



SmartSnippets User Guide Documentation

Release 3.8

May 14, 2015

Contents

1	Revision History	2
2	Introduction	2
2.1	Scope	2
2.2	Framework	3
	The toolbox	4
	The Horizontal toolbar	4
3	Getting Started	4
3.1	Installation Notes	4
3.2	Work Flow	5
3.3	Project and Virtual COM port / JTAG selection	6
3.4	Board Setup	8
4	Booter	9
4.1	Downloading Code	10
5	UART Terminal	10
6	Power Profiler	11
6.1	Power Profiler software cursors	12
6.2	Power Profiler control panel	13
6.3	Power Profiler modes	13
6.4	Power Profiler actions	13
6.5	Chart management via mouse	13
6.6	Chart management via keyboard	14
6.7	Power Profiler configuration	14
7	Sleep Mode Advisor	16
8	OTP Programmer	18
8.1	OTP Image	18
8.2	OTP Header	20
8.3	OTP NVDS	23

9 SPI Flash Programmer	23
10 EEPROM Programmer	24
11 Proprietary Header Programmer	25
12 OTA Services (over the air services)	27
12.1 Software patch over the air (SPOTA)	27
Link Establishment and Termination	28
Patch Download	29
12.2 Software update over the air (SUOTA)	29
13 Data Rate Monitor	31
14 Working with multiple tools	33
15 Working with multiple projects	33
16 Logs	33
17 Command-line implementation	34

Contents:

1 Revision History

Version	Date(M-D-Y)	Description
1.0	08-11-2013	Initial version
2.0	20-12-2013	Updated version for SmartSnippets version 2.0
3.0 beta	20-02-2014	Updated version for SmartSnippets version 3.0 beta
3.0 beta 3	28-02-2014	Added documentation for new CLI option '-chip'
3.0 beta 4	07-03-2014	Updated version for SmartSnippets version 3.0 beta 4
3.0	24-03-2014	Updated version for SmartSnippets version 3.0
3.1	17-06-2014	Updated version for SmartSnippets version 3.1
3.2	17-07-2014	Updated version for SmartSnippets version 3.2
3.3	06-08-2014	Updated version for SmartSnippets version 3.3
3.4	19-08-2014	Updated version for SmartSnippets version 3.4
3.5	23-09-2014	Updated version for SmartSnippets version 3.5
3.6	3-11-2014	Updated version for SmartSnippets version 3.6
3.7	6-02-2015	Updated version for SmartSnippets version 3.7
3.8	14-05-2015	Updated version for SmartSnippets version 3.8

2 Introduction

2.1 Scope

The SmartSnippets framework is provided with Dialog's Development Kit of the DA14580 Bluetooth Smart chipset. It is targeting the main activities of programming and optimizing code for best power performance. It enables:

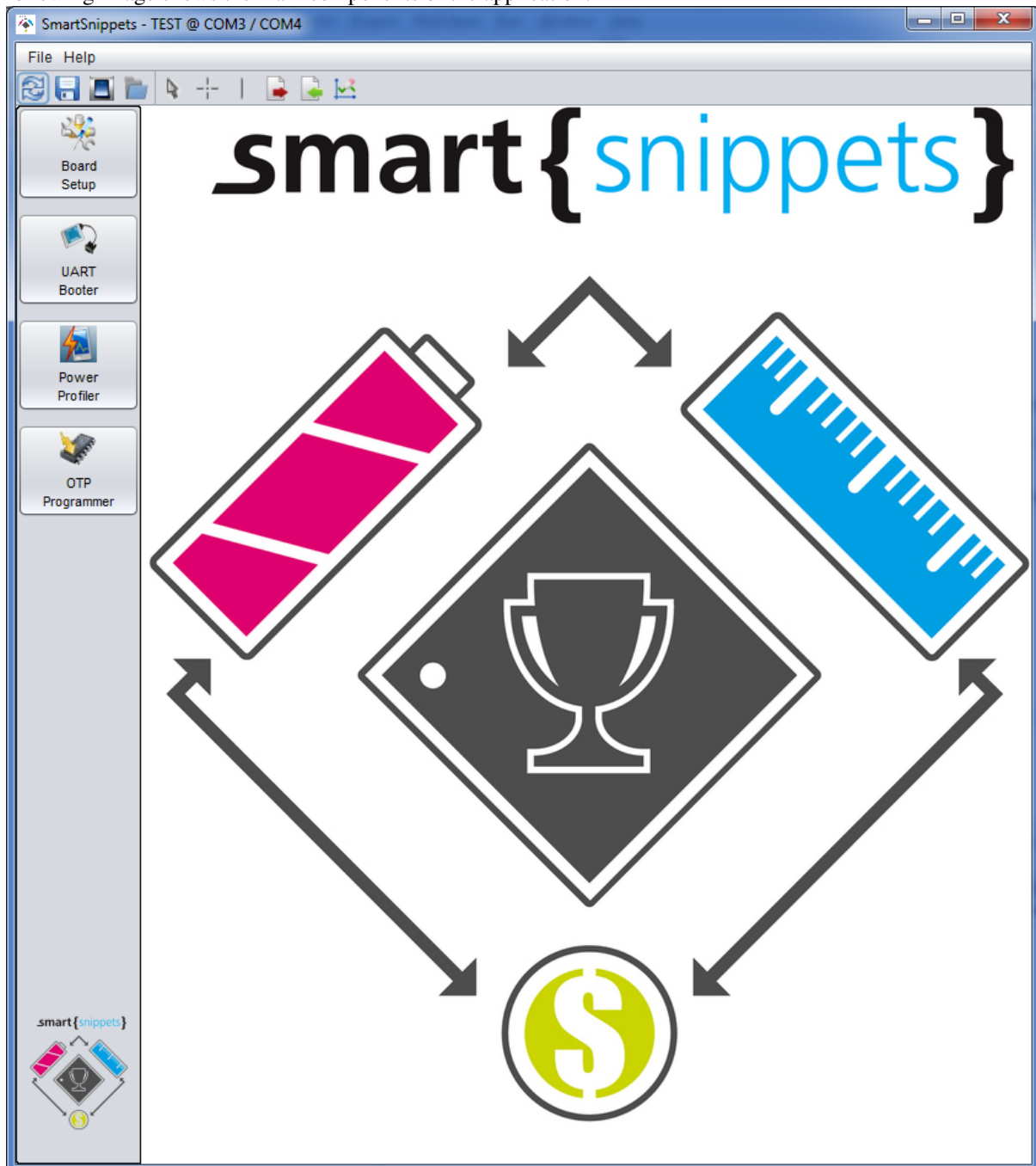
- programming the internal OTP with the actual application compiled image
- accurate examination of the power profile and how it is affected by application software

- downloading a SW image to SRAM over UART and execute

The SmartSnippets framework makes maximum use of the available features on the motherboard and thus allowing developers of Bluetooth smart applications to work without expensive and bulky equipment. The tool will provide full visibility on the chip activity, which is crucial in developing ultra low power wireless applications.

2.2 Framework

SmartSnippets is practically a framework that enables multiple tools be hosted under the same environment. The following image shows the main components of the application:














The toolbox

The toolbox contains the list of all available tools and is located on the left-hand side of the application window. Each tool is accessible through a dedicated button. By clicking on a button, the corresponding window becomes visible and takes focus. The available tools are:

1. Board Setup
2. UART Booter
3. Power Profiler
4. OTP Programmer
5. SPI Flash Programmer
6. EEPROM Programmer
7. Software Patch over the Ait (SPotA)
8. Sleep Mode Advisor

Moving the mouse over the button will provide a small description of each SmartSnippets tool.

The Horizontal toolbar

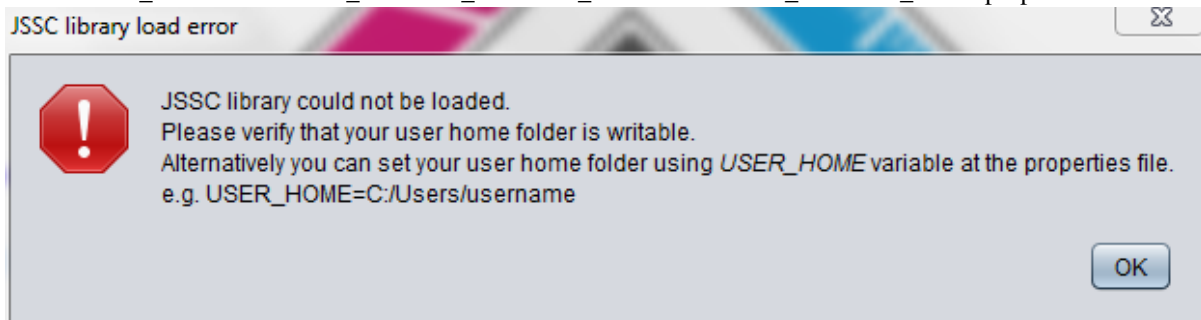
Icon	Description
	Opens the 'Project and Virtual COM port selection' screen. The user can change the selected project and / or the selected virtual COM port pair
	Saves the project information to the 'project.sms' file. Apart from the project name, description and the selected virtual COM port pair, it also saves information entered by the user at every SmartSnippets tool. The project will also be automatically saved when the user exits the application
	Opens or closes the vertical toolbar
	Loads the default layout of tools. Can also be used to open all the SmartSnippets tools at once in a grid-like layout
	Selects and moves a Time Marker at the Power Profiler diagram
	Measures the difference in time and current between two points in the Power Profiler diagram
	Adds a Time Marker to the Power Profiler diagram
	Exports Power Profiler data to CSV format
	Imports data on Power Profiler from CSV file
	Clears the secondary current data and hides it from the chart view
	Takes an image snapshot of the Power Profiler chart window in .png format

3 Getting Started

3.1 Installation Notes

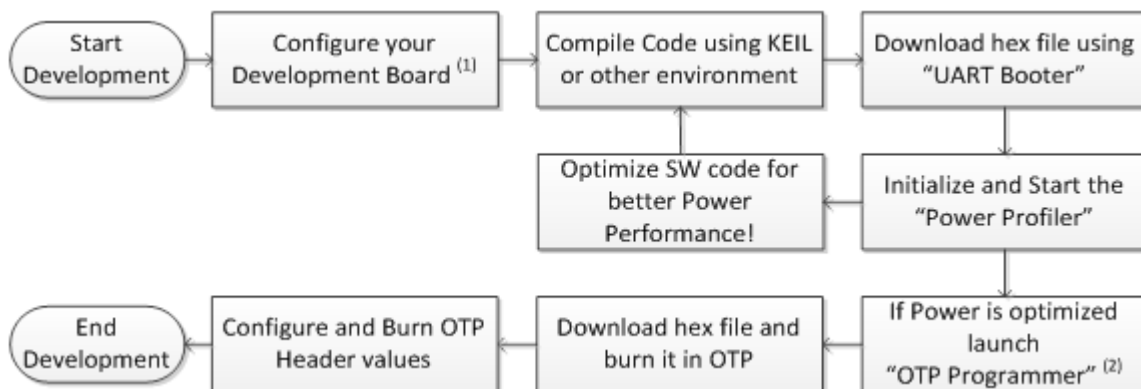
1. SmartSnippets supports both 32-bit and 64-bit environments. For win32, please use the 'SmartSnippets_install_win32.exe' installer. For win64, please use the 'SmartSnippets_install_win64.exe' installer.

2. To use a DA14580 board, the appropriate FTDI drivers need to be installed on the system. Just use the standard Windows 'new hardware has been found' wizard to install them.
3. To use a DA14580-01 Bluetooth dongle, the appropriate JLINK drivers need to be installed on the system. The latest drivers can be found at <http://www.segger.com/jlink-software.html>. To download them, please click on the 'Download' button under the 'J-Link software & documentation pack for Windows' section and then choose the option 'I do not have a serial number because I own an eval board with J-Link on-board. How can I download J-Link software for it?'. While installing these drivers, it is recommended that the Bluetooth dongle is not connected to the USB port.
4. If an error occurs while using the SPotA tool and trying to connect to the Bluetooth dongle, please make sure that the 'Microsoft Visual C++ 2010 Redistributable Package' libraries are installed at C:\Windows\System32 folder. Otherwise, please download and install the latest Redistributable libraries for Visual Studio 2010 from <http://support.microsoft.com/kb/2019667/>.
5. If the following error dialog appears, Java Simple Serial Connector library (JSSC), that is used by Smart-Snippets for serial communication, could not be loaded successfully. The reason is that none of the user home folder and user Temp folder is writable. JSSC library extracts to user home folder (or user Temp folder, if user home folder is not writable) the required DLLs for serial connection. The user can change the access rights of these folders or overwrite the default user folder location. In order to overwrite the default user home path, the user can use USER_HOME property. For example the user can set USER_HOME=C:/Users/_username_ or USER_HOME=C:\Users_username_ at the properties file.



