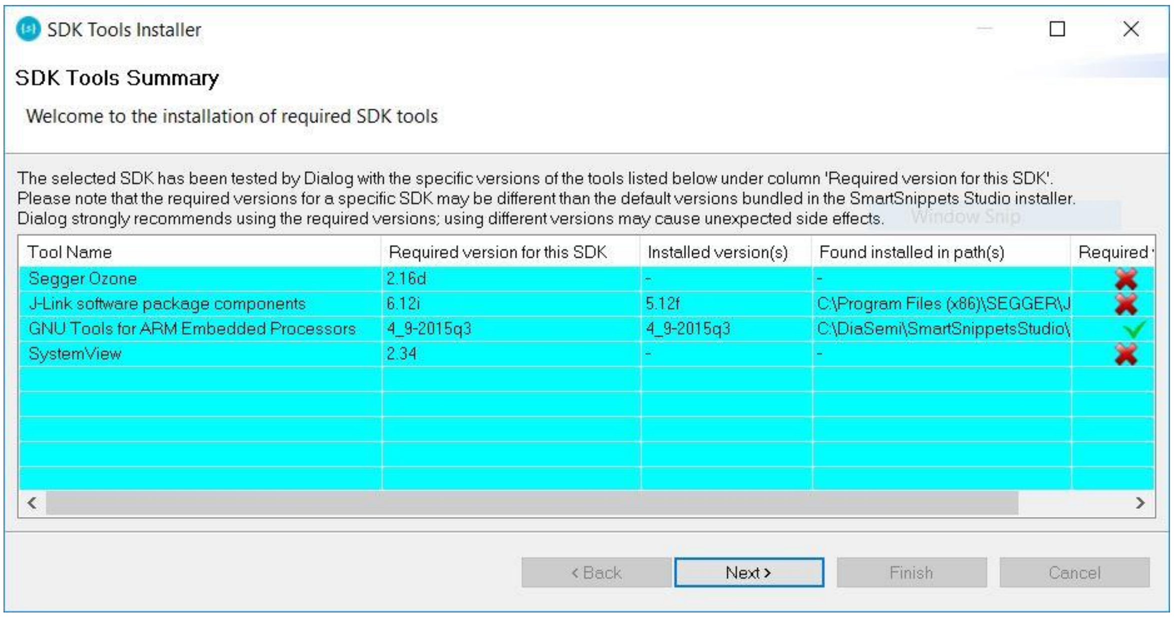
Click Change to select a different installation folder.

Install Dialog SmartSnippets Studio (v1.6.3.918) to:

C:\DiaSemi\

Figure 12: Select Smart Snippets Studio install directory

- The installer will then check if the other required SEGGER tools are installed. On a clean PC they will probably not be there and so the next steps are to download, install and point SmartSnippets to them



SDK Tools Installer

SDK Tools Summary

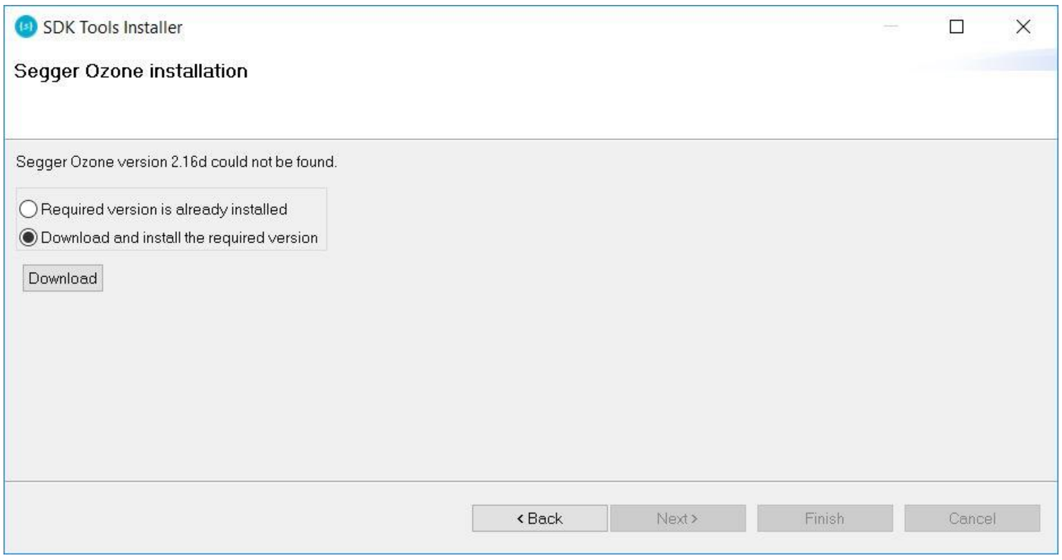
Welcome to the installation of required SDK tools

The selected SDK has been tested by Dialog with the specific versions of the tools listed below under column 'Required version for this SDK'.
Please note that the required versions for a specific SDK may be different than the default versions bundled in the SmartSnippets Studio installer.
Dialog strongly recommends using the required versions; using different versions may cause unexpected side effects.

Tool Name	Required version for this SDK	Installed version(s)	Found installed in path(s)	Required
Segger Ozone	2.16d	-	-	✗
J-Link software package components	6.12i	5.12f	C:\Program Files (x86)\SEGGER\J	✗
GNU Tools for ARM Embedded Processors	4_9-2015q3	4_9-2015q3	C:\DiaSemi\SmartSnippetsStudio\	✓
SystemView	2.34	-	-	✗

Figure 13: Tools that require manual installation

- Tick the **download and install** radio button and then press **Download** button for SEGGER Ozone tool



SDK Tools Installer

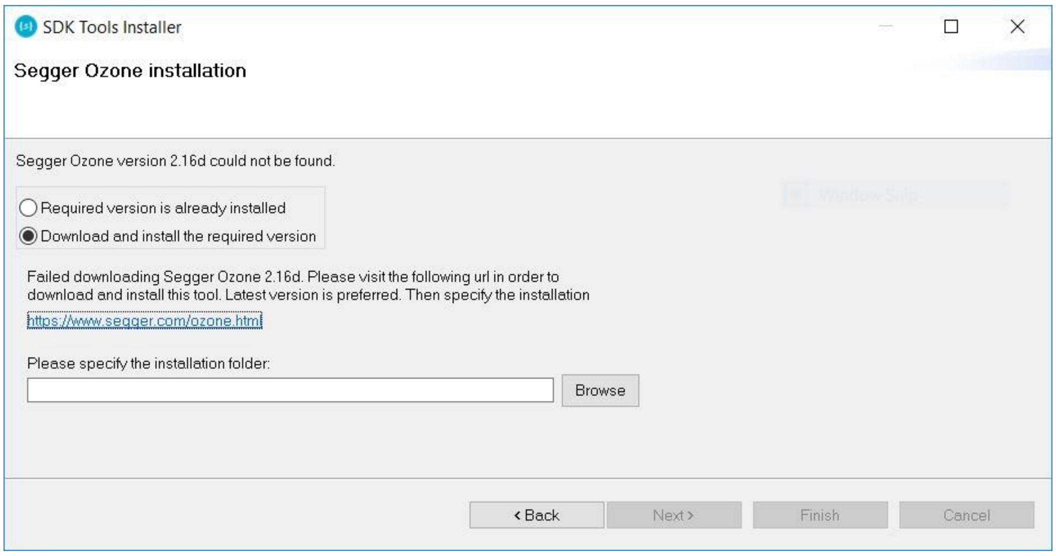
Segger Ozone installation

Segger Ozone version 2.16d could not be found.

( ) Required version is already installed
(●) Download and install the required version

Figure 14: Start Ozone download

- The download will fail and so click on the link to manually download Ozone



SDK Tools Installer

Segger Ozone installation

Segger Ozone version 2.16d could not be found.

( ) Required version is already installed
(●) Download and install the required version

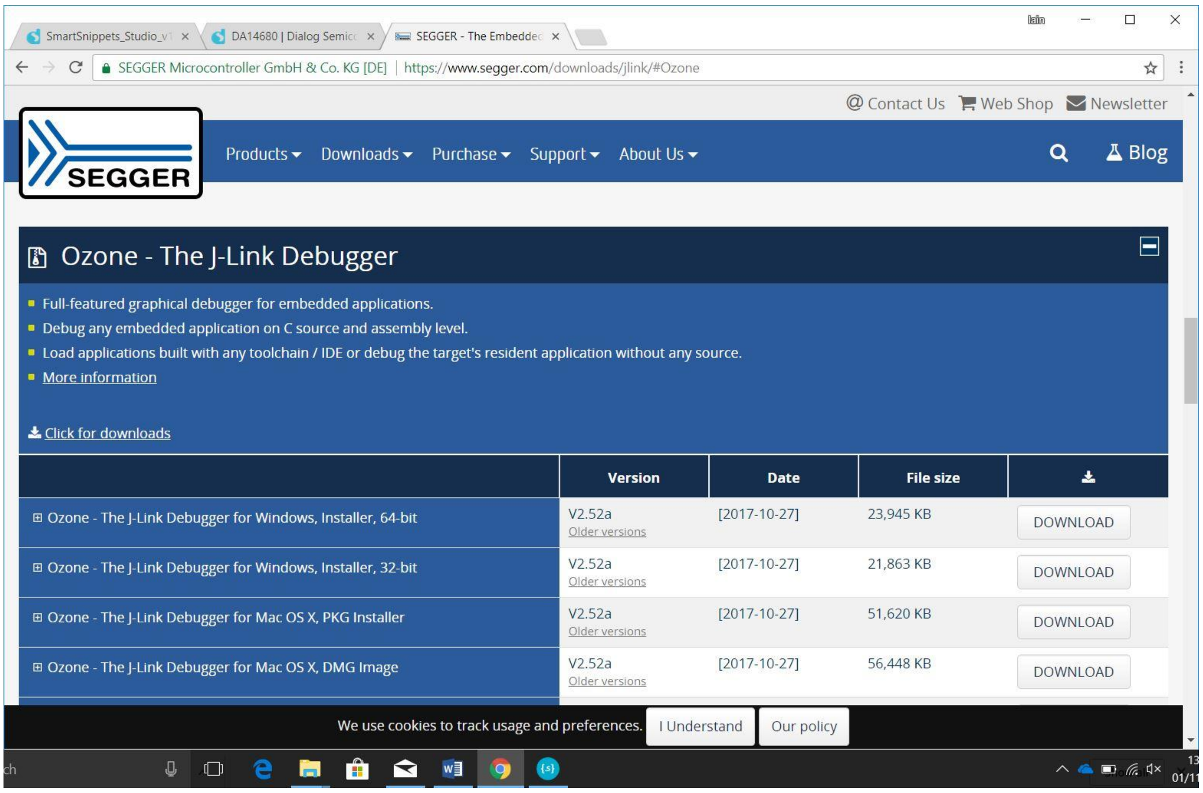
Failed downloading Segger Ozone 2.16d. Please visit the following url in order to download and install this tool. Latest version is preferred. Then specify the installation

https://www.segger.com/ozone.html

Please specify the installation folder:

Figure 15: Automatic Ozone download failed

- Using a browser download the latest version from https://www.segger.com/downloads/jlink/#Ozone which at the point of writing was **2.52a** and run the installer.



Ozone - The J-Link Debugger

- Full-featured graphical debugger for embedded applications.
- Debug any embedded application on C source and assembly level.
- Load applications built with any toolchain / IDE or debug the target's resident application without any source.
- More information

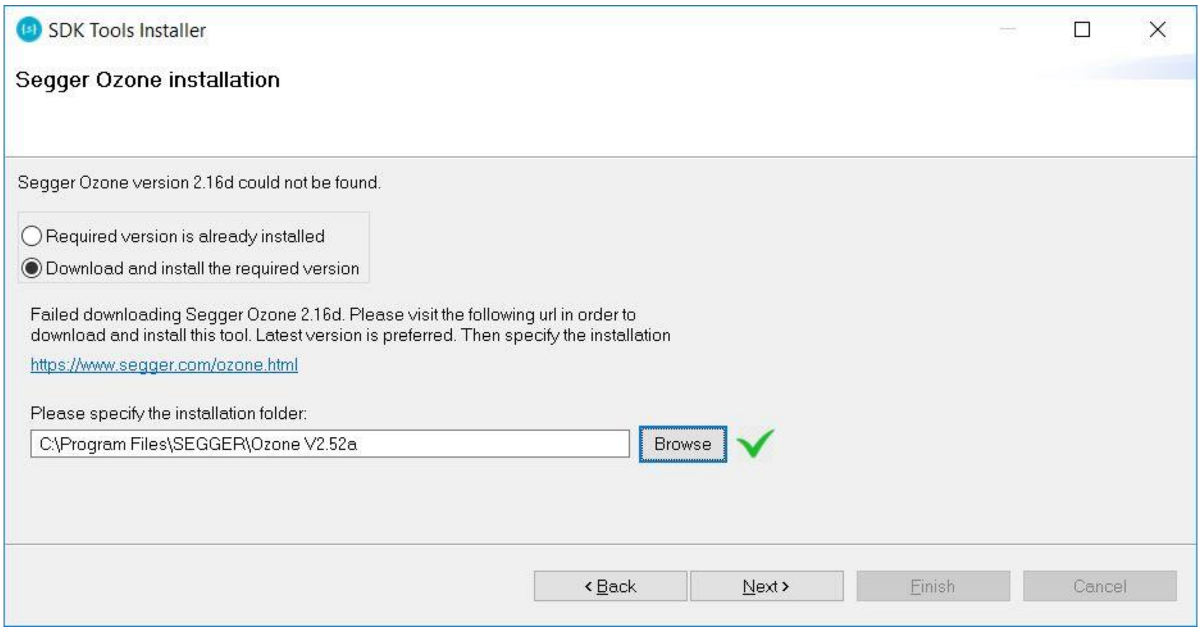
Click for downloads

	Version	Date	File size	
Ozone - The J-Link Debugger for Windows, Installer, 64-bit	V2.52a Older versions	[2017-10-27]	23,945 KB	DOWNLOAD
Ozone - The J-Link Debugger for Windows, Installer, 32-bit	V2.52a Older versions	[2017-10-27]	21,863 KB	DOWNLOAD
Ozone - The J-Link Debugger for Mac OS X, PKG Installer	V2.52a Older versions	[2017-10-27]	51,620 KB	DOWNLOAD
Ozone - The J-Link Debugger for Mac OS X, DMG Image	V2.52a Older versions	[2017-10-27]	56,448 KB	DOWNLOAD

We use cookies to track usage and preferences. I Understand Our policy

Figure 16: SEGGER Ozone download page

- This will install the tool to C:\Program Files\SEGGER\Ozone v2.52a. Use the **Browse** button to find this folder and then press **Next**.



SDK Tools Installer

Segger Ozone installation

Segger Ozone version 2.16d could not be found.

( ) Required version is already installed
(●) Download and install the required version

Failed downloading Segger Ozone 2.16d. Please visit the following url in order to download and install this tool. Latest version is preferred. Then specify the installation

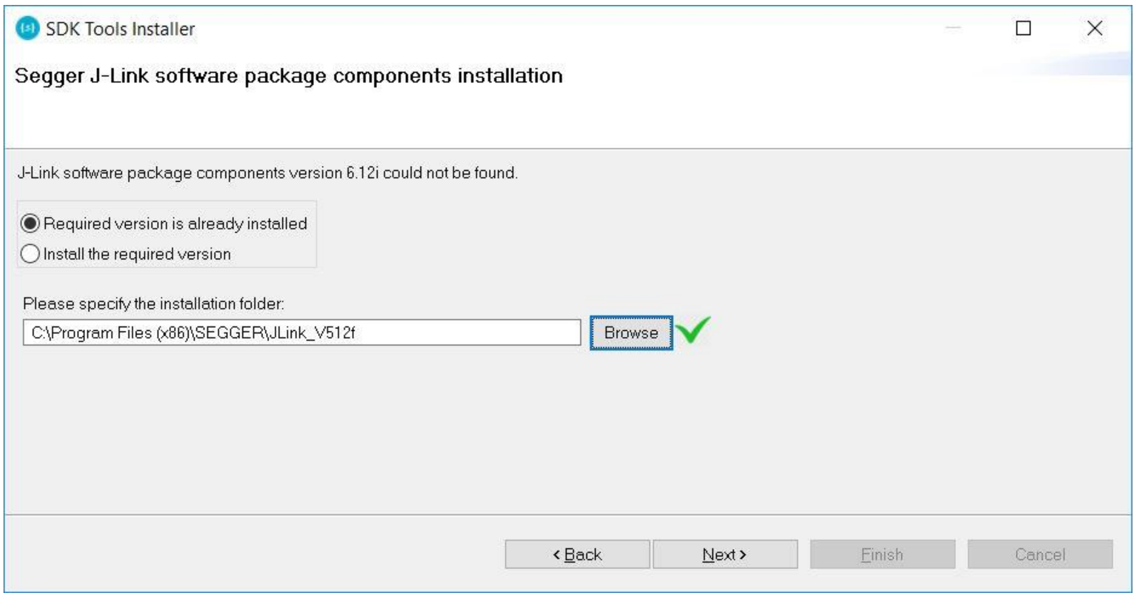
https://www.segger.com/ozone.html

Please specify the installation folder:

C:\Program Files\SEGGER\Ozone V2.52a

Figure 17: Set Ozone installation directory

- The next stage is to set the J-Link installation directory. This software is installed as part of Ozone install and so it is only necessary to set the directory which should be `C:\Program Files (x86)\SEGGER\JLink_V512f`.


SDK Tools Installer

Segger J-Link software package components installation

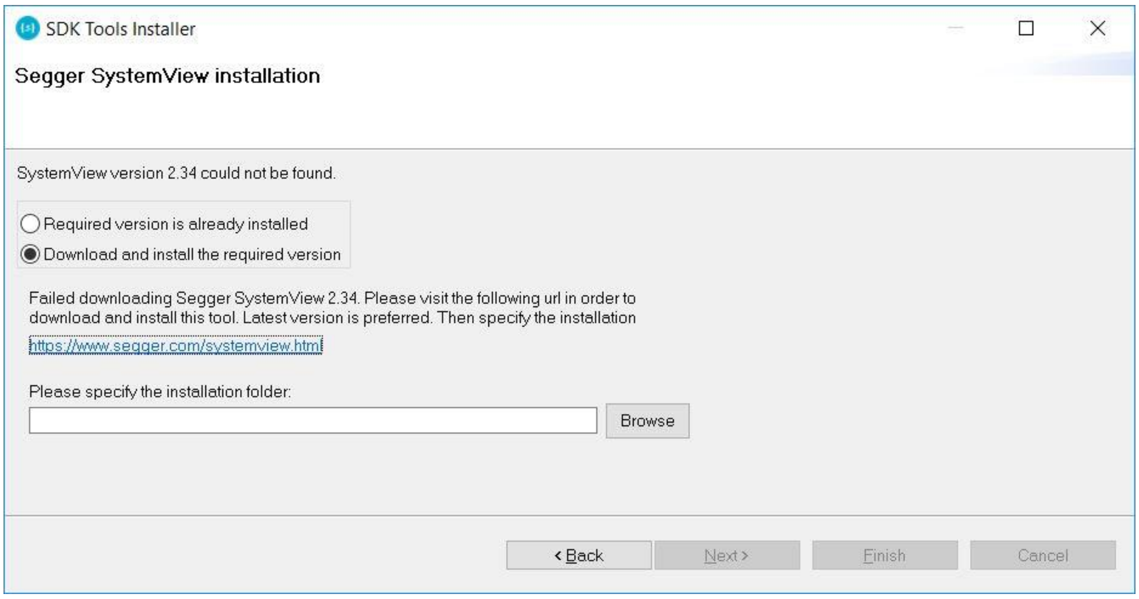
J-Link software package components version 6.12i could not be found.

(●) Required version is already installed
( ) Install the required version

Please specify the installation folder:
C:\Program Files (x86)\SEGGER\JLink_V512f

Figure 18: Set J-Link installation directory

- The final stage is to install SEGGER SystemView. The automatic download will fail so it must be done manually


SDK Tools Installer

Segger SystemView installation

SystemView version 2.34 could not be found.

( ) Required version is already installed
(●) Download and install the required version

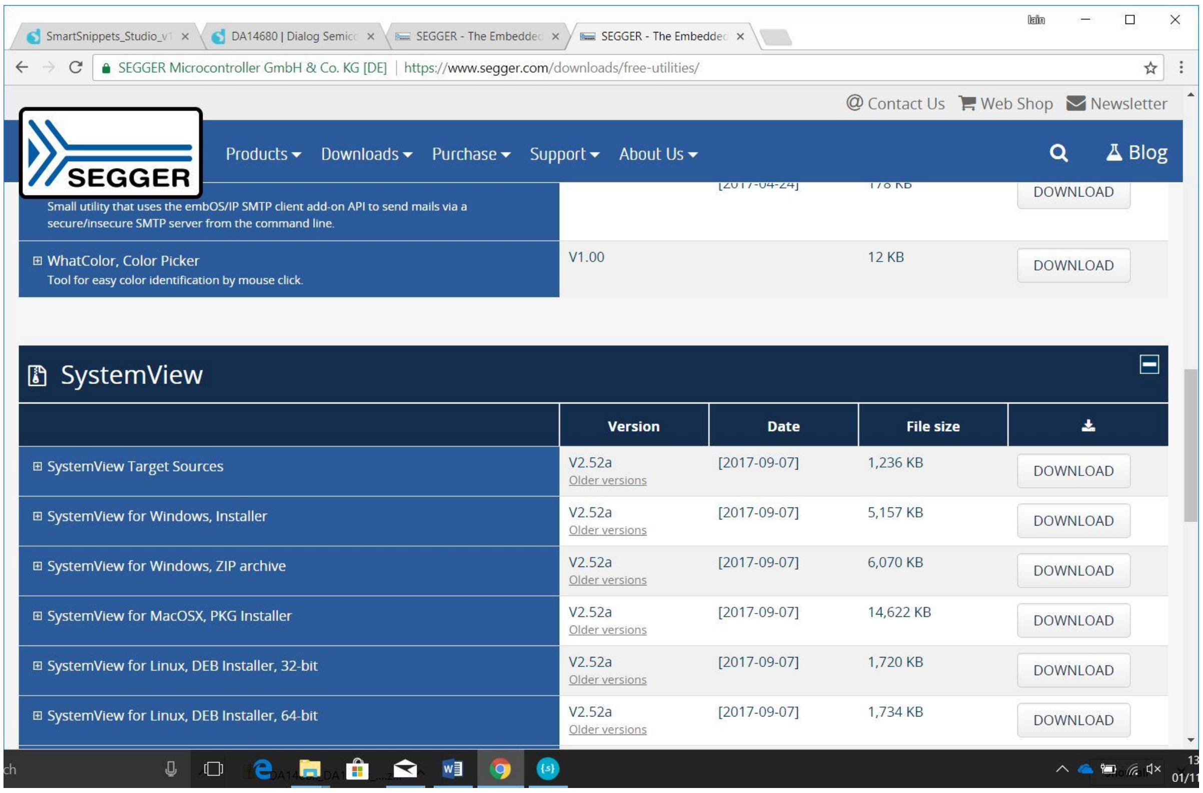
Failed downloading Segger SystemView 2.34. Please visit the following url in order to download and install this tool. Latest version is preferred. Then specify the installation

https://www.segger.com/systemview.html

Please specify the installation folder:

Figure 19: SystemView Download fails

- Using a browser download the latest SystemView installer from https://www.segger.com/downloads/jlink/#SystemView . The latest version at the time of writing was v2.52a.