3.2 Work Flow

This section is providing a guide to start using the SmartSnippets framework, a proposed way of working with the tools available for the basic tasks during development. This basic flow of working is illustrated in the following figure:



[1] Dialog's DA14580 Development Board needs no configuration

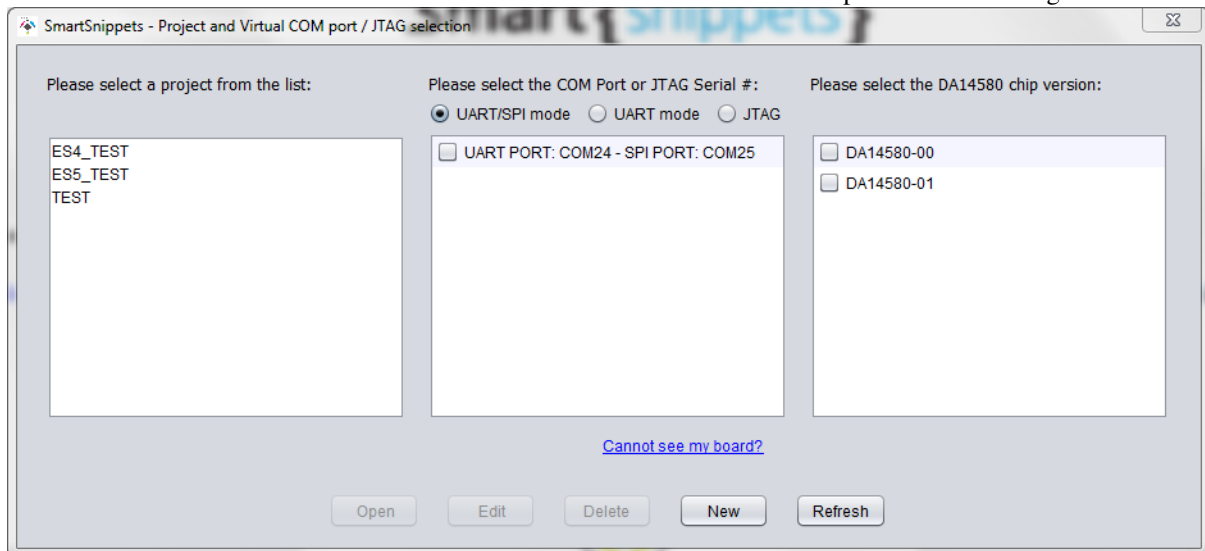
[2] From this point on, application development and debugging is done, and OTP will be programmed

The user might experiment with the Power Profiler for optimising the code as much as possible. This can take several iterations since the user can insert SW cursors (see *Power Profiler software cursors*) to identify the code executions translated into current dissipation. As soon as it is completed, the programming of the code into the

OTP might start so that the system is self-sustained. Upon OTP burning, the system can operate on a battery and realize complete power cycles e.g. active and sleep intervals.

3.3 Project and Virtual COM port / JTAG selection

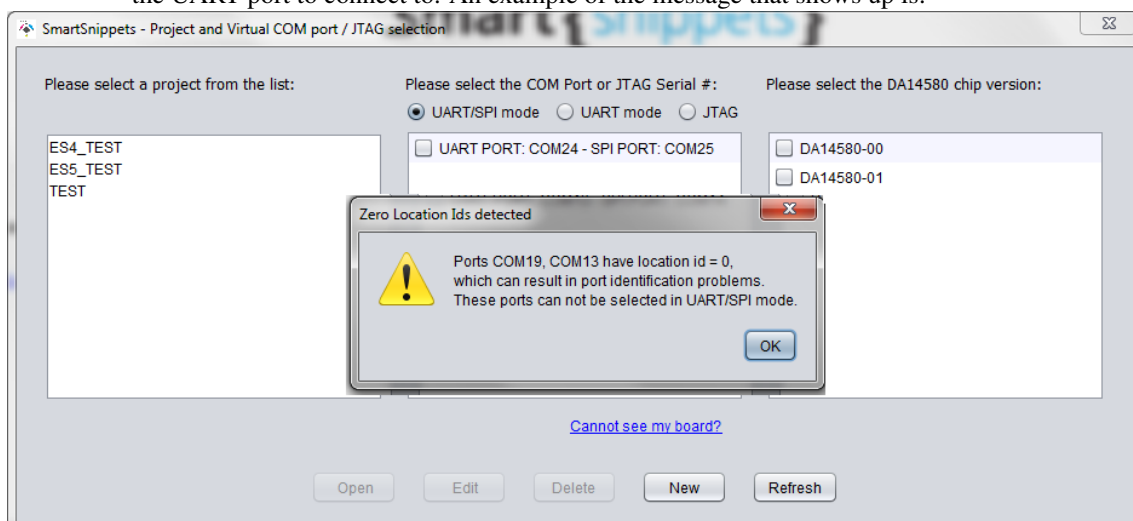
The 'Project and Virtual COM port / JTAG selection screen' allows users to add, edit and delete projects. The following image shows the list of available projects on the left, the list of available virtual COM ports currently connected to FTDI devices in the middle and the list of available DA14580 chip versions on the right:



The following buttons are available:

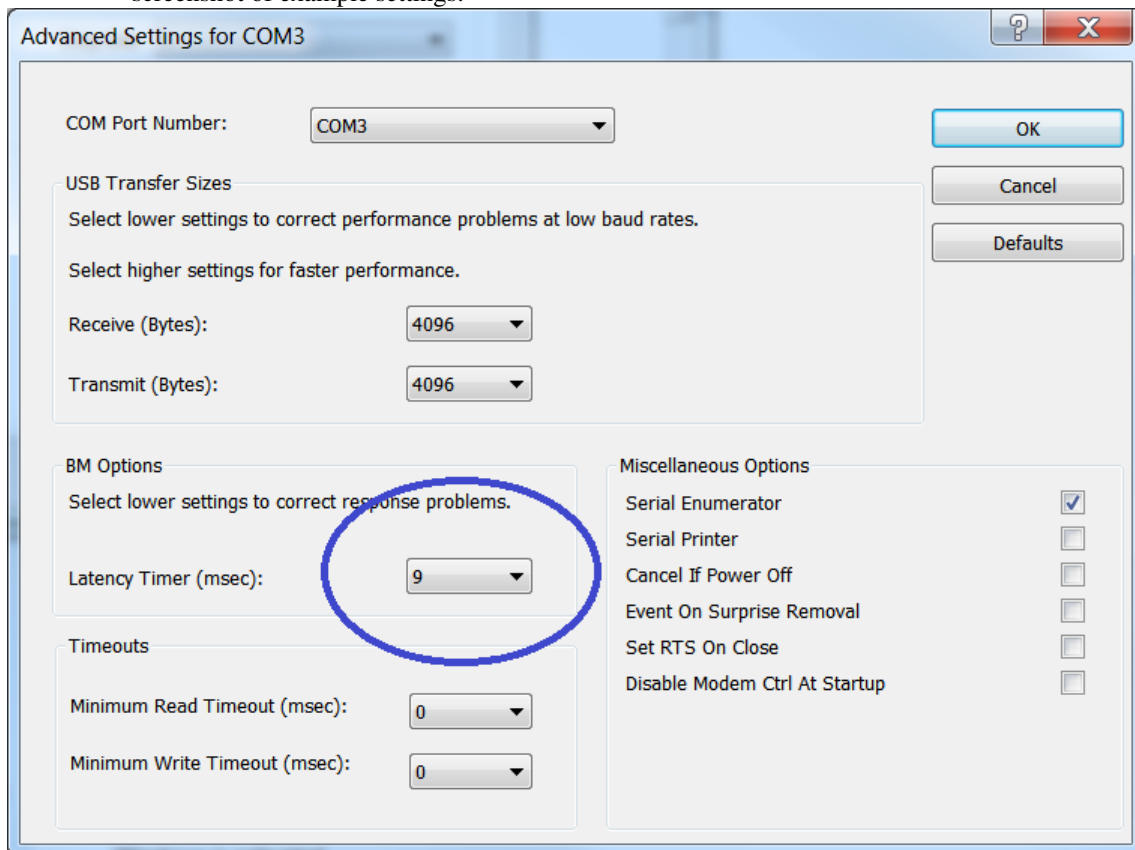
Virtual COM port / JTAG mode: there are three available modes:

1. In **UART/SPI mode**, SmartSnippets scans and lists the Virtual COM port pairs currently connected to FTDI devices. Note that in case that SmartSnippets detects more than one UART ports with location ID = 0 (usually happens when system has USB 3.0 hosts with Windows 7 or older) it can no longer distinguish between these ports in UART/SPI mode. To avoid connecting and working with a port different than the one chosen by the user, the user should switch to UART mode (see below) and select the UART port to connect to. An example of the message that shows up is:



2. In **UART mode**, SmartSnippets scans for individual COM ports without the need to be part of an FTDI pair. In this mode, all ports are treated as UART ports. This means that if the user selects to connect over one of these ports, the SPI functionality (e.g. Power Profiler) is disabled. Please note that the latency time of the FDDI cable in use has to be set to some value below 10ms; otherwise

the connection between SmartSnippets and the board over the COM link will be unstable. Below is a screenshot of example settings:



3. **JTAG mode:** The user has a JTAG attached to the DA14580 DK and wants to communicate over JTAG. He has to select the serial number that corresponds to the attached JTAG.

Open: Used for opening the project currently selected on the left-hand side of the window and connecting through the Virtual COM port pair or single UART port or JTAG selected above. After selecting a project and a virtual COM port / JTAG and pressing 'Open', the project is associated with user's selection and this information is stored in the 'project.sms' file. The next time the user selects the same project, the virtual COM port / JTAG that was used last time will be preselected. The user is allowed to open a project without selecting a virtual COM port / JTAG; in this case all the actions that require communication with the DA14580 DK board are disabled. The user has then the option to change the virtual COM port / JTAG selection by clicking on the toolbar button.

New: When the application launches for the first time, there will be no projects; the user will have to create one by pressing the 'New' button. The project name should be unique and should not contain any spaces or special characters. When saving the new project, a new folder with the name of the project is created under the Projects directory of the SmartSnippets workspace (i.e. under the folder pointed by environment variable %SMARTSNIP-PETS_WORK%). Under this folder, an xml file named 'project.sms' is created to store the information and the user preferences regarding this project.

Edit: By pressing the 'Edit' button, the user can edit the project description.

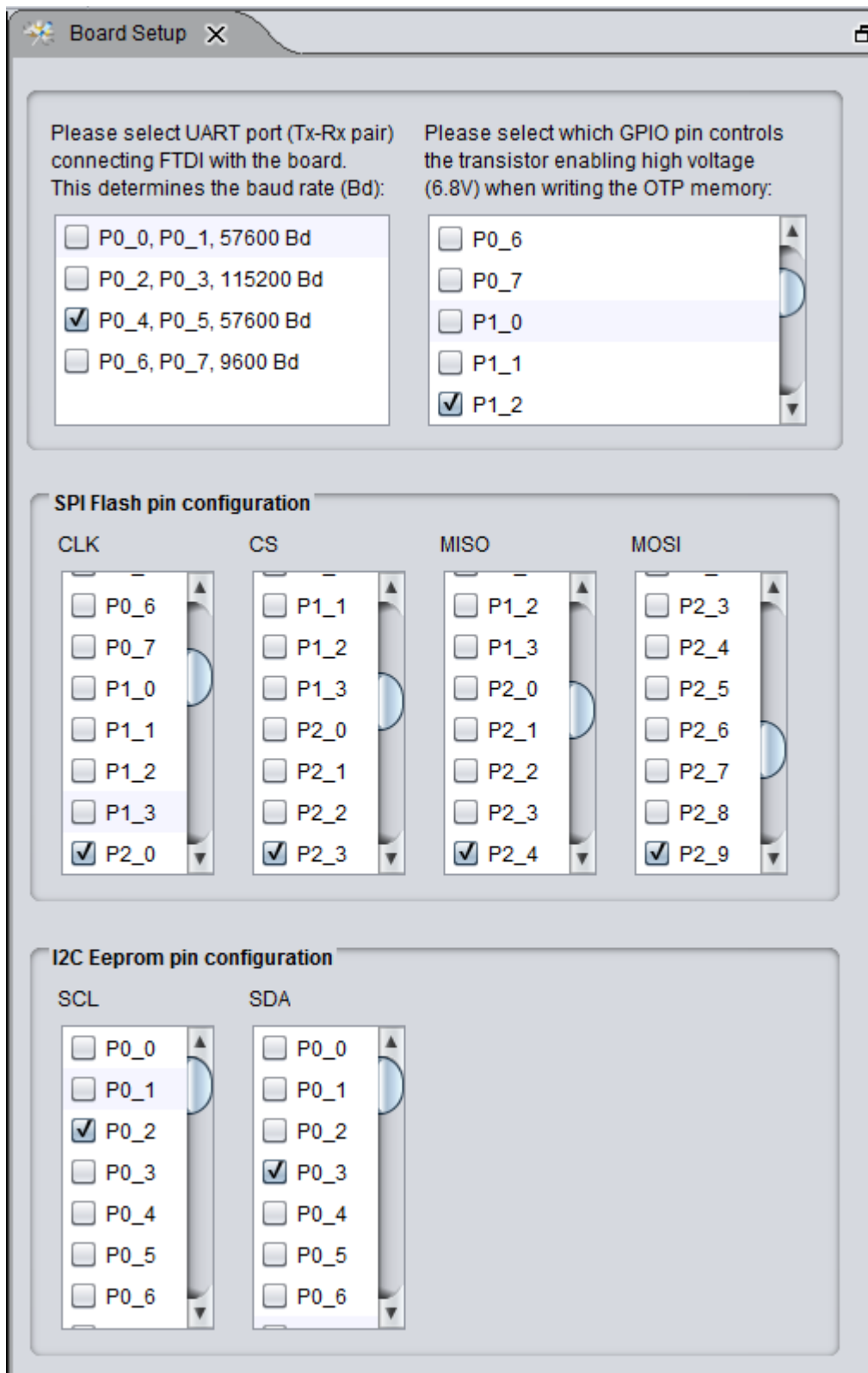
Delete: The 'Delete' button deletes the selected project from the workspace.

Refresh: The user may also delete or add a new project by deleting the corresponding 'Projects' subfolder or by adding a new folder under the 'Projects' folder. The 'Refresh' button can then be used in order to refresh the list of available projects. The 'Refresh' button is also used for refreshing the list of available virtual COM ports. If the board is connected but not listed, the user is advised to use a different USB port, wait a few seconds and press again the 'Refresh' button. Problems in identifying the FTDI device may indicate an invalid installation of the FTDI drivers.

3.4 Board Setup

The 'Board Setup' tool is used for establishing communication with the DK during the boot sequence and therefore should be used before any other tool. The boot procedure is described in the DA14580 datasheet and in application note AN-B-001.

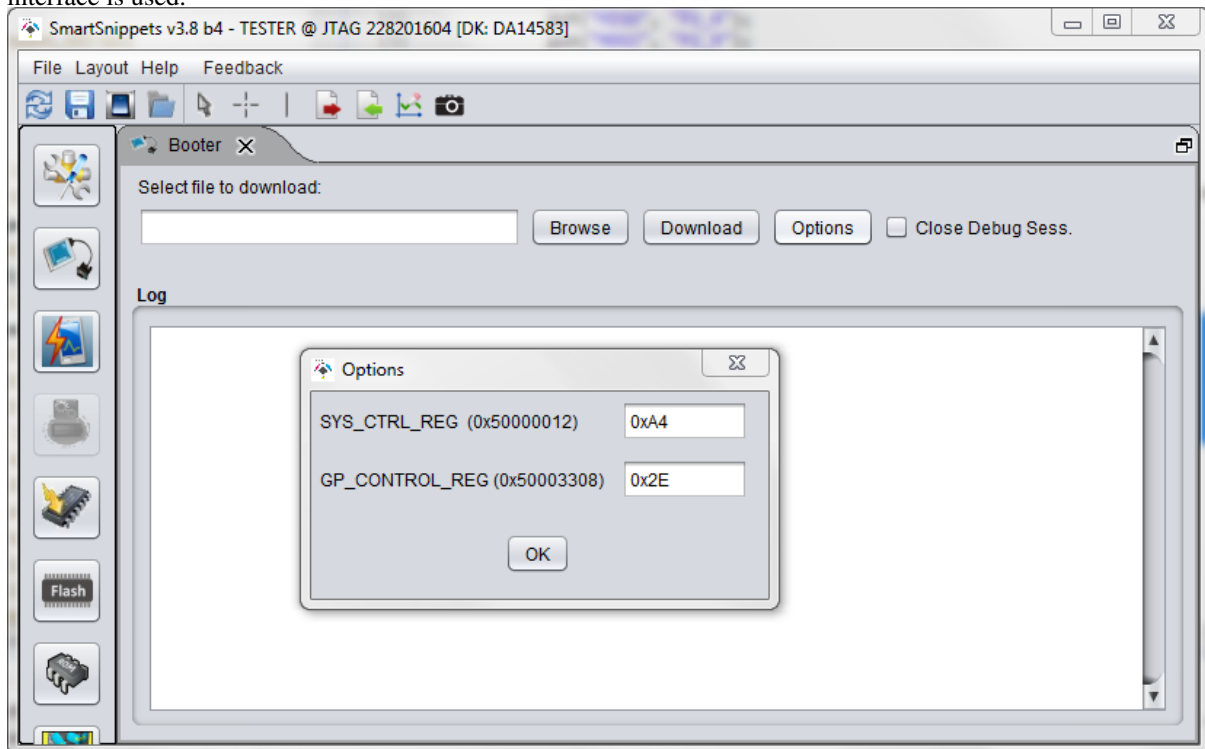
1. The first list is used for selecting the UART port (Tx-Rx pair) connecting the FTDI with the DA14580 chip. The default option is the third one (P0_4, P0_5) and is preselected. The selected port is saved to the 'project.sms' file. The UART port selection also determines the baud rate, which is shown next to the Rx-Tx pair of the UART port.
2. The second list is used for selecting the GPIO pin that controls the transistor enabling high voltage for OTP programming. The default value is P1_2. Similarly to the UART port selection, the selected GPIO pin is saved to the 'project.sms' file and is preselected the next time the user opens the same project. Detailed Tools Description
3. The SPI Flash Pin Configuration Section is used for configuring the gpios of SPI Flash. Note the default gpios differ between DA14583 chips and older chip versions.
4. The EEPROM Pin Configuration Section is used for configuring the gpios of EEPROM. Note the default gpios differ between DA14583 chips and older chip versions.



4 Booter

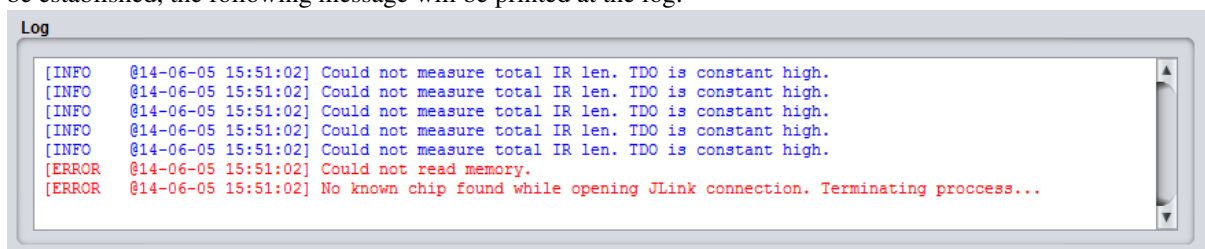
This tool is used for downloading code directly from the Laptop/PC into the main SysRAM of the DA14580 chip and for resetting the chip to execute from there. The purpose is to enable customers and developers quickly and easily update the firmware while testing various configurations or different applications. The 'Options' button

can be used to specify the values for System Control Register and General Purpose Control Register, when JTAG interface is used.



4.1 Downloading Code

By pressing the 'Browse' button, the user is presented with a file browser to select the .hex, .ihex or .bin file to download. When a connection over uart has been selected, after pressing the 'Download' button, the user is prompted to press the reset button of the device via a message at the Log. The application waits for 15 seconds for the reset to be pressed. If reset button is not pressed for 15 seconds, the user will have to press again the 'Download' button and repeat the process. If a 'CRC does not match' shows up, please press the 'Download' button again and then the hardware reset button on the board to restart the download process. When JTAG connection is used, the file is automatically downloaded, without the need to press the reset button. The 'Close Debug Session' option applies only to communication over JTAG and terminates the connection with the DA14580 after downloading the file. The next time the 'Download' button is pressed, it is checked whether the JTAG connection is active or not, and it is reestablished, if needed. If the communication over JTAG with the DA14580 could not be established, the following message will be printed at the log:

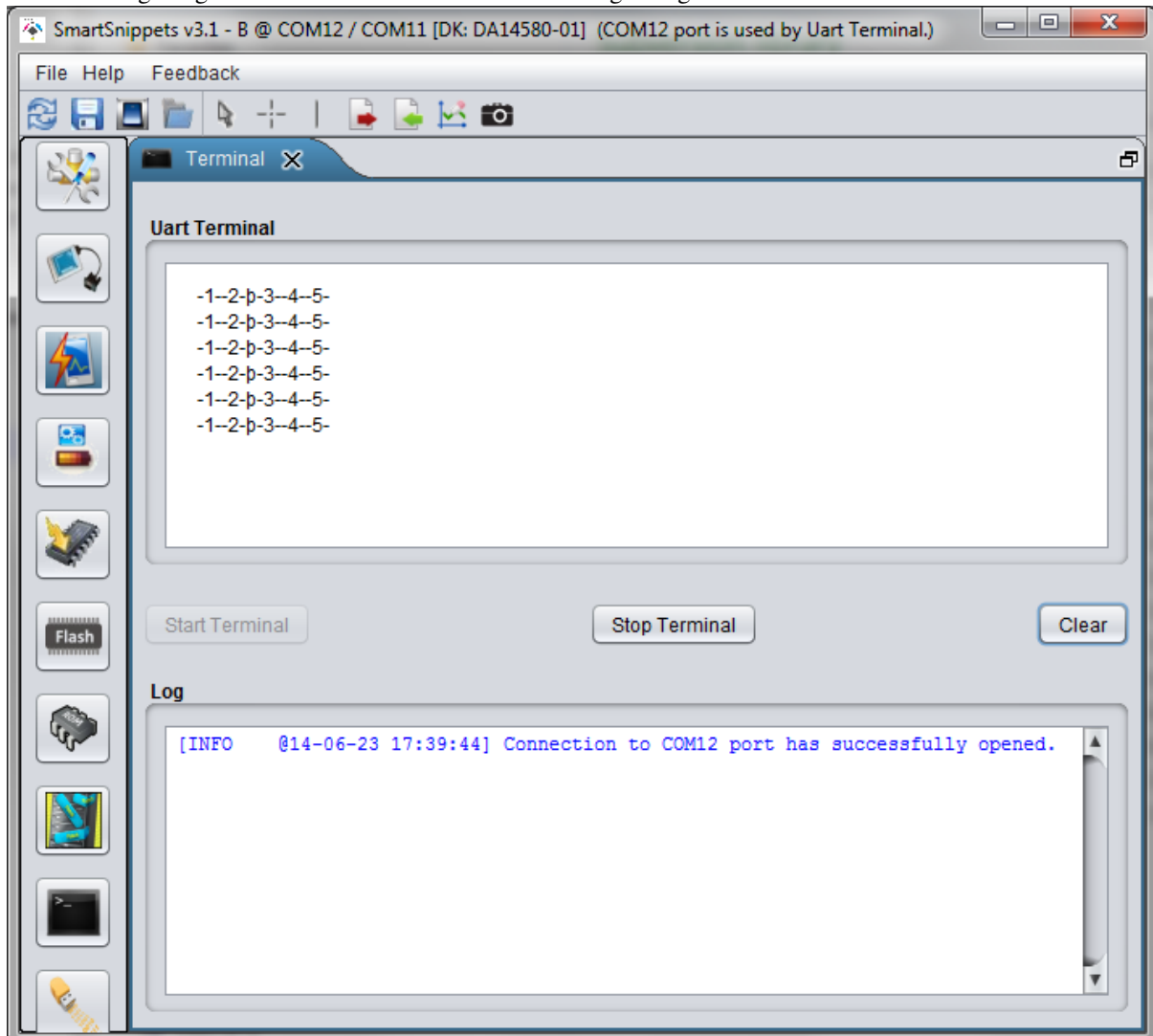


5 UART Terminal

UART Terminal is available only for connection over UART. After successfully downloading the selected file to the DA14580 chip, the 'Start Terminal' button is activated and the user can press it in order to receive data from UART. While the connection to the UART is open, the user cannot open a second UART connection and for this reason the 'Connect' buttons of other tool pages are disabled. The user has to press the 'Stop Terminal' button in