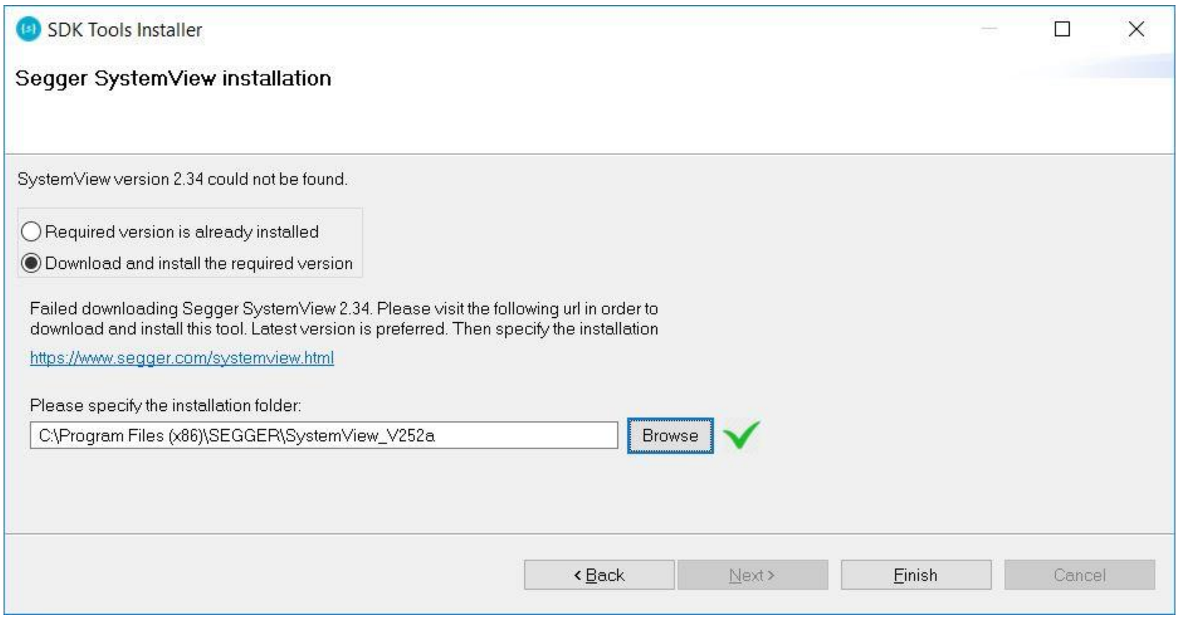

https://www.segger.com/downloads/free-utilities/

SystemView

	Version	Date	File size	
SystemView Target Sources	V2.52a Older versions	[2017-09-07]	1,236 KB	DOWNLOAD
SystemView for Windows, Installer	V2.52a Older versions	[2017-09-07]	5,157 KB	DOWNLOAD
SystemView for Windows, ZIP archive	V2.52a Older versions	[2017-09-07]	6,070 KB	DOWNLOAD
SystemView for MacOSX, PKG Installer	V2.52a Older versions	[2017-09-07]	14,622 KB	DOWNLOAD
SystemView for Linux, DEB Installer, 32-bit	V2.52a Older versions	[2017-09-07]	1,720 KB	DOWNLOAD
SystemView for Linux, DEB Installer, 64-bit	V2.52a Older versions	[2017-09-07]	1,734 KB	DOWNLOAD

Figure 20: Download SystemView installer

- Run the installer which will install SystemView to `C:\Program Files (x86)\SEGGER\SystemView_V252a`. Use **Browse** button to set this directory and then press **Finish**.


SDK Tools Installer

Segger SystemView installation

SystemView version 2.34 could not be found.

( ) Required version is already installed
(●) Download and install the required version

Failed downloading Segger SystemView 2.34. Please visit the following url in order to download and install this tool. Latest version is preferred. Then specify the installation https://www.segger.com/systemview.html

Please specify the installation folder:
C:\Program Files (x86)\SEGGER\SystemView_V252a

Figure 21: Set SystemView installation directory

- SmartSnippetsTM Studio is now installed.

7.2.2 Linux

Before running the installer, it is necessary to install some Linux packages that are mandatory for the execution of SmartSnippetsTM Studio. Without them SmartSnippetsTM Studio will not run correctly but will fail with no reported error. More details are available in [2].

- For **CentOS 7:**

  ```
  sudo yum install install epel-release
  sudo yum install webkitgtk.x86_64
  sudo yum install glibc.i686 ncurses-libs.i686
  sudo yum install qt-x11 (required for SystemView tool)
  ```