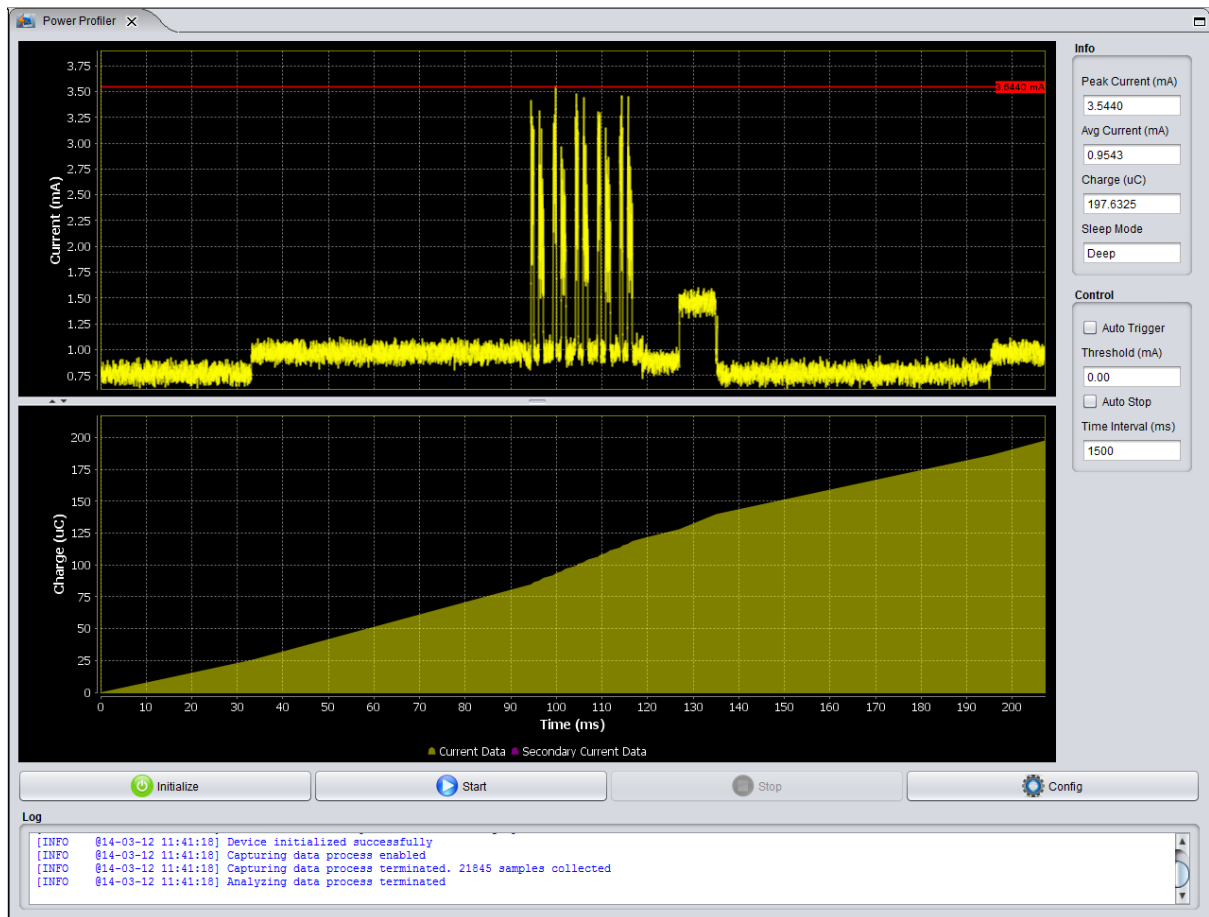
order to close the UART connection. At any point, the user can press the 'Clear' button to clean up the area where data received from the UART is displayed.

The following image shows UART Terminal while receiving debug data from the UART:



6 Power Profiler

The purpose of the 'Power Profiler' is to plot the current (and associated charge) drawn by the battery on the DA14580 DK in real time over USB.



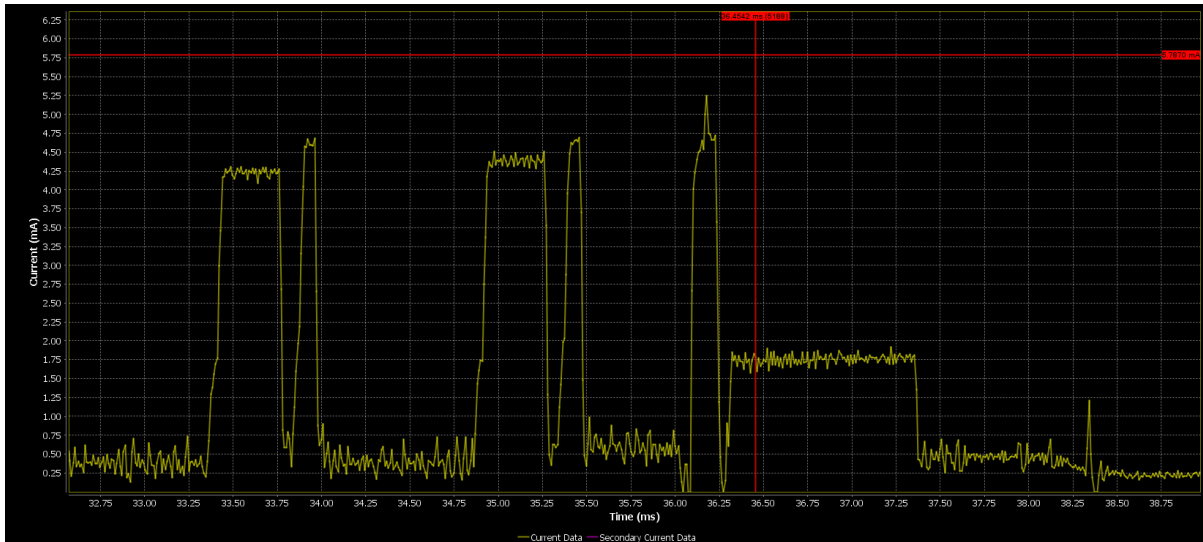
The user can press the ‘Initialize’ button and then the ‘Start’ button in order to start the data capturing process. If auto stop mode is disabled the process will run constantly. In auto stop mode, after collecting a specific number of data samples, the data capturing process is terminated. By pressing the ‘Stop’ button, the user can terminate the process earlier.

The ‘Peak Current (mA)’, the ‘Average Current (mA)’ and the ‘Charge (uC)’ values of the info panel are updated while Power Profiler is capturing data. The info panel also shows the ‘Sleep Mode’ of the current measurement: it can be either in ‘Deep Sleep’ or in ‘Extended Sleep’ mode (the mode is configurable through the configuration page). While in sleep mode, Power Profiler still calculates the power dissipation during this period of inactivity by assuming a 550nA consumption while in Deep Sleep and a 1.2uA consumption while in Extended Sleep mode. This significantly improves the ‘Average Current (mA)’ measurement since it takes into account the power consumption for the entire period of time.

6.1 Power Profiler software cursors

The User has the ability to insert a vertical cursor in the Power Profiler display (SW cursor) by toggling a specific GPIO in the SW running on the ARM Cortex M0 CPU. Instructions on how to implement this are described in detail in the document UM-B-005, Dialog Semiconductor.

The SW cursor provides an accurate correlation between the SW and the Power profile of the system in real time. The vertical line is not movable on the display window.



6.2 Power Profiler control panel

Threshold (mA): If 'Auto Trigger' is checked and start button is pressed, the data capturing process starts when signal exceeds the threshold value.

Time Interval (ms): If 'Auto Stop' is checked and data capturing is in progress, the process stops automatically after 'Time Interval' ms.

6.3 Power Profiler modes

The user can work in the following modes:

Select: Selection mode (standard). In this mode a cursor allows selecting 'Time Marker' objects in the chart. If Del key is pressed the selected 'Time Marker' gets deleted.

Measure: In this mode a cursor allows showing distances of time, current and charge between two points of the chart window. While in this mode, if user presses the left mouse button and moves the cursor, a line connecting these two points is drawn. The distance between the points is highlighted to the right of the second point of the line. It looks like 'dt: [distance of time between two points] ms / dI: [distance of current between two points] mA'. If left mouse is pressed for a second time, the cursor is free to start a new measurement. Cursor can be switched to Select mode by pressing the Esc button.

Add Timemarker: In this mode a cursor allows adding a new Timemarker when left mouse button is pressed.

6.4 Power Profiler actions

The following buttons are available:

- **Export CSV:** Exports the captured current data to csv file
- **Import CSV:** Imports old current data from csv file
- **Clear Secondary Current Data:** Clears the secondary current data and hides it from the chart view
- **Take chart snapshot:** Takes an image snapshot of the Power Profiler chart window in .png format

6.5 Chart management via mouse

- **Timemarker scrolling:** while in **Select** mode, by clicking on any point of a Timemarker with the left mouse, holding the button pressed and then horizontally moving the cursor

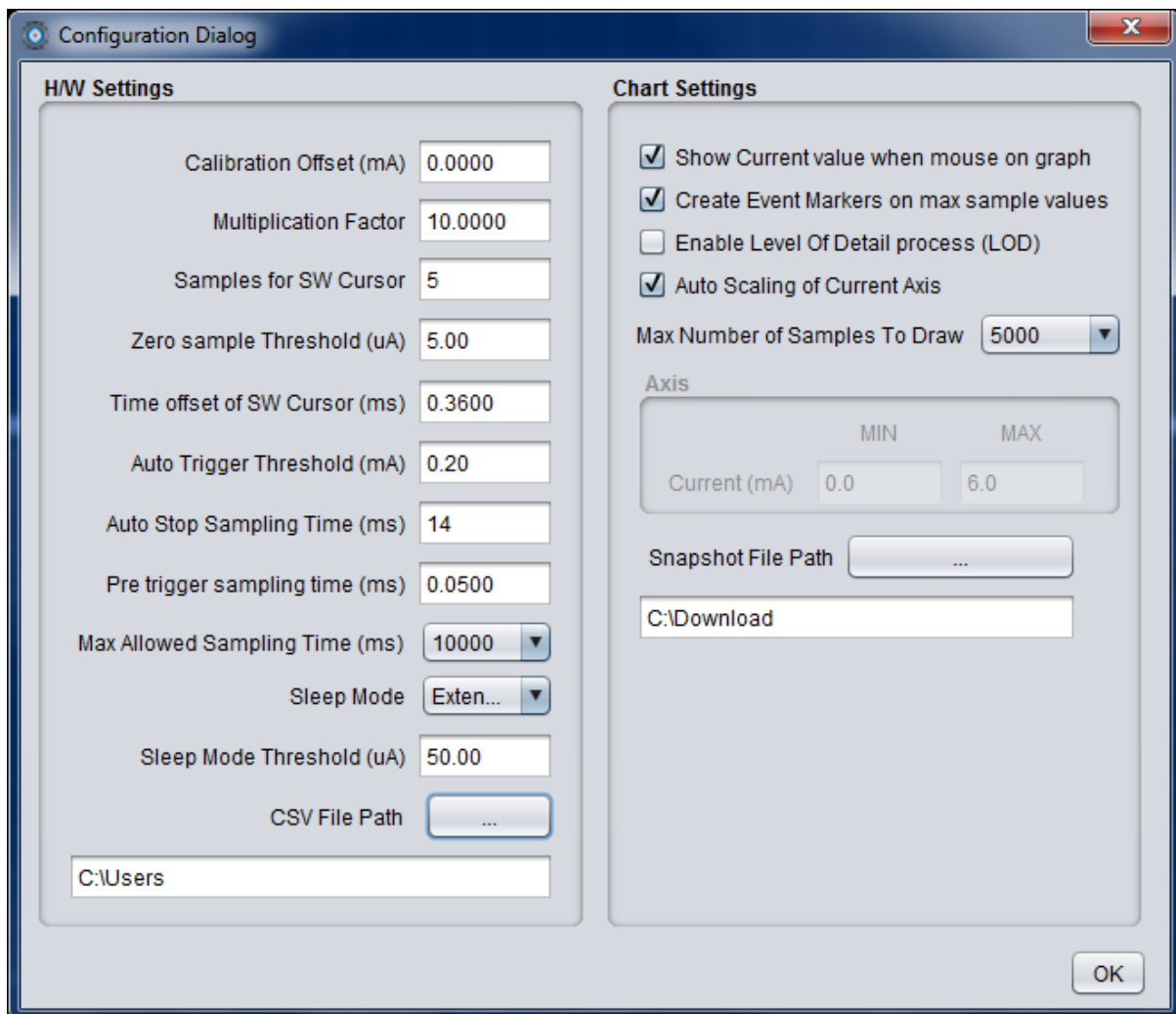
- Chart scrolling: while in **Select** mode, by clicking on any point in the chart window, holding the button pressed and then horizontally moving the cursor
- Zoom In/Out: while in any mode, by moving the mouse wheel over the chart window