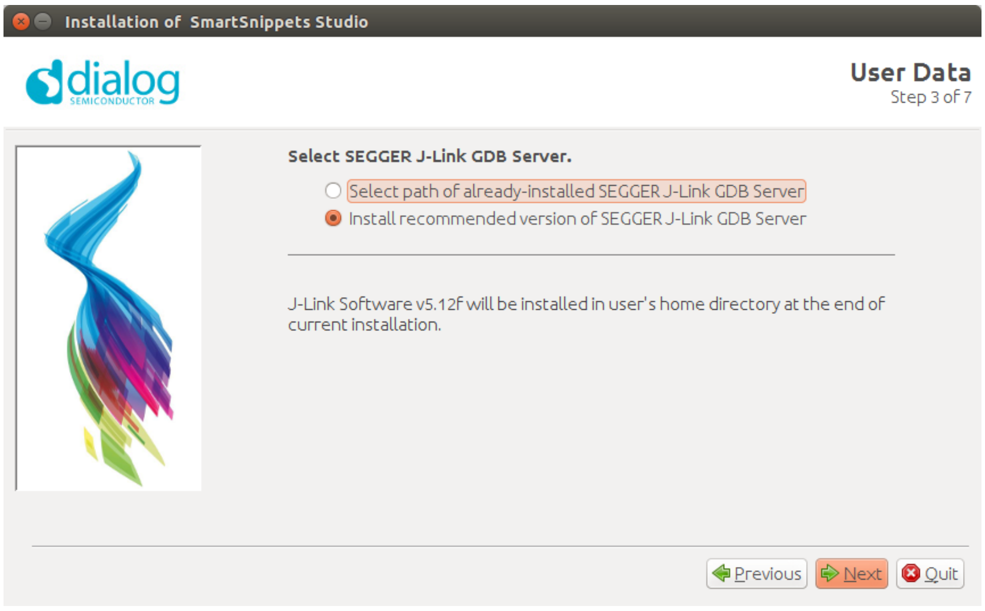
- For **Fedora 25**:

  ```
  sudo yum install webkitgtk.x86_64
  sudo yum install glibc.i686 ncurses-libs.i686
  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 SmartSnippetsTM 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.



Select SEGGER J-Link GDB Server.

Select path of already-installed SEGGER J-Link GDB Server [✓ Install recommended version of SEGGER J-Link GDB Server]

J-Link Software v5.12f will be installed in user's home directory at the end of current installation.

Figure 22: Automatically install J-Link

- Select the destination folder for the SmartSnippetsTM Studio.

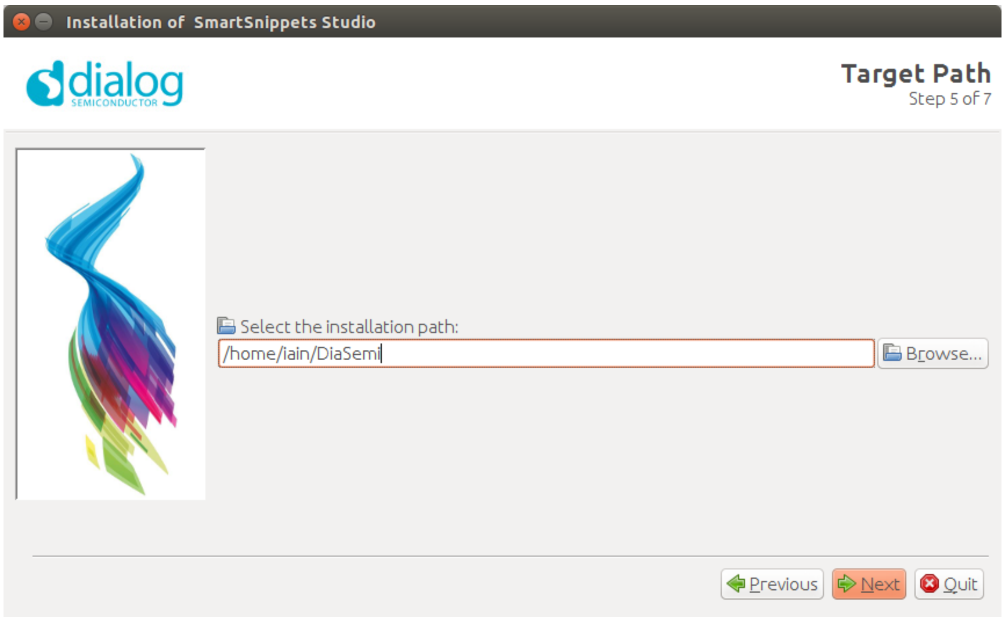
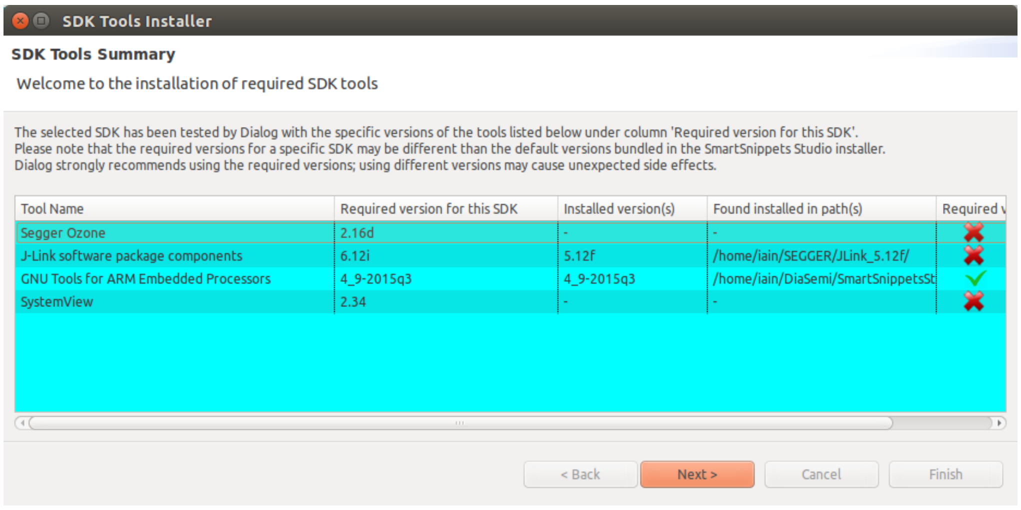


Figure 23: Select Smart Snippets Studio install directory

- The installer will then check if other SEGGER tools it needs are installed. On a clean PC they will probably not be there and so the next steps are to download, install and point SmartSnippets to them

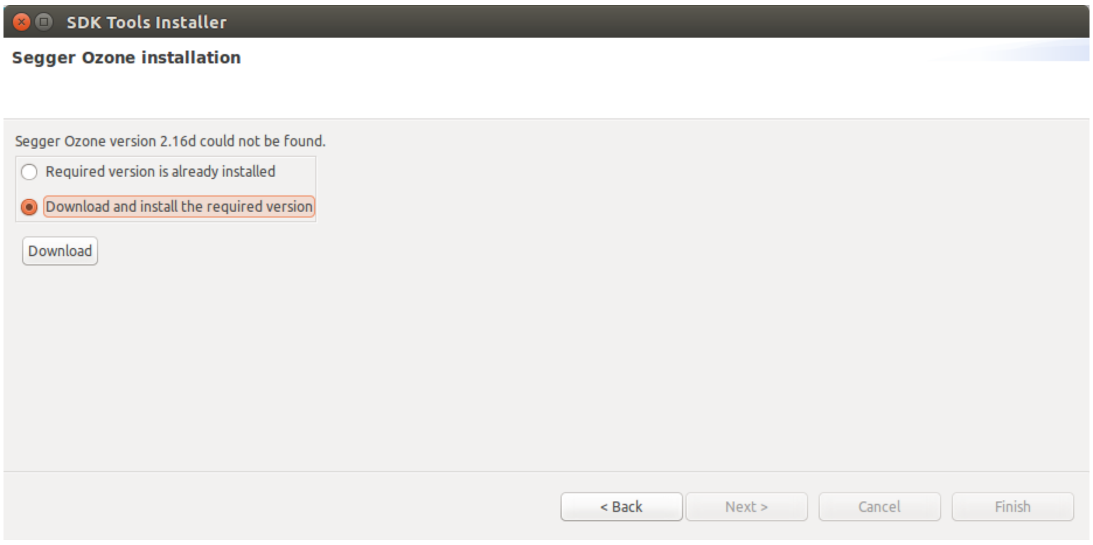


The selected SDK has been tested by Dialog with the specific versions of the tools listed below under column 'Required version for this SDK'.
Please note that the required versions for a specific SDK may be different than the default versions bundled in the SmartSnippets Studio installer.
Dialog strongly recommends using the required versions; using different versions may cause unexpected side effects.

Tool Name	Required version for this SDK	Installed version(s)	Found installed in path(s)	Required v
Segger Ozone	2.16d	-	-	✗
J-Link software package components	6.12i	5.12f	/home/iain/SEGGER/JLink_5.12f/	✗
GNU Tools for ARM Embedded Processors	4_9-2015q3	4_9-2015q3	/home/iain/DiaSemi/SmartSnippetsSt	✓
SystemView	2.34	-	-	✗

Figure 24: Tools that require manual installation

- Tick the **download and install** radio button and then press **Download** button for SEGGER Ozone tool

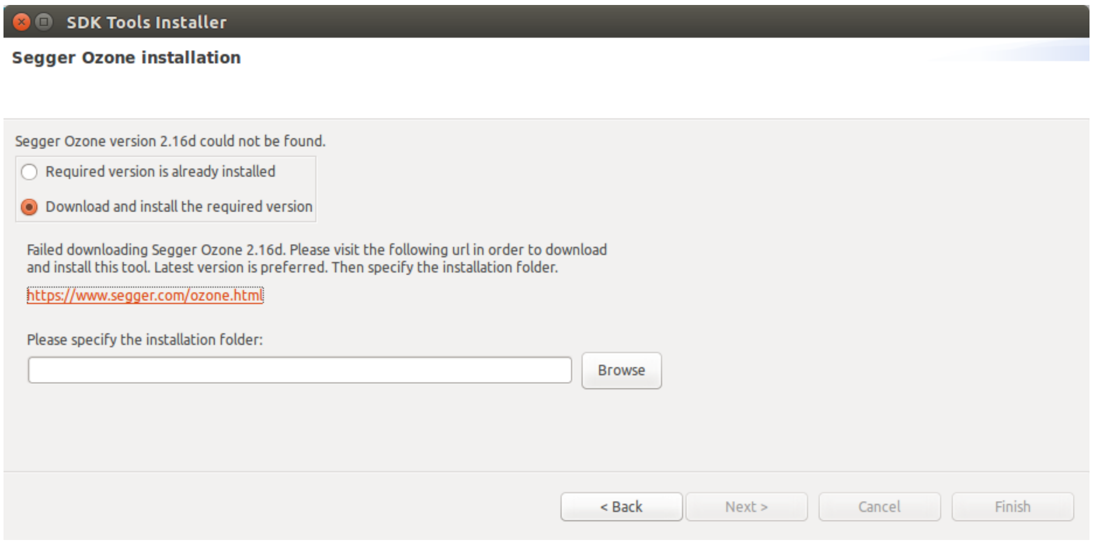


Segger Ozone version 2.16d could not be found.

( ) Required version is already installed
(●) Download and install the required version

Figure 25: Start Ozone download

- The download will fail and so click on the link to manually download Ozone



Segger Ozone version 2.16d could not be found.

( ) Required version is already installed
(●) Download and install the required version

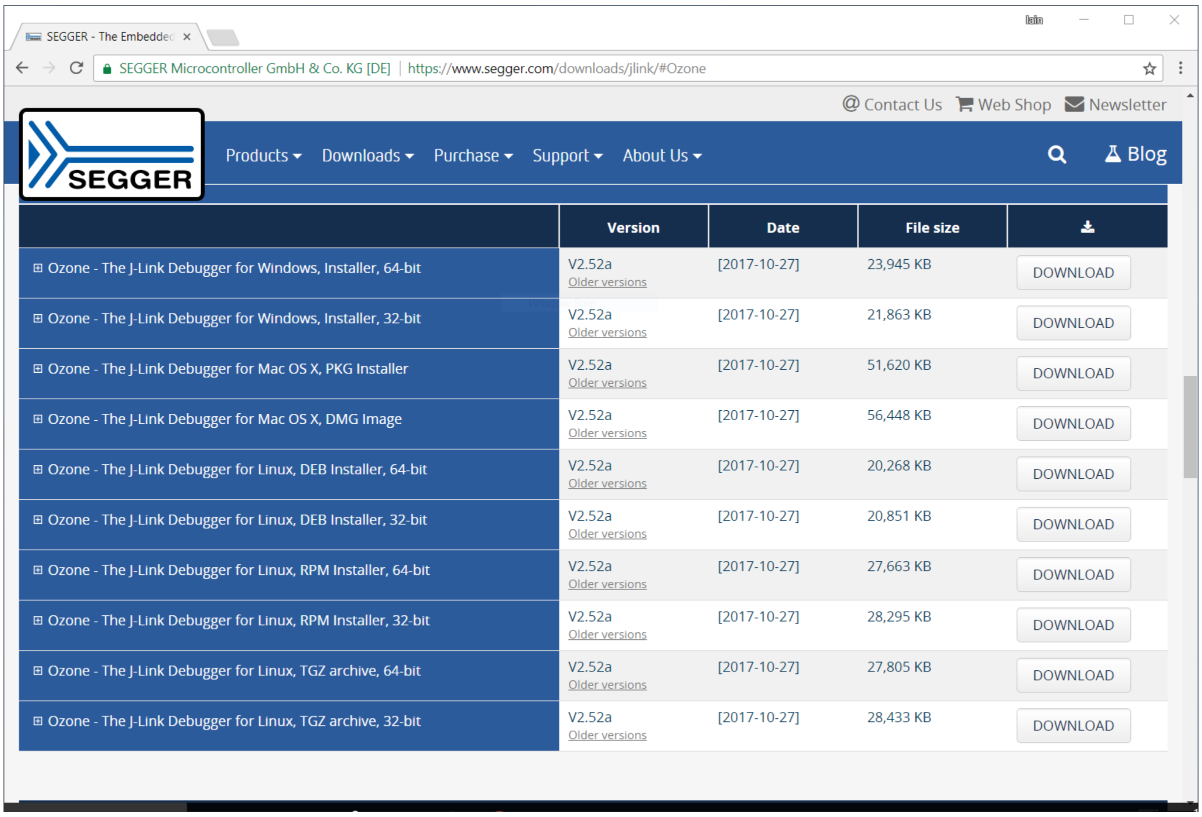
Failed downloading Segger Ozone 2.16d. Please visit the following url in order to download and install this tool. Latest version is preferred. Then specify the installation folder.

https://www.segger.com/ozone.html

Please specify the installation folder:

Figure 26: Automatic Ozone download failed

- Using a browser download the latest version of the appropriate package for the Linux distribution in use from https://www.segger.com/downloads/jlink/#Ozone



SEGGER - The Embedded ×

SEGGER Microcontroller GmbH & Co. KG [DE] | https://www.segger.com/downloads/jlink/#Ozone

Contact Us Web Shop Newsletter

Products Downloads Purchase Support About Us Blog

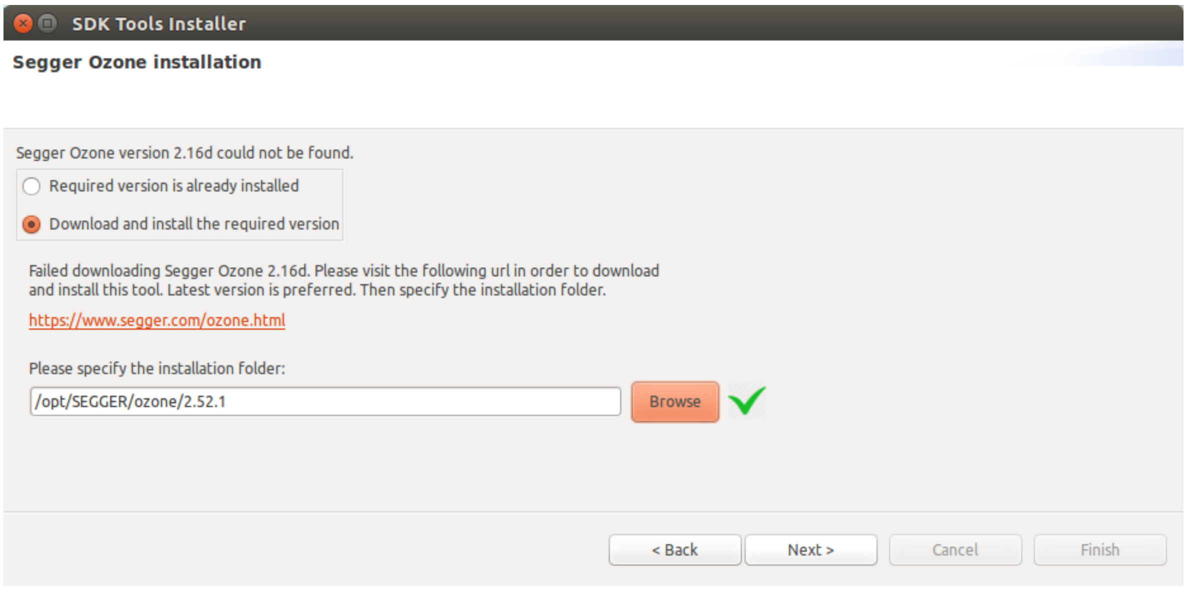
	Version	Date	File size	
⊞ Ozone - The J-Link Debugger for Windows, Installer, 64-bit	V2.52a Older versions	[2017-10-27]	23,945 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Windows, Installer, 32-bit	V2.52a Older versions	[2017-10-27]	21,863 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Mac OS X, PKG Installer	V2.52a Older versions	[2017-10-27]	51,620 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Mac OS X, DMG Image	V2.52a Older versions	[2017-10-27]	56,448 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Linux, DEB Installer, 64-bit	V2.52a Older versions	[2017-10-27]	20,268 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Linux, DEB Installer, 32-bit	V2.52a Older versions	[2017-10-27]	20,851 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Linux, RPM Installer, 64-bit	V2.52a Older versions	[2017-10-27]	27,663 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Linux, RPM Installer, 32-bit	V2.52a Older versions	[2017-10-27]	28,295 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Linux, TGZ archive, 64-bit	V2.52a Older versions	[2017-10-27]	27,805 KB	DOWNLOAD
⊞ Ozone - The J-Link Debugger for Linux, TGZ archive, 32-bit	V2.52a Older versions	[2017-10-27]	28,433 KB	DOWNLOAD

Figure 27: SEGGER Ozone download page

- For an Ubuntu 16.04 installation install the .deb package as follows.

```
~/Downloads$ sudo apt install ./ozone_2.52.1_x86_64.deb
```

- This will install the tool to `/opt/SEGGER/ozone/2.52.1`. Use the **Browse** button to find this folder and then press **Next**.



SDK Tools Installer

Segger Ozone installation

Segger Ozone version 2.16d could not be found.

Required version is already installed [✓ Download and install the required version]

Failed downloading Segger Ozone 2.16d. Please visit the following url in order to download and install this tool. Latest version is preferred. Then specify the installation folder.

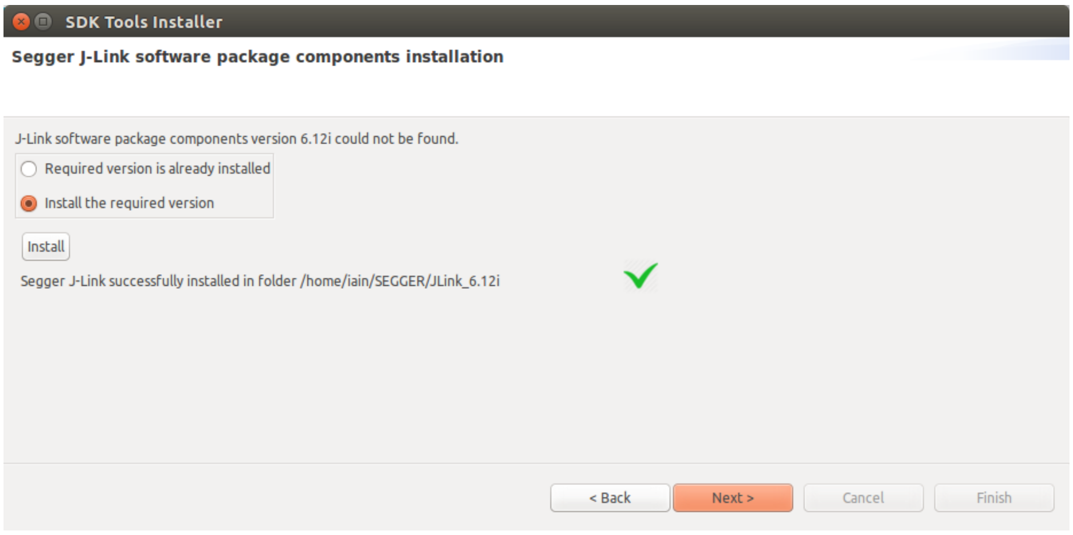
https://www.segger.com/ozone.html

Please specify the installation folder:

/opt/SEGGER/ozone/2.52.1

Figure 28: Set Ozone installation directory

- The next stage is to set the J-Link installation directory. This software is installed as part of Ozone install and so it is only necessary select **Install the required version** and press **Install**.



SDK Tools Installer

Segger J-Link software package components installation

J-Link software package components version 6.12i could not be found.

Required version is already installed

[✓ Install the required version]

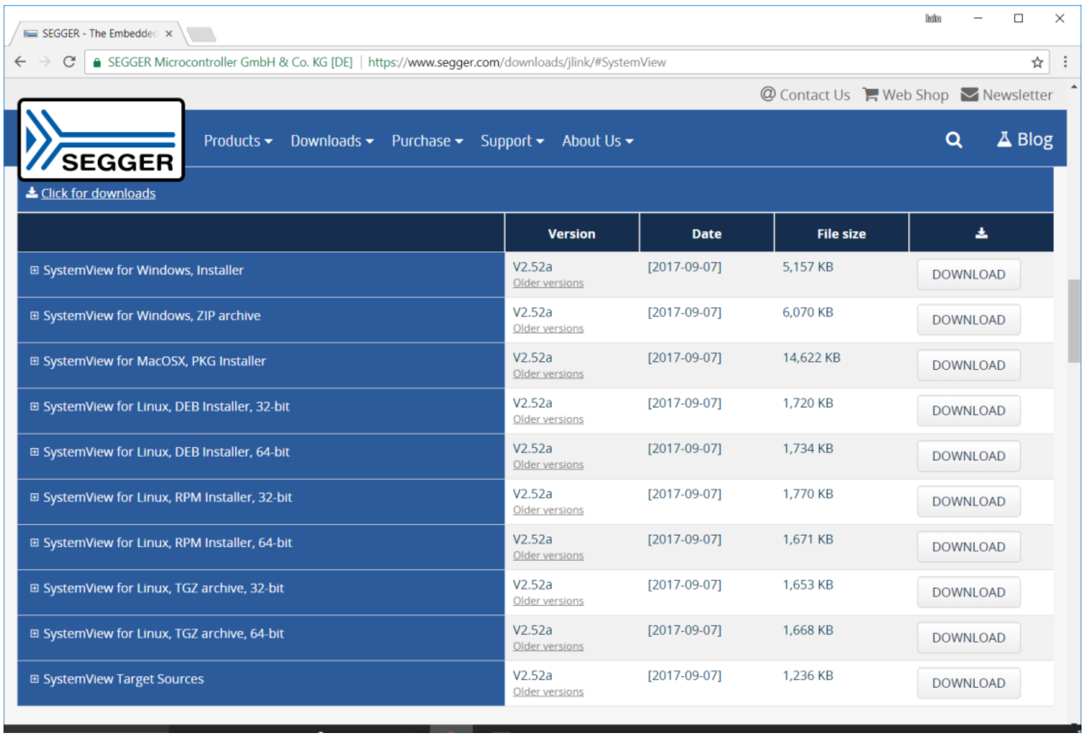
Install

Segger J-Link successfully installed in folder /home/iain/SEGGER/JLink_6.12i

< Back | Next > | Cancel | Finish

Figure 29: Set J-Link installation directory

- The final stage is to install SEGGER SystemView. The automatic download will fail so it must be done manually
- Using a browser download the appropriate Linux package for latest SystemView installer from https://www.segger.com/downloads/jlink/#SystemView. The latest version at the time of writing was v2.52a.



SEGGER Microcontroller GmbH & Co. KG [DE] | https://www.segger.com/downloads/jlink/#SystemView

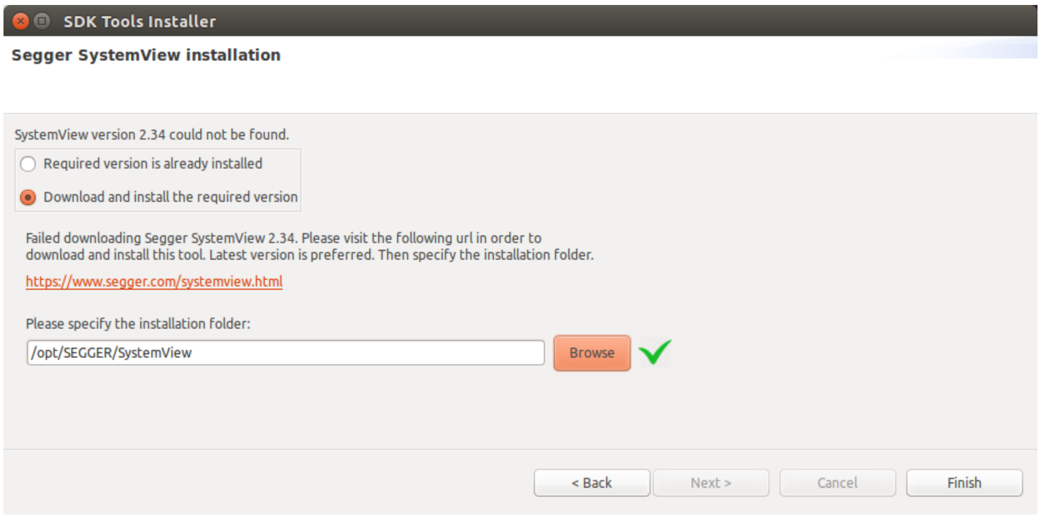
	Version	Date	File size	
SystemView for Windows, Installer	V2.52a Older versions	[2017-09-07]	5,157 KB	DOWNLOAD
SystemView for Windows, ZIP archive	V2.52a Older versions	[2017-09-07]	6,070 KB	DOWNLOAD
SystemView for MacOSX, PKG Installer	V2.52a Older versions	[2017-09-07]	14,622 KB	DOWNLOAD
SystemView for Linux, DEB Installer, 32-bit	V2.52a Older versions	[2017-09-07]	1,720 KB	DOWNLOAD
SystemView for Linux, DEB Installer, 64-bit	V2.52a Older versions	[2017-09-07]	1,734 KB	DOWNLOAD
SystemView for Linux, RPM Installer, 32-bit	V2.52a Older versions	[2017-09-07]	1,770 KB	DOWNLOAD
SystemView for Linux, RPM Installer, 64-bit	V2.52a Older versions	[2017-09-07]	1,671 KB	DOWNLOAD
SystemView for Linux, TGZ archive, 32-bit	V2.52a Older versions	[2017-09-07]	1,653 KB	DOWNLOAD
SystemView for Linux, TGZ archive, 64-bit	V2.52a Older versions	[2017-09-07]	1,668 KB	DOWNLOAD
SystemView Target Sources	V2.52a Older versions	[2017-09-07]	1,236 KB	DOWNLOAD

Figure 30: Download SystemView installer

- For an Ubuntu 16.04 installation install the .deb package as follows.