6.6 Chart management via keyboard

- Plus(+): zoom in
- Minus(-): zoom out
- Home: shift chart till the last bar
- End: shift chart till the first bar
- Page Up: shift chart in time axis by size of one window backwards
- Page Down: shift chart in time axis by size of one window forward
- Left Cursor: shift chart in time axis by size of 5% window backwards
- Right Cursor: shift chart in time axis by size of 5% window forward
- Del: Delete the selected Time Marker
- F12: Takes an image snapshot of chart window in png format

6.7 Power Profiler configuration

By pressing the 'Config' button, the Power Profiler configuration dialog shows up:



H/W Settings

- **Calibration offset (mA):** Determines the offset value while converting captured values to current (in mA) values
- **Multiplication factor:** Determines the multiplier value while converting captured values to current (in mA) values
- **Samples for SW cursor:** Power Profiler multiplies this number by 3 to determine the number of consecutive zero bytes that mark a S/W cursor. In this case, a S/W cursor appears in the chart.
- **Zero sample Threshold (uA):** The upper threshold under which samples are considered to be equal to zero. Related to S/W cursors and the fact that although we expect to measure 0uA during a S/W cursor, in reality we measure values that are usually lower than 5uA.
- **Time offset of SW Cursor (ms):** Correction time offset of SW cursors.
- **Auto trigger threshold (mA):** Determines the initial value of 'Threshold' textbox on control panel
- **Auto stop sampling time (ms):** Determines the initial value of 'Time Interval' textbox on control panel
- **Pre trigger sampling time (ms):** The amount of captured data (in ms), just before the start of data capturing process
- **Max allowed sampling time (ms):** The maximum allowed sampling time during the data capturing process
- **Sleep Mode:** Determines whether the measurement is done while in Extended or Deep sleep mode
- **Sleep Mode Threshold (uA):** The upper threshold under which measured values are considered equal to the standard current dissipation for that mode (1.2uA if in 'Deep Sleep' and 0.6uA if in 'Extended Sleep')

mode)

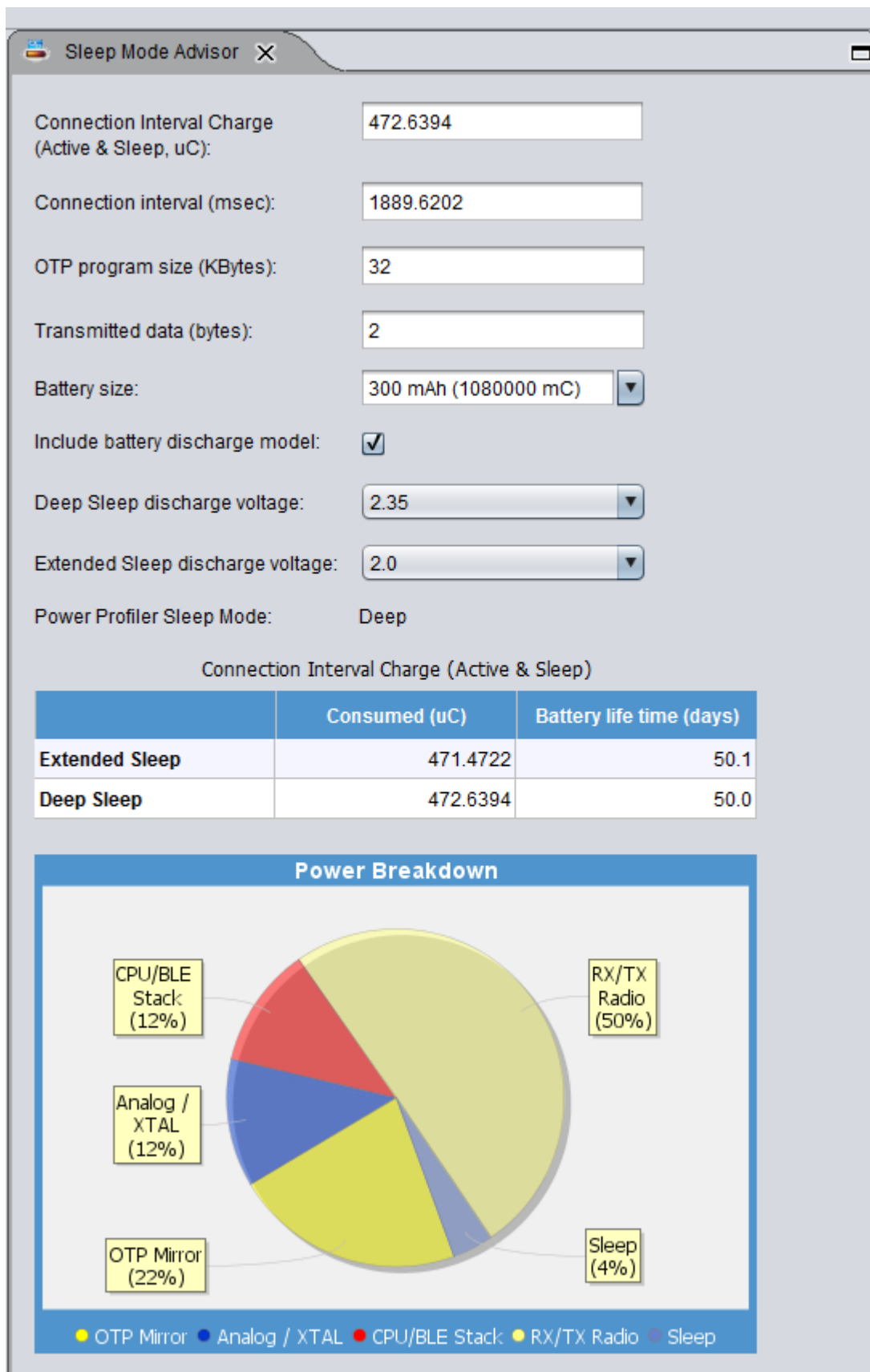
- **CSV File path:** Export/Import destination for csv files

Chart Settings

- **Show current value when mouse on graph:** If checked, a tooltip shows the time and current value when the mouse is over the waveform
- **Create event markers on max sample values:** If checked, a marker is created on maximum current value
- **Enable Level Of Detail process (LOD):** If checked, Power Profiler will compute a multilevel approximation of waveform.
- **Auto scaling of Current/Time axis:** If checked, the current axis range is determined by the minimum and maximum values of the waveform. If not checked, the current axis range is determined by the minimum and maximum values that the user provides under the 'Axis' table
- **Max Number of Samples To Draw:** it allows the user control the plot quality by determining the maximum number of samples to draw during downsampling
- **Snapshot File Path:** Save destination folder for snapshot .png files.

7 Sleep Mode Advisor

The purpose of this tool is to help users understand how much power their application dissipates in Deep Sleep and Extended Sleep modes and what is its impact in battery lifetime duration. Sleep Mode Advisor depends on the samples collected through the Power Profiler in order to estimate parameters like the consumed charge and the battery life time. If no data samples have been collected in Power Profiler, the Sleep Mode Advisor cannot be used. The user can configure parameters that affect the estimations, such as the battery size, the OTP program size mode, etc., according to the exact use case. The following picture illustrates an example configuration for Sleep Mode Advisor:



In this example configuration, we can see that Extended Sleep mode is preferred, since the battery lasts for 50.1 days, compared to the 50.0 days that have been estimated for the Deep Sleep mode.

The power breakdown chart illustrates the percentage of power that has been consumed in each one of the follow-

ing groups:

1. Analog/XTAL
2. CPU/BLE Stack
3. RX/TX Radio
4. OTP Mirror
5. Sleep

Each field of Sleep Mode Advisor is explained in the following section:

Connection Interval Charge (Active & Sleep, uC): Average charge per connection interval. Each time Power Profiler is executed and collects a number of samples, this value is recalculated in Power Profiler and updated in Sleep Mode Advisor.

Connection Interval: The average duration between two connection events. This value is also estimated each time Power Profiler is executed.

OTP program size (in Kbytes): This parameter specifies the size in Kbytes of the program downloaded to DA14580 through the OTP Image tab. If parameter DMA length has been specified in OTP Header, OTP program size will contain the DMA length equivalent value in bytes. The user can also edit the 'OTP program size' text field in order to overwrite its value. Accepted values are decimal numbers between 0.00 and 32.00.

Transmitted Data (in bytes): The number of bytes transmitted. Can be an integer number in the between 1 and 23.

Battery size: User can select from the drop down list the size of the battery used by DA14580 DK. The user may also add additional battery sizes in the following way: first he should click with the mouse in the battery size text field, then insert an integer number which indicates the battery's energy storage capacity in mAh and finally click somewhere else in the GUI. SmartSnippets adds the new battery size in the drop-down list and recalculates the battery life time over deep and extended sleep modes according to the new battery size.

Include battery discharge model: If this checkbox is selected, battery discharge will be taken into account in the calculations. If the maximum battery life time (as estimated by the battery discharge specifications for CR2032) is shorter than the calculated battery life time (as estimated taking into account the selected battery size and power consumption), the maximum life time (instead of the calculated value) will be shown in the power consumption table.

Deep Sleep discharge voltage: Determines the voltage threshold at which the battery is completely discharged when the DA14580 is operating in deep sleep mode. Default value is 2.35V.

Extended Sleep discharge voltage: Determines the voltage threshold at which the battery is completely discharged when the DA14580 is operating in extended sleep mode. Default value is 2.00V.

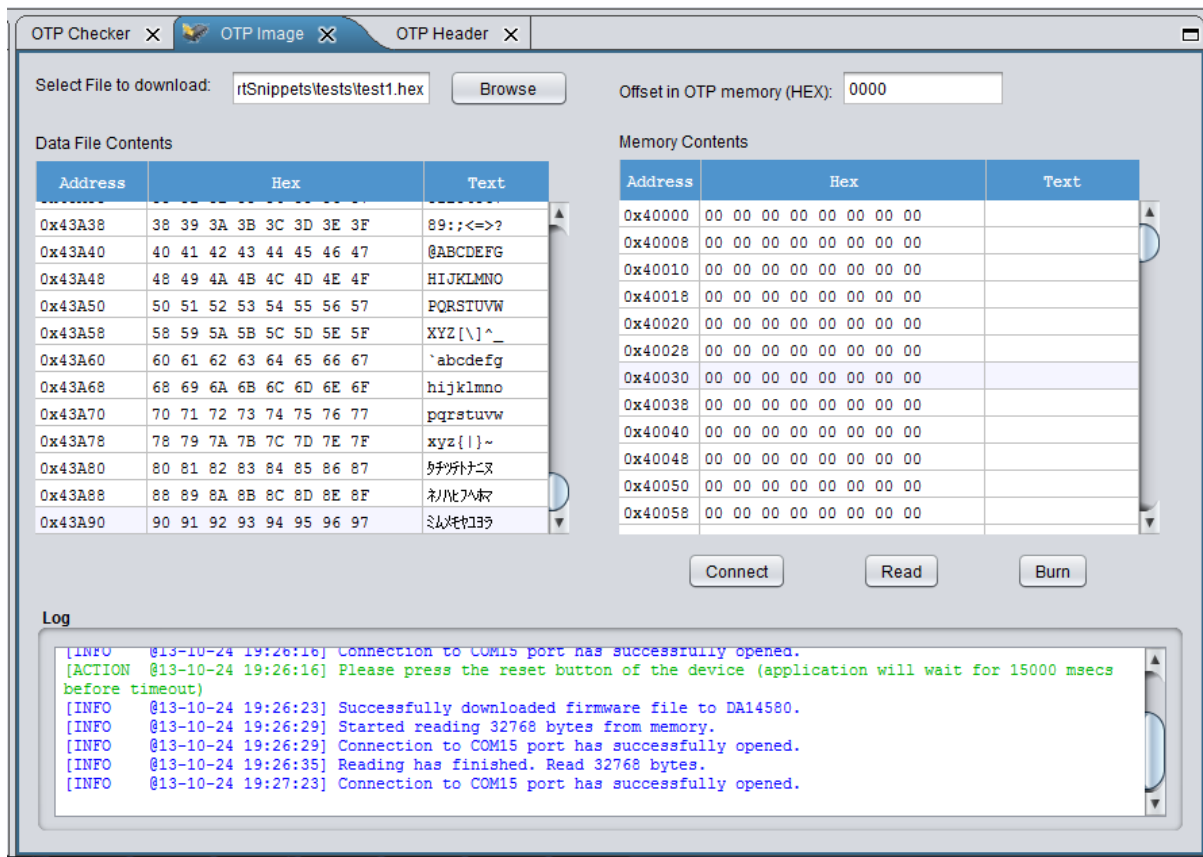
Power Profiler Sleep Mode: It is used for letting the user know whether the samples collected in Power Profiler have been collected while in Deep Sleep mode or Extended Sleep mode.