```
~/Downloads$ sudo apt install ./systemview_2.52.1_x86_64.deb
```

- This will install the tool to `/opt/SEGGER/SystemView`. Use the **Browse** button to find this folder and then press **Finish**.



SDK Tools Installer

Segger SystemView installation

SystemView version 2.34 could not be found.

Required version is already installed [✓ Download and install the required version]

Failed downloading Segger SystemView 2.34. Please visit the following url in order to download and install this tool. Latest version is preferred. Then specify the installation folder.

https://www.segger.com/systemview.html

Please specify the installation folder:

/opt/SEGGER/SystemView

Figure 31: Set J-Link installation directory

- SmartSnippetsTM Studio is now installed.

7.3 Extracting and using the SDK

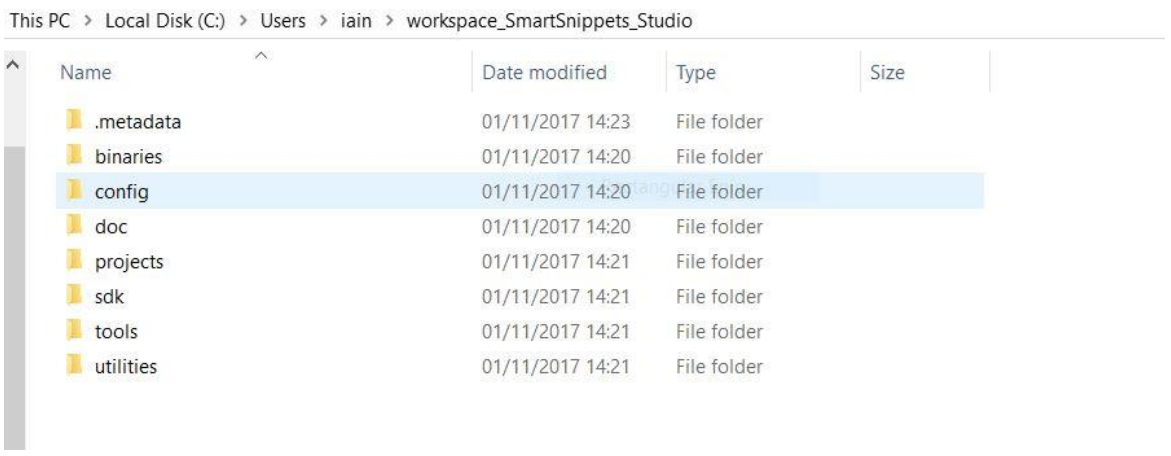
As SmartSnippetsTM Studio is an Eclipse based Integrated Development Environment (IDE) all the source files are contained within a workspace folder which contains all the project sources, build configurations etc.

The SmartSnippetsTM SDK has been developed in a way that means that the workspace directory and the SDK installation directory are the same. The configuration process for the SDK is easier if the SDK has already been installed to the workspace folder.

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

Download the SmartSnippetsTM SDK from Software Development Kit section of Software and Tools tab on the product page https://support.dialog-semiconductor.com/connectivity/product/da14680.

Extract the contents of the SDK zip file to the workspace folder. The workspace must not contain the SDK directory (i.e. `DA1468x_DA15xxx_SDK_1.0.10.1072`) otherwise the projects will not build. It must contain the contents of the directory as shown in Figure 32. This shows a Windows filesystem, the principle is the same for a Linux filesystem.



This PC > Local Disk (C:) > Users > iain > workspace_SmartSnippets_Studio

Name	Date modified	Type	Size
.metadata	01/11/2017 14:23	File folder	
binaries	01/11/2017 14:20	File folder	
config	01/11/2017 14:20	File folder	
doc	01/11/2017 14:20	File folder	
projects	01/11/2017 14:21	File folder	
sdk	01/11/2017 14:21	File folder	
tools	01/11/2017 14:21	File folder	
utilities	01/11/2017 14:21	File folder	

Figure 32: Extract SDK to Workspace (on Windows)

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.

7.4 Starting SmartSnippetsTM Studio

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

- Select the workspace folder for SmartSnippetsTM Studio. This should be the created directory workspace_SmartSnippets_Studio.
- If asked specify which product family SmartSnippetsTM Studio should assume the code is for (DA1468x 1.0.10 SDK or DA1458x 6.0.4 SDK).
- All the other tools required by the SDK and SmartSnippetsTM Studio will be automatically installed, such as GNU ARM GCC, SmartSnippets Toolbox, SEGGER drivers etc.

7.5 Additional Software

SmartSnippetsTM Toolbox is installed with SmartSnippetsTM Studio. It 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.

SmartSnippetsTM 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 DA1468x 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 SmartSnippetsTM DA1468x SDK. More details about CLI programmer are available in Appendix C in [4].

The SmartSnippetsTM 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 Your First DA1468x Application – Blinky

8.1 Introduction

The following sections explain how the user can build, program and run a simple software application called `Blinky` on the ProDK development board using the SmartSnippetsTM DA1468x SDK.

This application is based on a pre-defined project template called `freertos_retarget_template` which is located at `<sdk_root_directory>\projects\dk_apps\templates`. After modification the application will toggle an on-board LED with a pre-defined frequency.

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

8.2 Software Architecture

The `freertos_retarget_template` project is set to run in release mode from DA1468x internal RAM by default. 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, trim values are applied to the DA1468x device. When the clock has been set up properly, the main routine creates task `SysInit` and starts the RTOS scheduler. Now the RTOS scheduler is running it will start its first task which is `SysInit`.

In the `SysInit` task the power and clock management module is started. Next, the hardware peripherals which are used by this particular project are initialized. This includes setting up the correct GPIOs, enabling clocks and setting initial values to the hardware peripherals in use. Last thing done before `SysInit` task exits is to create another task which is the main application task running until the program gets stopped. The function code implementing the main task is as follows:

```
static void prvTemplateTask( void *pvParameters )
{
        OS_TICK_TIME xNextWakeTime;
        static uint32_t test_counter=0;

        /* Initialise xNextWakeTime - this only needs to be done once. */
        xNextWakeTime = OS_GET_TICK_COUNT();

             for( ;; ) {
                /* Place this task in the blocked state until it is time to run again.
                   The block time is specified in ticks, the constant used converts ticks
                   to ms.  While in the Blocked state this task will not consume any CPU
                   time. */
                vTaskDelayUntil( &xNextWakeTime, mainCOUNTER_FREQUENCY_MS );
                test_counter++;

                if (test_counter % (1000 / OS_TICKS_2_MS(mainCOUNTER_FREQUENCY_MS)) == 0)
{
                        printf("#");
                        fflush(stdout);
                }
        }
}
```

Code 1: The main task in SysInit() the prvTemplateTask()

The software as provided in the SmartSnippetsTM DA1468x SDK – once up and running - interacts with a host PC and sends the character '#' via the serial UART interface every 1sec. This can be verified by setting up a terminal application (see 6.3.3 for the procedure, with the only difference that now the baud rate must be set to 115200).

The software shall be modified to toggle LED `D2` every 200ms (Figure 33), connected to pin `P1_5`.

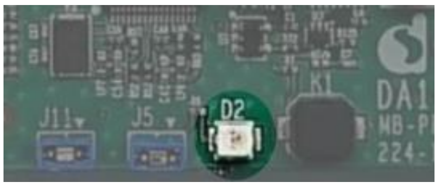


Figure 33: LED D2 on ProDK

The software first needs to reserve and initialize the selected GPIO. This typically takes place inside the `periph_init()` routine which is located inside the `main.c` file. This is done by adding the following code to the end of this function:

```
hw_gpio_set_pin_function(HW_GPIO_PORT_1,HW_GPIO_PIN_5, HW_GPIO_MODE_OUTPUT,
HW_GPIO_FUNC_GPIO);
```

Code 2: Set function for GPIO

This call makes sure pin 5 of port 1 is set as GPIO output. To toggle the GPIO every 200ms the main routine inside the `prvTemplateTask()` routine needs to be **modified** as follows:

```
for( ;; )
                {
/* Place this task in the blocked state until it is time to run again. The block time is specified in
ticks, the constant used convertsticks to ms.  While in the Blocked state this task will not consume
any CPU time. */
                        vTaskDelayUntil( &xNextWakeTime, mainCOUNTER_FREQUENCY_MS
);
                        test_counter++;