8 OTP Programmer

The 'OTP Programmer' tool is used for burning the OTP Memory and OTP Header. By visualizing and testing the OTP image while still under development, it helps increasing productivity and avoiding fatal mistakes of erroneously programming the OTP header flags. 'OTP Programmer' consists of two sub tools: the 'OTP Image' and the 'OTP Header'.

8.1 OTP Image

This OTP tool enables downloading the default firmware into the SysRAM and burning the OTP memory with a user-defined .hex/.ihex/.bin file. The following picture shows the OTP Image tab:



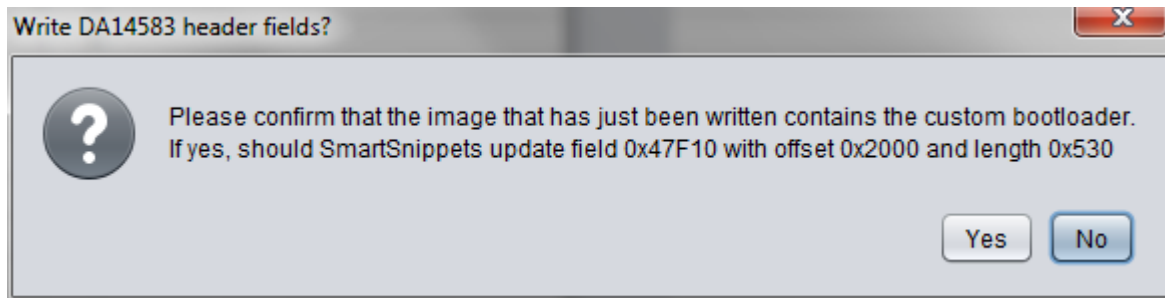
On the left side of the OTP Image tab, the user is able to select the Intel hex or binary file to be downloaded to the OTP Memory. If an Intel hex file is selected, it is parsed and its contents are presented in the Data File Contents table. If a binary file is selected, the table remains empty, but a log message indicates that the file has been read along with its size in bytes.

On the right side of the OTP Image tab, the OTP Memory contents are shown. There are 3 actions associated with it:

Connect: As a first step before reading memory contents and burning the OTP memory, the user has to establish connectivity with the DA14580 DK by pressing the ‘Connect’ button and waiting for the ‘Press the reset button’ action message to appear on the log. Similarly to UART Booter, the user has 15 seconds to press the reset button to download the default firmware file to the chip. If a ‘CRC does not match’ shows up, please press the ‘Connect’ button again and then the hardware reset button on the board to restart the download process.

Read: After successfully downloading the firmware file to the DA14580 chip, the user can press the ‘Read’ button to read the OTP memory.

Burn: If a file has been selected for downloading, the user can press the ‘Burn’ button to burn the OTP memory with it. Before performing the actual burn action, SmartSnippets checks if the memory segment the user is attempting to burn already contains data and notifies the user accordingly; it is up to the user to decide whether to proceed with the burning or not. After a burn action, a read action is automatically performed in order to refresh the memory contents with the new data. When a DA14583 chip is selected, the user has the option to automatically burn “Advanced Bootloader Offset and Length” Header field with the offset and length of the custom bootloader burned in OTP via the OTP Image. A pop-up dialog requests for permission to burn the header field:



The 'Offset in OTP Memory' field allows the user to enter the offset (in hex number of bytes) from which a read or burn action starts.

The One Time Programmable cells in the DA14580 are un-programmed when containing a logic 0 level. After programming they will become logic 1.

8.2 OTP Header

The 'OTP Header' tool is used for burning the OTP header. Before burning, it validates every header field to ensure that the OTP header is correctly programmed, avoiding in this way fatal mistakes which could damage the DA14580 chip.

The screenshot shows the 'OTP NVDS [DK: DA14580-01]' application window. It features a table with four columns: Address, Parameter, Description, and Value. The table lists various OTP parameters, including Application Flags, IQ Trim, and Customer Specific Fields. Below the table are buttons for 'Connect', 'Import Header from file', 'Read from memory', 'Burn', and 'Export Header to file'. At the bottom, there is a 'Log' window displaying system messages.

Address	Parameter	Description	Value
0x47F00	Application Flag 1	0x00000000: Empty OTP, 0x1234A5A5: Application Burned	No
0x47F04	Application Flag 2	0x00000000: Empty OTP, 0x1234A5A5: Application Burned	No
0x47F08	IQ_Trim	Bits[31:16]=RF_MIXER_CTRL1_REG ,Bits[15:0]=BIAS_CTRL1_REG	11358888
0x47F0C	Reserved	Free for future use	11121314
0x47F10	Reserved	Free for future use	ABCDEF12
0x47F14	Reserved	Free for future use	00000000
0x47F18	Reserved	Free for future use	00000000
0x47F1C	Reserved	Free for future use	00000000
0x47F20	Reserved	Free for future use	00000000
0x47F24	Reserved	Free for future use	00000000
0x47F28	Reserved	Free for future use	00000000
0x47F2C	Reserved	Free for future use	00000000
0x47F30	Reserved	Free for future use	00000000
0x47F34	Reserved	Free for future use	00000000
0x47F38	Reserved	Free for future use	00000000
0x47F3C	Reserved	Free for future use	00000000
0x47F40	Reserved	Free for future use	00000000
0x47F44	Reserved	Free for future use	00000000
0x47F48	Reserved	Free for future use	00000000
0x47F4C	Reserved	Free for future use	00000000
0x47F50	Reserved	Free for future use	00000000
0x47F54	Customer Specific Fields		33445566
0x47F58	Customer Specific Fields		00000000
0x47F5C	Customer Specific Fields		00000000

```

Log
[INFO @14-02-17 11:38:44] Firmware File C:\Users\artpap\SmartSnippets\resources\programmer_ES5.bin has been selected
[INFO @14-02-17 11:38:45] Connection to COM19 port has successfully opened.
[ACTION @14-02-17 11:38:45] Please press the reset button of the device (application will wait for 15000 msec before timeout)
[INFO @14-02-17 11:38:48] Successfully downloaded firmware file to DA14580.
[INFO @14-02-17 11:38:48] Successfully disconnected from port COM19.
[INFO @14-02-17 11:38:48] Started reading 256 bytes from address 47F00.
[INFO @14-02-17 11:38:48] Connection to COM19 port has successfully opened.

```

This tool displays the header table, which allows the user view and edit the value of each header field. The values should be hexadecimal values with size equal to the one shown at the 'Size (words)' column of the table. Note that one word is represented with 8 hexadecimal digits. There are two types of fields:

1. 'Integer': field is treated as an integer, which means that if the user enters fewer hex values than expected according to field size the value is patched with leading zeroes before burning it to the OTP memory (e.g. '14580' becomes '00014580' for a 1-word field). For 'integer'-type fields, the least significant byte of a word is stored in the smallest address (little-endian). E.g. if a user types 0A0B0C0D for field 'DMA Length', 0x0A will be written at 0x47FFB and 0x0D will be written at 0x47FF8.
2. 'String': field is treated as a string, which means that if the user enters fewer hex values than expected according to field size the value is patched with trailing zeroes before burning it to the OTP memory (e.g. '14580' becomes '14580000' for a 1-word field). For 'string'-type fields, the most significant (left-most) byte of a word is stored in the smallest address (big-endian). E.g. if a user types 0A0B0C0D for field 'Device Unique ID', 0x0A will be written at 0x47FD4 and 0x0D will be written at 0x47FD7.

Most of the fields that contain combo boxes cannot be programmed more than once. For example if 'RC32KHz' (hex value 0xAA) has been burned as the 32KHz source, it is not allowed to overwrite it with the 'XTAL32KHz' value (hex value 0x00). In such a case, the combo boxes are disabled to avoid confusion. The following actions are available:

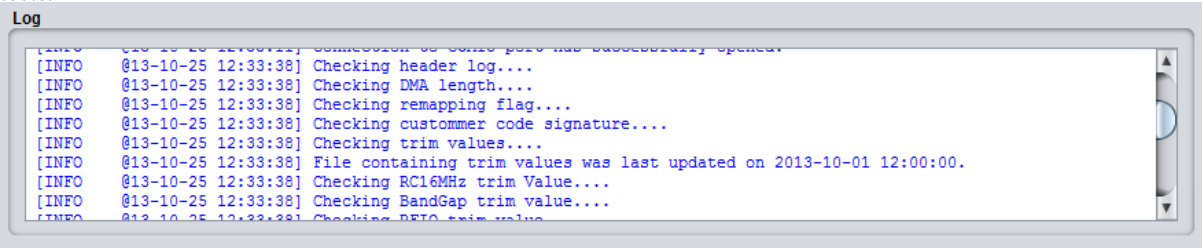
Connect: The user has to establish the connection with the DA14580 DK if no previous connection has been

established via the 'OTP Image' tool. Until the connection is established the 'Read from memory' and 'Burn' buttons are disabled.

Import Header from file: Used for selecting a file in Intel hex (.hex) or binary (.bin) format, containing the OTP header, and importing it into the header table for editing and burning. The imported header file is advised to be a file that has been generated with the export header to file button of the OTP header. When importing a file in OTP header, it is validated that the file contains the expected number of parameters (22) and that each parameter has the correct length, as it is indicated by the second column of the header table. If the validation tests fail, the file cannot be imported in OTP header, since it is risky to import a file containing wrong header data and burn it in OTP header.

Read from memory: After successfully downloading the firmware file to the DA14580 chip, the user can press the 'Read from memory' button to read the current contents of the header in the memory.

Burn: The user can press the 'Burn' button to burn the OTP header with the current contents of the header table. Before performing the burn action, a set of validation tests is executed in order to ensure the correctness of each header field. The following image is an example of the messages printed to the log during the header validation tests.



```
Log
[INFO @13-10-25 12:33:38] Connection to COM3 port has successfully opened.
[INFO @13-10-25 12:33:38] Checking header log....
[INFO @13-10-25 12:33:38] Checking DMA length....
[INFO @13-10-25 12:33:38] Checking remapping flag....
[INFO @13-10-25 12:33:38] Checking customer code signature....
[INFO @13-10-25 12:33:38] Checking trim values....
[INFO @13-10-25 12:33:38] File containing trim values was last updated on 2013-10-01 12:00:00.
[INFO @13-10-25 12:33:38] Checking R16MHz trim Value....
[INFO @13-10-25 12:33:38] Checking BandGap trim value....
[INFO @13-10-25 12:33:38] Checking BFO trim value....
```

When a validation test fails or the user is about to make an important change to the header contents, a popup dialog notifies the user accordingly. For each dialog, the user has the option to either stop the burning process or ignore it and continue with the validation checks. The following validation tests are currently performed:

1. Last burn validation: The existence of 'header_log.txt' file is checked. If found, a message informs the user when the memory was burned for the last time.
2. DMA length validation: 'DMA length' value should always be smaller than 32768 bytes. It is automatically set to the maximum-allowed value if it is greater or equal to 32768 bytes and the user ignores the DMA length check popup dialog. Moreover, if a file has been loaded on OTP Image, it is checked that the DMA length is higher than the number of data bytes in that file.
3. Remapping flag selection: Displays an informative message if the 'remapping flag' value has changed to 0.
4. Customer Code Signature validation: If the 'signature algorithm' field has been set and a file has been selected for downloading to the OTP Image, it calculates a hash of the image file code. The calculated value should match to the value of the field Customer Code Signature.
5. Trim values validation: The fields at addresses 0x47f7C to 0x47f90 are validated against the latest trim values provided by Dialog. The latest trim values are included in file 'trimValues.txt' located under the %SMARTSNIPPETS_WORK%\resources folder. The file also includes a timestamp indicating the last time the trim values were updated. If the value entered by the user at a trim value field does not match the respective trim value at trimValues.txt file, the user is notified accordingly.
6. Calibration flag validation: The 'calibration flag' should be in accordance with the trim values that have been set. The description field of the calibration flag indicates which bit corresponds to which trim value. If a trim value has been set and the corresponding bit of the calibration flag has not been set or vice versa, the user will be notified accordingly.
7. 32KHz source selection: Displays a message informing the user about the selected 32KHz source.
8. Package selection: Displays a message informing the user about the selected package.
9. Header written already validation: Before performing the actual burn action, SmartSnippets checks if the memory segment the user is attempting to burn already contains data and notifies the user accordingly; it is up to the user to decide whether to proceed with the burning or not.