                        /* make sure P1_5 is set and reset alternatively */
                        if(test_counter % 2)
                            hw_gpio_set_active(HW_GPIO_PORT_1, HW_GPIO_PIN_5 );
                        else
                            hw_gpio_set_inactive(HW_GPIO_PORT_1, HW_GPIO_PIN_5 );
                }
```

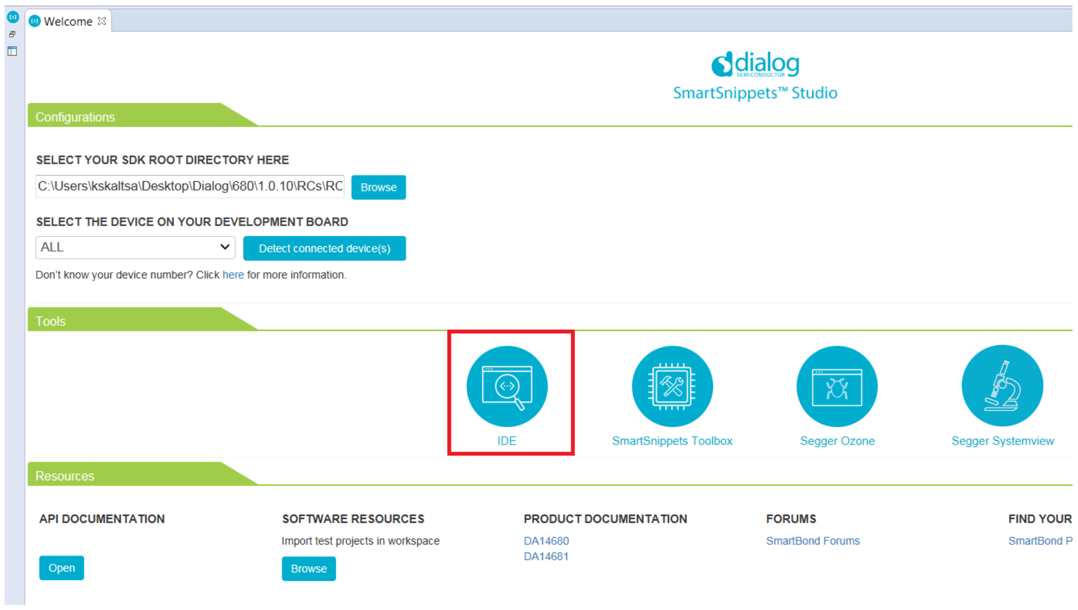
Code 3: The main routine inside the prvTemplateTask()

In Code 3 the `"if"` statement has been **modified**. The routines (`hw_gpio_set_active()` and `hw_gpio_set_inactive()`) are an example of how to use low level drivers (LLD). These APIs are defined in `hw_gpio.h` **(**located in `<sdk_root_directory>/peripherals/include`**).**

8.3 Software Build

As already described in the previous section the SmartSnippetsTM DA1468x SDK contains a template project featuring the FreeRTOS OS which will be used as a starting point to develop a customized SW project for the DA1468x family of devices. This section describes all the steps required to import, build and run this first project.

1. In the SmartSnippetsTM Studio welcome page click on the IDE icon from the Tools tab as shown in Figure 34.



SELECT YOUR SDK ROOT DIRECTORY HERE

C:\Users\kskaltsa\Desktop\Dialog\680\1.0.10\RCs\RC

SELECT THE DEVICE ON YOUR DEVELOPMENT BOARD

ALL

Detect connected device(s)

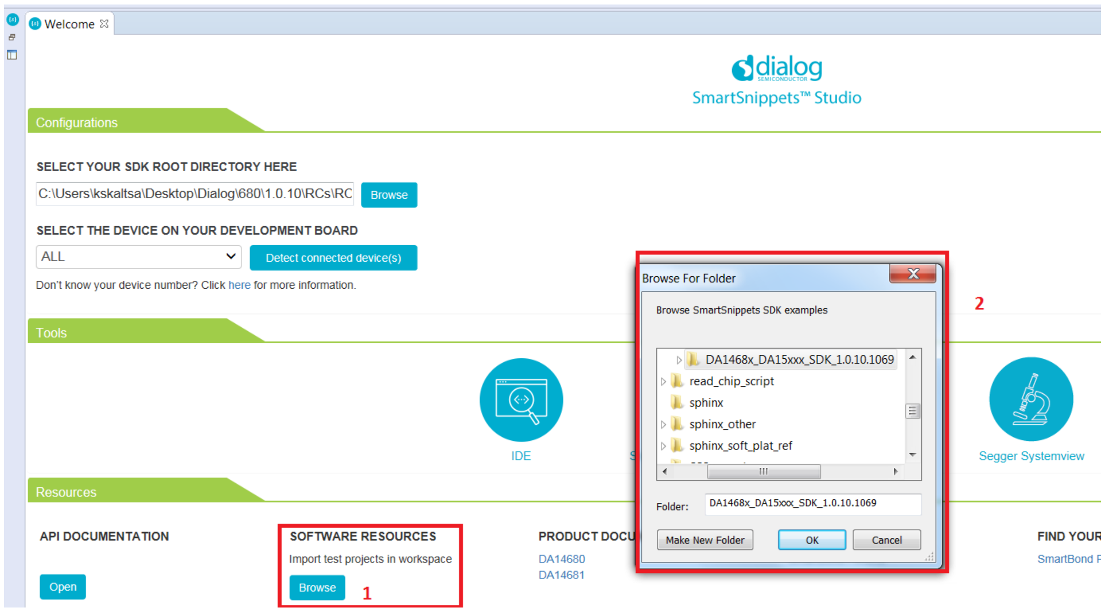
Don't know your device number? Click here for more information.

API DOCUMENTATION | SOFTWARE RESOURCES | PRODUCT DOCUMENTATION | FORUMS | FIND YOUR

Import test projects in workspace

Figure 34: SmartSnippetsTM Studio welcome page

2. Import the template project `template_freertos_retarget` from: `<sdk_root directory>\projects\dk_apps\templates\freertos_retarget` 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 35.
3.



SELECT YOUR SDK ROOT DIRECTORY HERE

C:\Users\kskaltsa\Desktop\Dialog\680\1.0.10\RCs\RC

SELECT THE DEVICE ON YOUR DEVELOPMENT BOARD

ALL

Detect connected device(s)

Don't know your device number? Click here for more information.

Browse For Folder

Browse SmartSnippets SDK examples

- DA1468x_DA15xxx_SDK_1.0.10.1069
- read_chip_script
- sphinx
- sphinx_other
- sphinx_soft_plat_ref

Folder: DA1468x_DA15xxx_SDK_1.0.10.1069

API DOCUMENTATION | SOFTWARE RESOURCES | PRODUCT DOCU | FIND YOUR

Import test projects in workspace

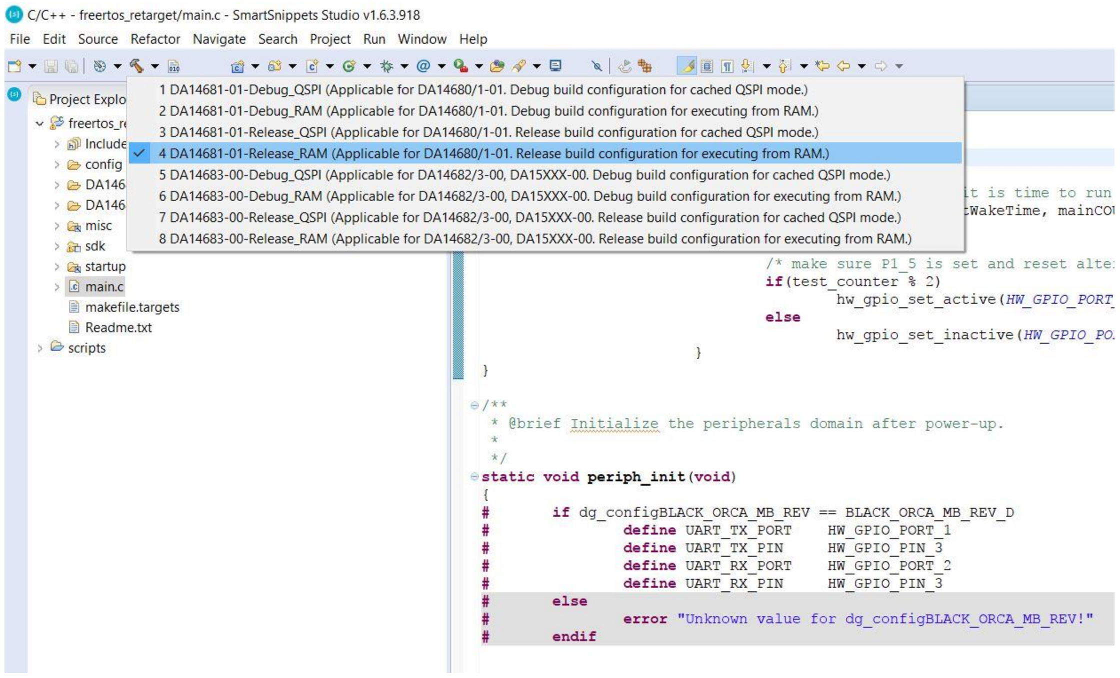
Figure 35: Project import

4. In the same way import the `scripts` project from: `<sdk_root directory>\utilities\scripts`
5. Modify the file `main.c` using the text editor inside Eclipse and save the changes. Replace the Code 1 in `prvTemplateTask()` function with Code 3 as described in detail in 8.2.
6. Extend `periph_init()` function inside `main.c` to set the function for the pin driving the external GPIO by adding Code 2 as described in detail in 8.2.

8.3.1 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 ( ) with the Release RAM configuration `DA14681-01-Release-RAM` as shown in Figure 36.


C/C++ - freertos_retarget/main.c - SmartSnippets Studio v1.6.3.918
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explo
freertos_r
Include
config
DA146
DA146
misc
sdk
startup
main.c
makefile.targets
Readme.txt
scripts

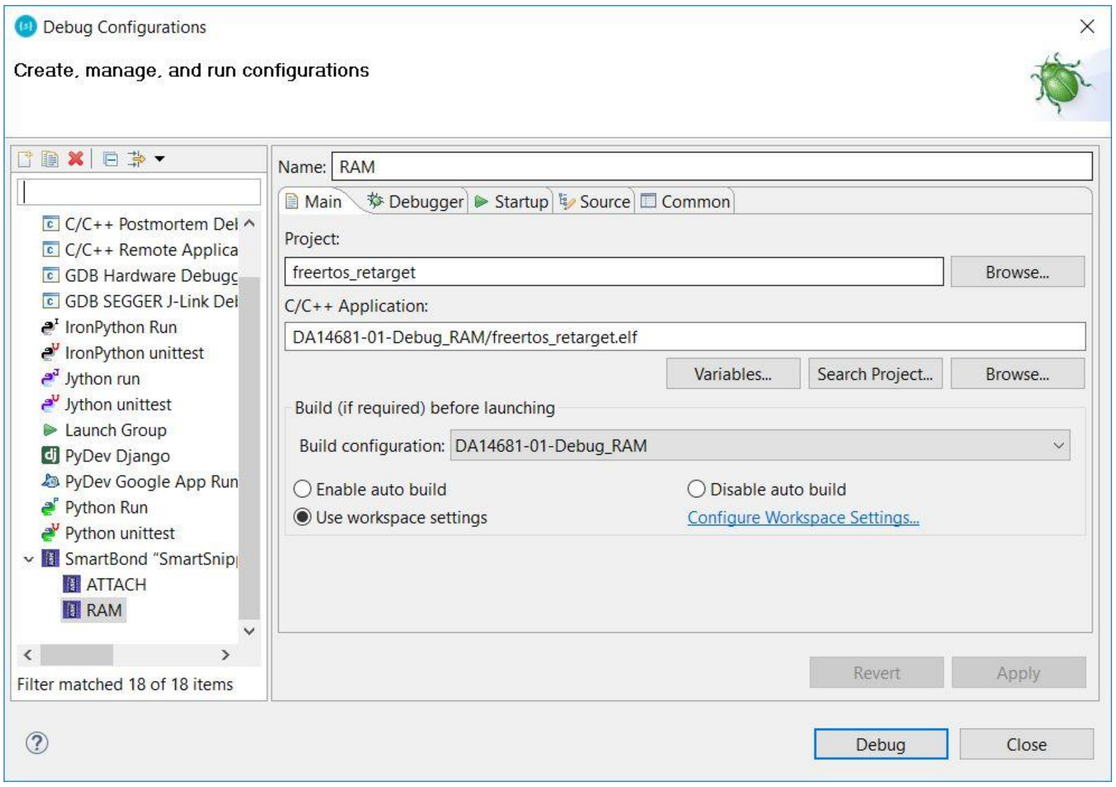
1 DA14681-01-Debug_QSPI (Applicable for DA14680/1-01. Debug build configuration for cached QSPI mode.)
2 DA14681-01-Debug_RAM (Applicable for DA14680/1-01. Debug build configuration for executing from RAM.)
3 DA14681-01-Release_QSPI (Applicable for DA14680/1-01. Release build configuration for cached QSPI mode.)
✓ 4 DA14681-01-Release_RAM (Applicable for DA14680/1-01. Release build configuration for executing from RAM.)
5 DA14683-00-Debug_QSPI (Applicable for DA14682/3-00, DA15XXX-00. Debug build configuration for cached QSPI mode.)
6 DA14683-00-Debug_RAM (Applicable for DA14682/3-00, DA15XXX-00. Debug build configuration for executing from RAM.)
7 DA14683-00-Release_QSPI (Applicable for DA14682/3-00, DA15XXX-00. Release build configuration for cached QSPI mode.)
8 DA14683-00-Release_RAM (Applicable for DA14682/3-00, DA15XXX-00. Release build configuration for executing from RAM.)

```
                                                                   it is time to run
                                                                   tWakeTime, mainCO
                                             /* make sure P1_5 is set and reset alte
                                             if(test_counter % 2)
                                                     hw_gpio_set_active(HW_GPIO_PORT_
                                             else
                                                     hw_gpio_set_inactive(HW_GPIO_PO
                                     }
}

/**
 * @brief Initialize the peripherals domain after power-up.
 *
 */
static void periph_init(void)
{
#       if dg_configBLACK_ORCA_MB_REV == BLACK_ORCA_MB_REV_D
#               define UART_TX_PORT    HW_GPIO_PORT_1
#               define UART_TX_PIN     HW_GPIO_PIN_3
#               define UART_RX_PORT    HW_GPIO_PORT_2
#               define UART_RX_PIN     HW_GPIO_PIN_3
#       else
#               error "Unknown value for dg_configBLACK_ORCA_MB_REV!"
#       endif
```

Figure 36: Build Blinky in Release RAM configuration

Once the Release RAM binary is built, the next step is to start the Debugger using **Run > Debug Configurations > SmartBond “SmartSnippets DA1468x SDK via J-Link GDB Server > RAM** as shown in Figure 37. 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.



Name: RAM

Project: freertos_retarget

C/C++ Application: DA14681-01-Debug_RAM/freertos_retarget.elf

Build (if required) before launching

Build configuration: DA14681-01-Debug_RAM

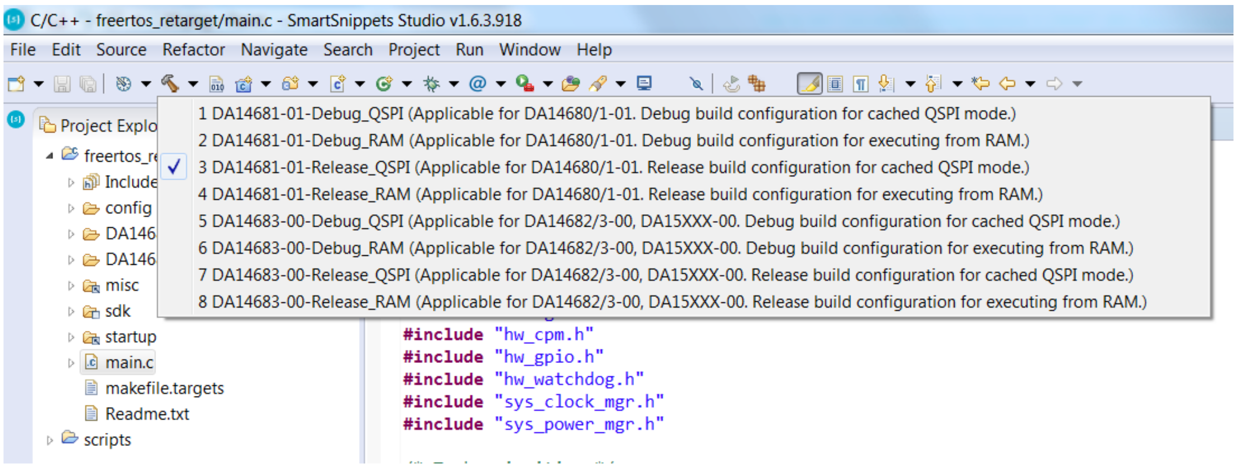
Enable auto build [✓ Use workspace settings] Disable auto build

Configure Workspace Settings...

Figure 37: Start Debug in RAM mode

8.3.2 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.
2. Build the project using the build icon (🔨▾) and select a QSPI configuration as shown in Figure 38.
3.


C/C++ - freertos_retarget/main.c - SmartSnippets Studio v1.6.3.918
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explo
freertos_re
Include
config
DA146
DA146
misc
sdk
startup
main.c
makefile.targets
Readme.txt
scripts

1 DA14681-01-Debug_QSPI (Applicable for DA14680/1-01. Debug build configuration for cached QSPI mode.)
2 DA14681-01-Debug_RAM (Applicable for DA14680/1-01. Debug build configuration for executing from RAM.)
✓ 3 DA14681-01-Release_QSPI (Applicable for DA14680/1-01. Release build configuration for cached QSPI mode.)
4 DA14681-01-Release_RAM (Applicable for DA14680/1-01. Release build configuration for executing from RAM.)
5 DA14683-00-Debug_QSPI (Applicable for DA14682/3-00, DA15XXX-00. Debug build configuration for cached QSPI mode.)
6 DA14683-00-Debug_RAM (Applicable for DA14682/3-00, DA15XXX-00. Debug build configuration for executing from RAM.)
7 DA14683-00-Release_QSPI (Applicable for DA14682/3-00, DA15XXX-00. Release build configuration for cached QSPI mode.)
8 DA14683-00-Release_RAM (Applicable for DA14682/3-00, DA15XXX-00. Release build configuration for executing from RAM.)