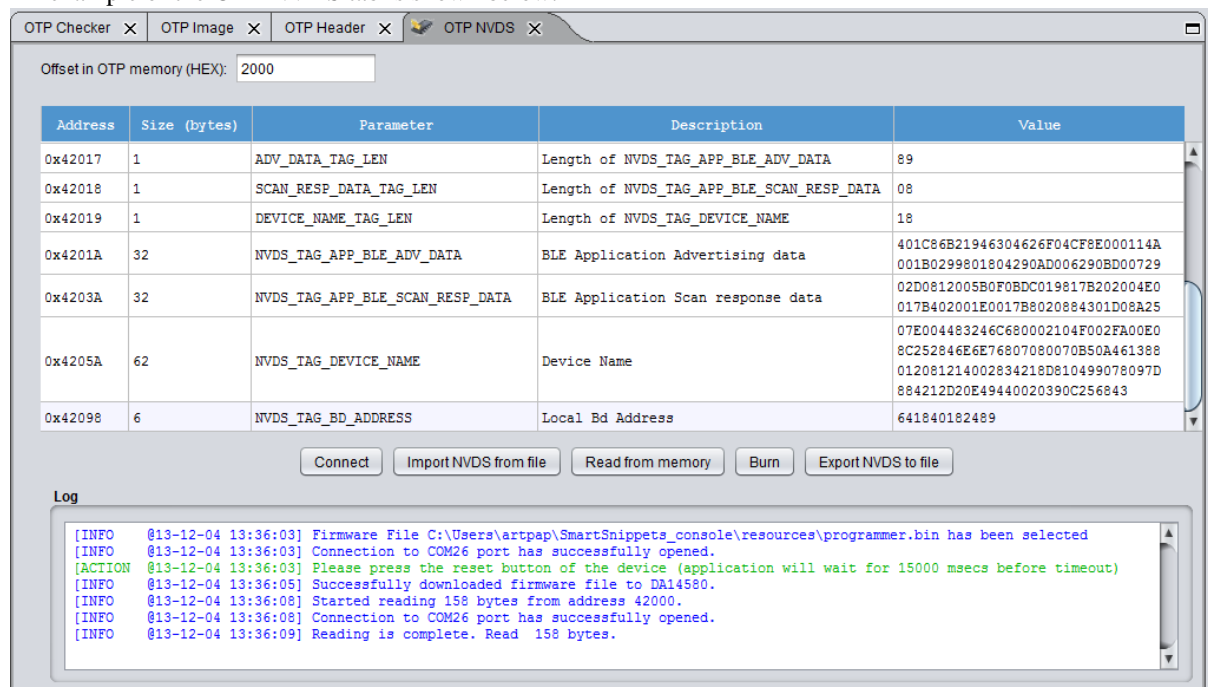
After a burn action, a read action is automatically performed in order to refresh the memory contents with the new data. Also, the entire OTP Header section is appended (together with a timestamp) to 'header_log.txt' file located under the project working directory for future reference.

Export header to file: Used for exporting the header to Intel hex (.hex) or binary (.bin) file.

8.3 OTP NVDS

The functionality of OTP NVDS tool is very similar to the functionality of the OTP Header and allows the user to read and write the NVDS memory block. By editing the offset text field, the user can change the address at which a burn and a read operation will be performed. The offset (in bytes) must be a hex number between 0x0000 and 0x8000. For burn operations, the offset should be such so that the address where the last NVDS data byte will be written is smaller than the address where OTP header starts.

An example of the OTP NVDS tab is shown below:

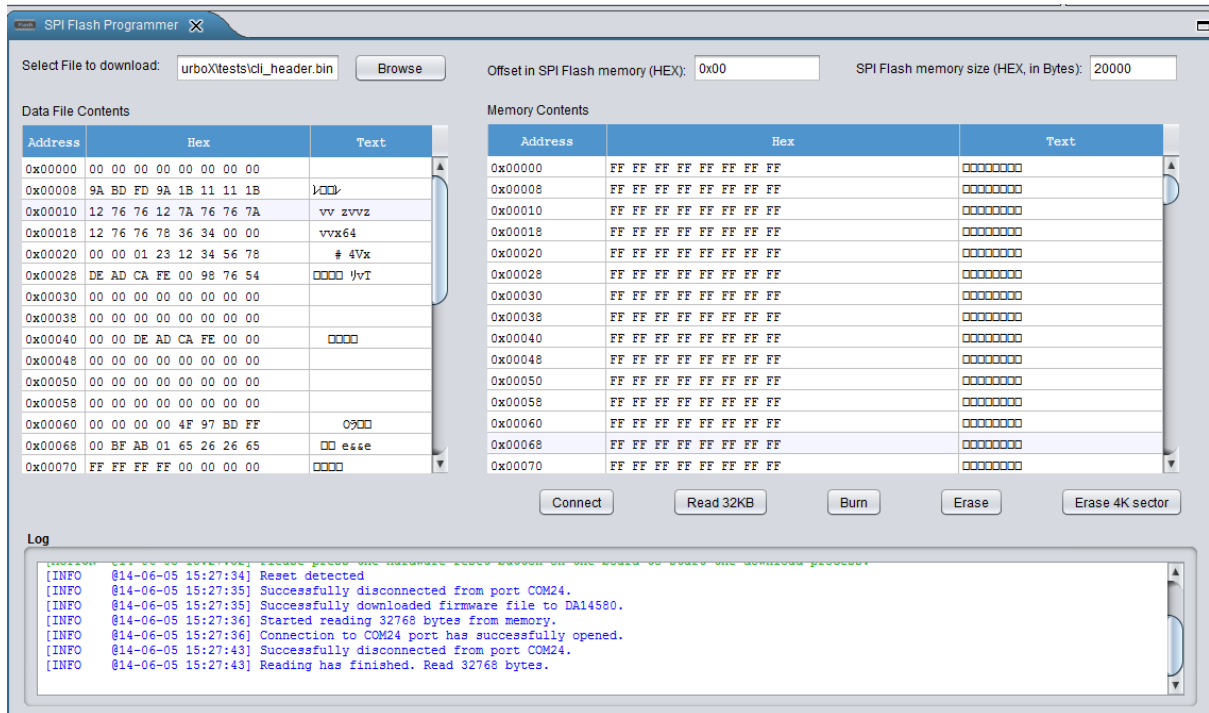


When the burn button is pressed, it is validated that the length of parameters NVDS_TAG_APP_BLE_ADV_DATA, NVDS_TAG_APP_BLE_SCAN_RESP_DATA and NVDS_TAG_DEVICE_NAME is equal to the value indicated by parameters ADV_DATA_TAG_LEN, SCAN_RESP_DATA_TAG_LEN and DEVICE_NAME_TAG_LEN respectively. The user is notified if the validation test fails, and has the option to proceed with the burn action or cancel it. Similar to OTP Header, there are integer-type and string-type fields. For OTP NVDS, the following fields are treated as strings:

- NVDS_TAG_APP_BLE_ADV_DATA
- NVDS_TAG_APP_BLE_SCAN_RESP_DATA
- NVDS_TAG_DEVICE_NAME
- NVDS_TAG_BD_ADDRESS

9 SPI Flash Programmer

SPI Flash Programmer is used for downloading an image file to the DA14580 SPI Flash Memory. Its functionality is similar to the functionality of the OTP Image tab.



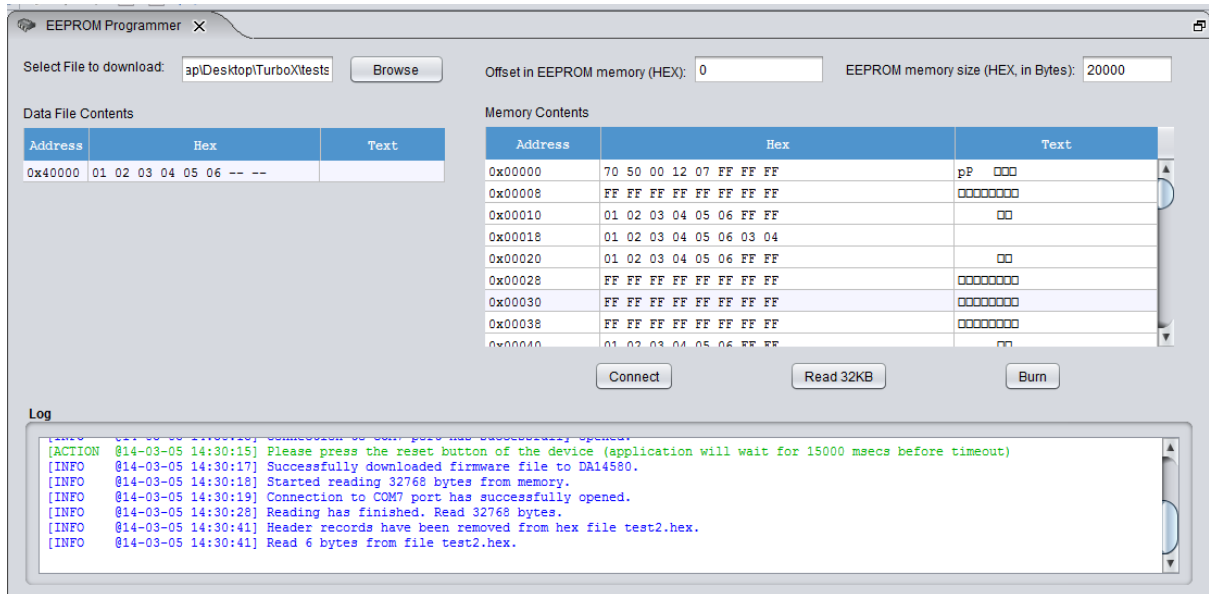
The user can select a .hex or binary image file in order to burn it to the SPI Flash Memory. The following actions are available:

- **Connect:** Special firmware is downloaded to the chip to allow the user interact with the SPI Flash memory. This is a mandatory step before enabling the other actions. Please note that this firmware is different from the firmware downloaded when pressing the 'Connect' button on the OTP Image tab. If a 'CRC does not match' shows up, please press the 'Connect' button again and then the hardware reset button on the board to restart the download process.
- **Read 32KB:** The SPI Flash memory is always read in 32KByte blocks starting from the specified offset. This offset cannot be greater than the maximum SPI Flash Memory size. Since the application does not know the SPI Flash memory size, the user can specify it by filling the 'SPI Flash Memory size (HEX, in bytes)' field. If the user does not know the size of the SPI Flash memory, a default value of 128KB is used.
- **Burn:** When trying to burn data at offset *0x00000* of the SPI Flash Memory, user is presented with the option to make it bootable. When the *bootable* option is selected, a special header is added before the data and the data is written starting from address *0x00008*. Please note that before downloading data to the SPI Flash, the firmware erases the appropriate 4KB sectors in the area that the data is about to be written.
- **Erase:** Erases the entire SPI Flash Memory
- **Erase 4K Sector:** Erases one sector of size 4K bytes from the SPI Flash Memory, starting from the offset entered by the user at the 'Offset in SPI Flash memory (HEX)' text field.

Please note that currently SPI Flash Programmer has been tested in BUCK mode at 3V and BOOST mode at 2.7V.

10 EEPROM Programmer

EEPROM Programmer is used for downloading an image file to the DA14580 EEPROM Memory. Its functionality is similar to the functionality of the OTP Image tab.



The user can select a .hex or binary image file in order to burn it to the EEPROM Memory. The following actions are available:

- **Connect:** Special firmware is downloaded to the chip to allow the user interact with the EEPROM memory. This is a mandatory step before enabling the other actions. Please note that this firmware is different from the firmware downloaded when pressing the ‘Connect’ button on the OTP Image tab. If a ‘CRC does not match’ shows up, please press the ‘Connect’ button again and then the hardware reset button on the board to restart the download process.
- **Read 32KB:** The EEPROM memory is always read in 32KByte blocks starting from the specified offset. This offset cannot be greater than the maximum EEPROM Memory size. Since the application does not know the EEPROM memory size, the user can specify it by filling the ‘EEPROM Memory size (HEX, in bytes)’ field. If the user does not know the size of the EEPROM memory, a default value of 128KB is used.
- **Burn:** When trying to burn data at offset *0x00000* of the EEPROM Memory, user is presented with the option to make it bootable. When the *bootable* option is selected, a special header is added before the data and the data is written starting from address *0x00020*.