```
#include "hw_cpm.h"
#include "hw_gpio.h"
#include "hw_watchdog.h"
#include "sys_clock_mgr.h"
#include "sys_power_mgr.h"
```

Figure 38: Build Blinky in Release QSPI configuration

The next step is to write the binary file to QSPI Flash. This is done using a script selected using the External Tool Configurations button. In Figure 39 the script `program_qspi_jtag_win` is used to program the QSPI Flash memory. Alternatively, use **Run > External Tools > program_qspi_jtag_win**.

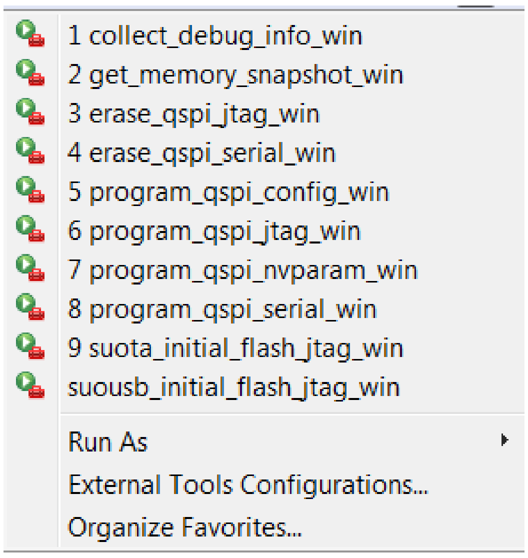
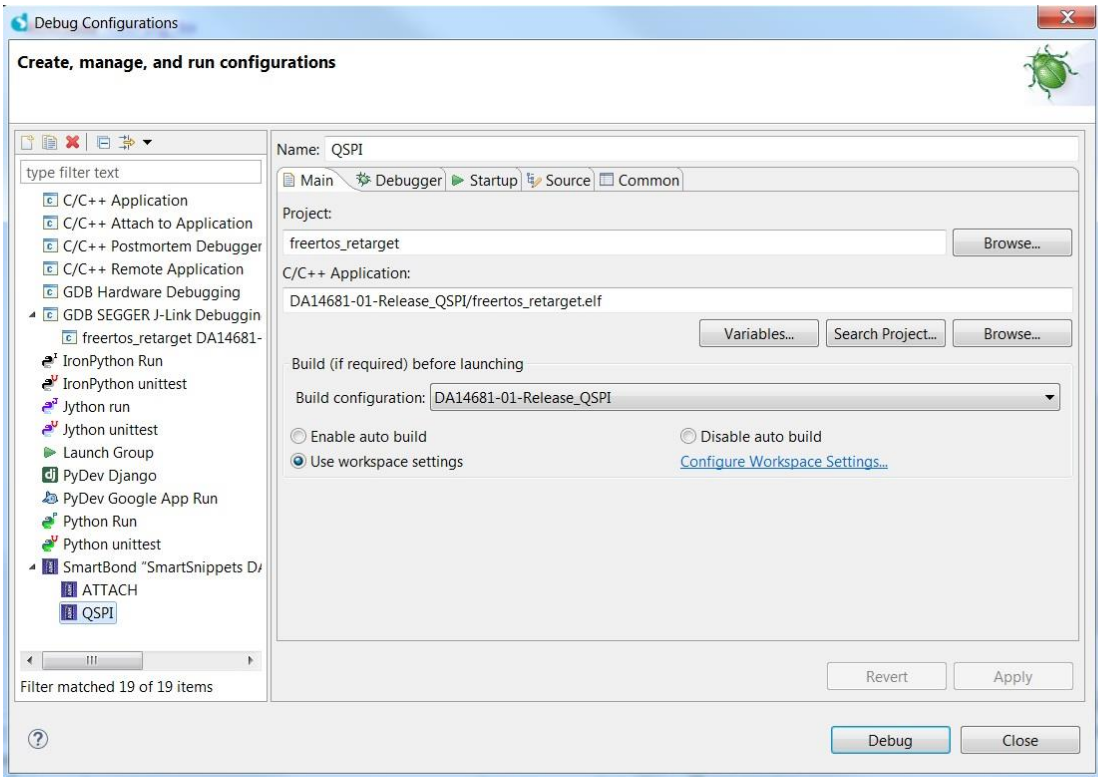

1 collect_debug_info_win
2 get_memory_snapshot_win
3 erase_qspi_jtag_win
4 erase_qspi_serial_win
5 program_qspi_config_win
6 program_qspi_jtag_win
7 program_qspi_nvparam_win
8 program_qspi_serial_win
9 suota_initial_flash_jtag_win
suousb_initial_flash_jtag_win
Run As
External Tools Configurations...
Organize Favorites...

Figure 39: Write Blinky to QSPI Flash

On a Linux machine the equivalent options are suffixed with `_linux` rather than `_win`. So, use `program_qspi_jtag_linux` instead.

Finally start the debugger using **Run >Debug configurations > Smartbond “SmartSnippets DA1468x SDK” > QSPI** and click ‘Debug’ as shown in Figure 40. This will start the debug perspective in Eclipse and load the symbols for the current project into the debugger.

4.



Name: QSPI

Project: freertos_retarget

C/C++ Application: DA14681-01-Release_QSPI/freertos_retarget.elf

Build (if required) before launching

Build configuration: DA14681-01-Release_QSPI

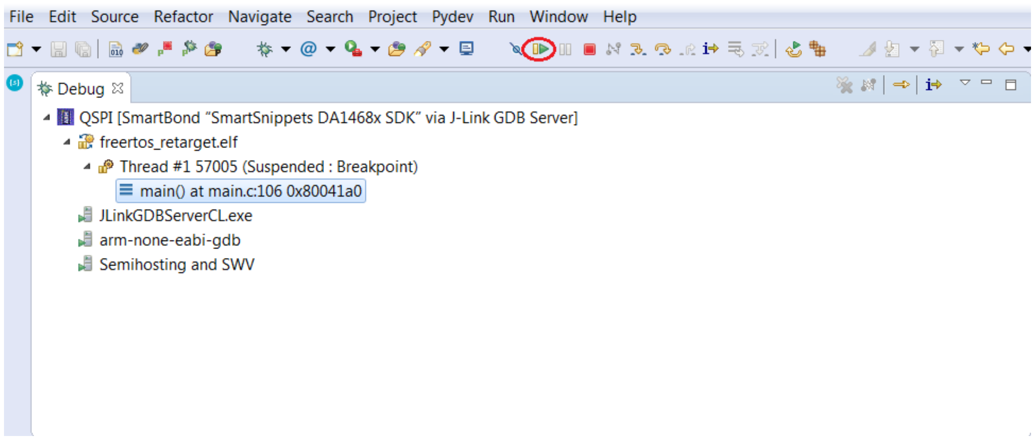
Enable auto build [✓ Use workspace settings] Disable auto build

Configure Workspace Settings...

Figure 40: Start Debug in QSPI mode

8.4 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 `Blinky` project by selecting **Resume** inside the Eclipse Run menu or by hitting the **play** icon as indicated in Figure 41.


File Edit Source Refactor Navigate Search Project Pydev Run Window Help
Debug

QSPI [SmartBond "SmartSnippets DA1468x SDK" via J-Link GDB Server]
freertos_retarget.elf
Thread #1 57005 (Suspended : Breakpoint)
main() at main.c:106 0x80041a0
JLinkGDBServerCL.exe
arm-none-eabi-gdb
Semihosting and SWV

Figure 41: Executing the `Blinky` project in Eclipse

3. Correct functionality of the built and downloaded `Blinky` project can be checked by watching the Orange LED D2 on the ProDK board (Figure 33). It should now blink with a frequency of around 2Hz.

8.5 Troubleshooting

Table 2: Troubleshooting Blinky

Issue	Note
Software execution stopped at `ASSERT_WARNING(is_compatible_chip_version());`	Project has not been compiled using the correct silicon version. Make sure the following parameters have been properly configured. (By default silicon version is taken care by the corresponding build configuration). Make sure parameter `dg_configBLACK_ORCA_IC_STEP` and `dg_configBLACK_ORCA_IC_REV` are set to the correct chip revision and the ProDK version during built. (Located in `<sdk_root_directory>/bsp/config/bsp_definitions.h`)
LED does not blink	The GPIO which was used to drive the external LED was P1_5 for this sample code. This might have to be adopted in case a different revision of the ProDK board is in use. To find the correct pin for your version of the ProDK board please check the [5] for details. Make also sure that `dg_configBLACK_ORCA_MB_REV` is set to the correct chip revision and the ProDK version during built. .(Located in `<sdk_root_directory>/bsp/config/bsp_definitions.h`)

Revision history

Revision	Date	Description
1.0	19-Nov-2015	First released version
2.0	22-Apr-2016	Update for SmartSnippets DA168x SDK Release 1.0.4.812
2.1	17-Jun-2016	Update for SmartSnippets DA168x SDK Engineering Release 1.0.5.885
3.0	26-Jul-2016	Update for SmartSnippets DA168x SDK Release 1.0.6.968
4.0	07-Dec-2016	Update for SmartSnippets DA168x SDK Release 1.0.8
4.1	09-Dec-2016	Update for SmartSnippets DA168x SDK Release 1.0.8
5.0	21-Jul-2017	Update for SmartSnippets DA168x SDK Release 1.0.10
5.1	01-Nov-2017	Update for SmartSnippets DA168x SDK Release 1.0.10
6.0	14-Dec-2017	Update for SmartSnippets DA168x SDK Release 1.0.12
7.0	23-Jul-2018	Update for SmartSnippets DA168x SDK Release 1.0.14
7.1	23-Feb-2022	Updated logo, disclaimer, copyright.

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.