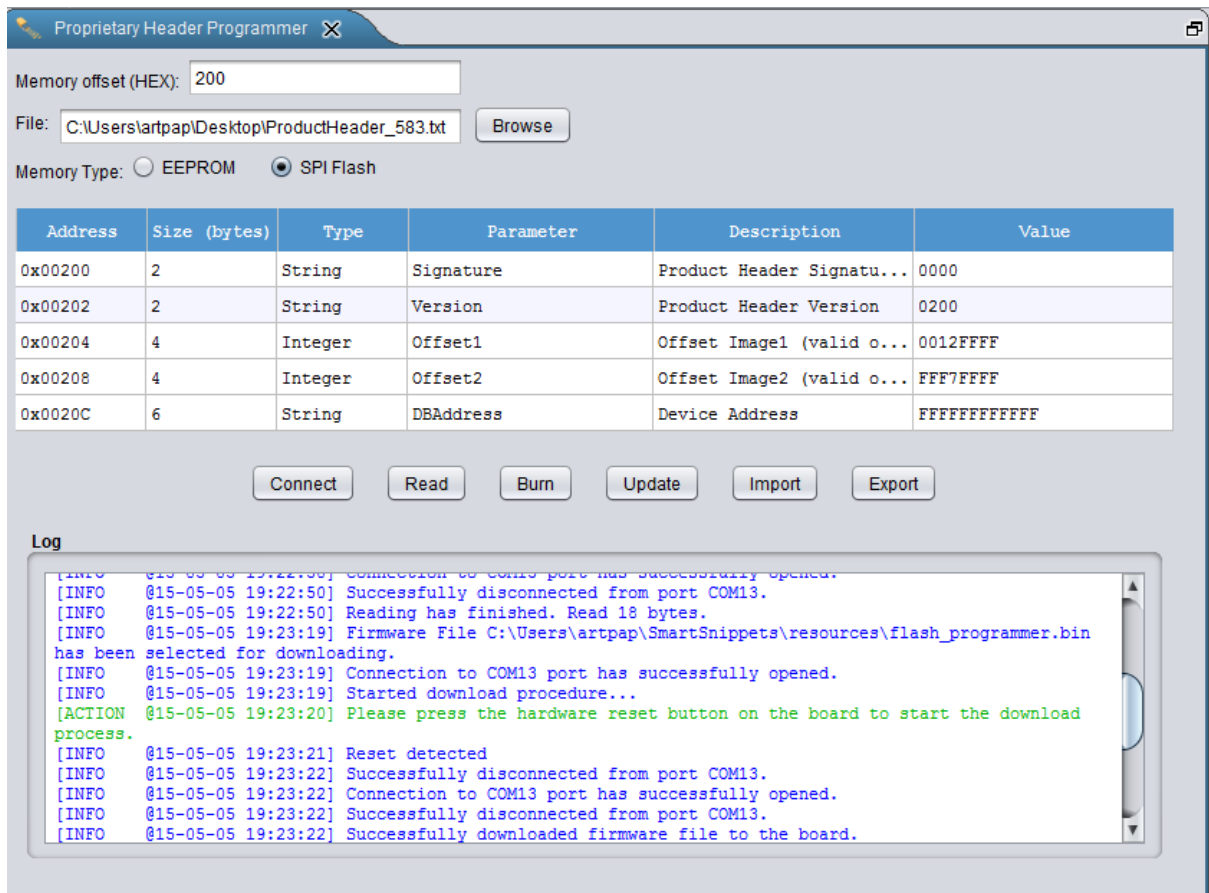
The following memory types are currently supported:

- I2C EEPROM M24M01-R

More information is given in document UM-B-005, Dialog Semiconductor. If someone wants to support other EEPROM flash types, he/she has to modify the flash programmer application included in DA14580 SDK (tools-flash_programmer) and replace flash_programmer.bin file in folder %SMARTSNIPPETS_WORK%resources.

11 Proprietary Header Programmer

Proprietary Header Programmer is used for burning custom header or NVDS to the DA14580 EEPROM or SPI Flash Memory.



The user first has to select the header model file. This is a txt file of the following format: <number of bytes> \t <Parameter_type> \t <Parameter name> \t <Parameter description>, where parameter type is “String” or “Integer” e.g:

```
2   String  Signature   Product Header Signature (7052)
2   String  Version   Product Header Version
4   Integer Offset1  Offset Image1 (valid only in dual image bootloader)
4   Integer Offset2  Offset Image2 (valid only in dual image bootloader)
6   String  DBAddress  Device Address
```

String parameters are patched with leading zeros, if their size is smaller than the number of bytes indicated by the 2nd column of the table. Integer parameters are patched with trailing zeros.

The selected model file and offset are saved to the project.sms file. The user can store a different model file and offset per memory type. Once a valid connection has been established and a valid model file has been loaded, the user will also be able to read / burn from / to the selected memory.

- **Connect:** Special firmware is downloaded to the chip to allow the user interact with the selected memory. This is a mandatory step before enabling the other actions.
- **Read:** The sum of bytes mentioned at the ‘Size (bytes)’ column of the table are read from the selected memory, starting from the specified offset. This offset cannot be greater than the maximum EEPROM/SPI Flash Memory size.
- **Burn:** When trying to burn data at offset *0x00000* of the EEPROM/SPI Flash Memory, user is presented with the option to make it bootable. When the *bootable* option is selected, a special header is added before the data and the data is written starting from address *0x00020* for EEPROM and *0x00008* for SPI Flash.
- **Update:** *Update* button is used in order to update an SPI Flash sector with input data provided by the user. The following actions are performed when this button is pressed:
 1. The contents of the SPI Flash sector containing the input offset are read and stored in byte array in RAM. Each sector’s size is 4KB (4096 bytes).

2. Bytes starting at input offset inside the sector contents byte array are replaced with the byte array created by parsing the value columns of the table.
3. The byte array representing the updated sector data is burned at SPI Flash Memory at the sector starting address.
 - **Import:** The user can import the data to burn to the selected memory by pressing the ‘Import’ button. The user is advised to import a file that has been exported using the *Export* button of the Memory Header/NVDS Programmer tool. When a .hex or .ihex file is imported, it is checked that the total number of parameters and the number of bytes per parameter match the custom model that has been loaded. If they do not match, the user can not import the selected file. When a .bin file is imported, if the size of the file in bytes exceeds the sum of the bytes of the custom model, the extra bytes are ignored. When the size of the file is smaller than the expected number of bytes, according to the loaded custom model, bytes with the default value 0x00 are added to the end of the file bytes.
 - **Export:** The user can export the bytes shown at the *Value* column to a .bin,.ihex or .hex file.

12 OTA Services (over the air services)

The user can select between two over the air services: SPOTA (Software Patch Over The Air) and SUOTA (Software Update Over The Air). By default SUOTA is selected.

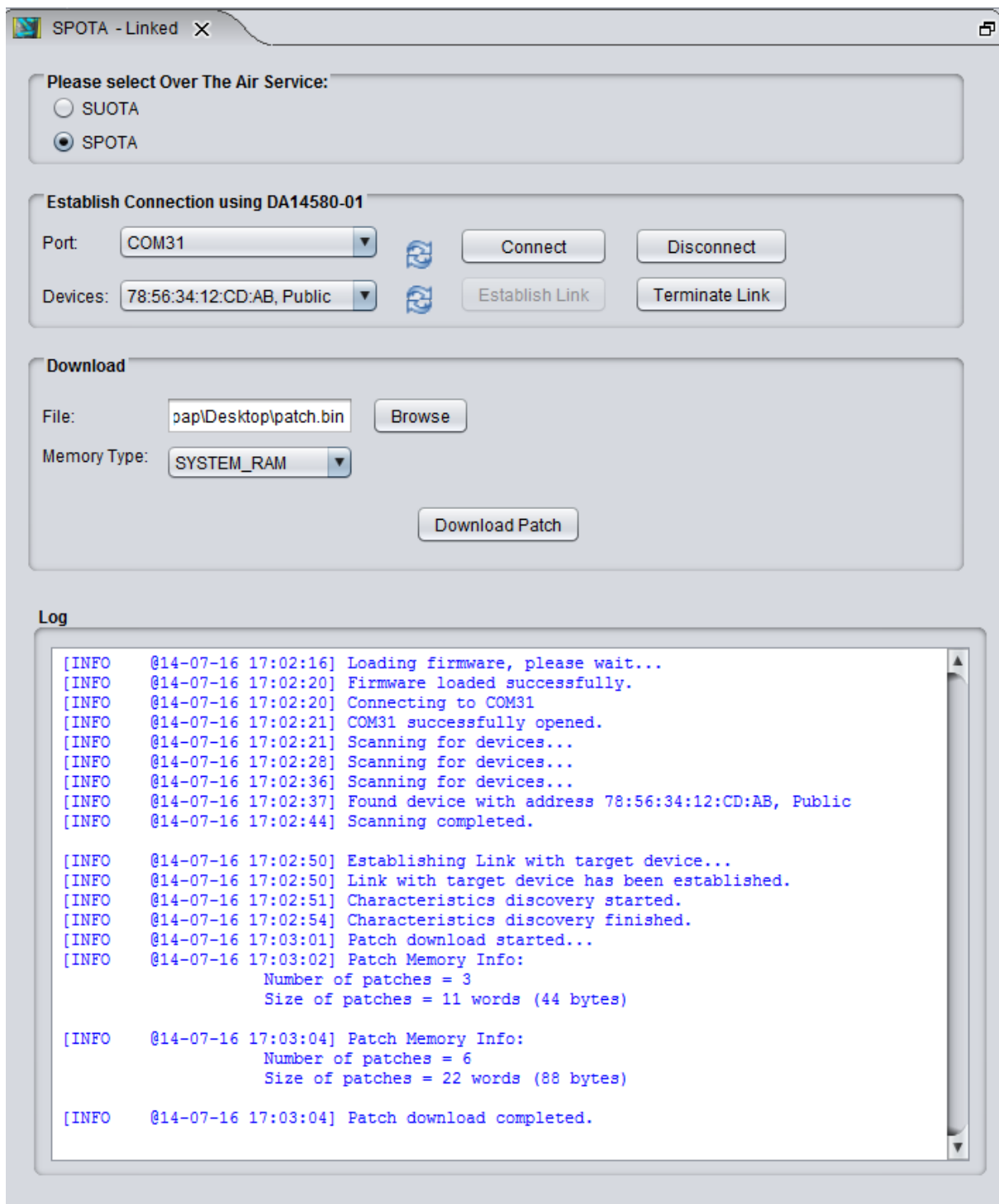
12.1 Software patch over the air (SPOTA)

Software patch over the air (SPOTA) service exposes a control point to allow a peer device initiate software patching over the air. The DA14580 is capable of executing SW patches that vary regarding the target device to be amended as well as the level of changes to be implemented. A patch can just change a single SW variable value in the code which resides in the SRAM. It can also change an instruction or data value read from the ROM used for the protocol realization. Furthermore, a patch can generate an exception and guide the Program Counter to a new function bypassing the existing one.


SPOTA defines 2 roles:

1. The “SPOTA Initiator” is the endpoint which transmits the patch payload
2. The “SPOTA Receiver” is the endpoint which receives and applies the patch payload

Patching code is sent by the Initiator using the Bluetooth Smart link. Receiver stores the patch into the internal RAM or an external non-volatile memory and then applies the patch. SPOTA is instantiated as a Primary Service. There is only one instance of this service on a device.



Link Establishment and Termination

Before being able to download a software patch to the DA14580, we have to establish a link with it via the Bluetooth Smart link. First we have to connect to a DA14580-01 Bluetooth dongle, by selecting the serial port that the dongle is connected to and pressing the **Connect** button. The appropriate firmware is downloaded to the Bluetooth Smart link and a scan is performed in order to detect available bluetooth devices. If the appropriate firmware has been downloaded to the DA14580 and the device is advertising its bluetooth address, it will be detected by the Bluetooth Smart link. As we can see in the previous image, the device with bluetooth address 78:56:34:12:CD:AB has been detected and has been added to the list of available devices. The user can use the two  icons in order to refresh the respective drop down list. In order to connect or establish a link with the DA14580 device, the user has to select the correct bluetooth address from the drop down list and press the

Establish Link button. After a successful link establishment, it is checked whether the target device supports the SPOTA service or not. If it is not supported, the message *Peer device does not support the SPOTA service* will be shown at the log and the user will not be able to continue with the patch download.

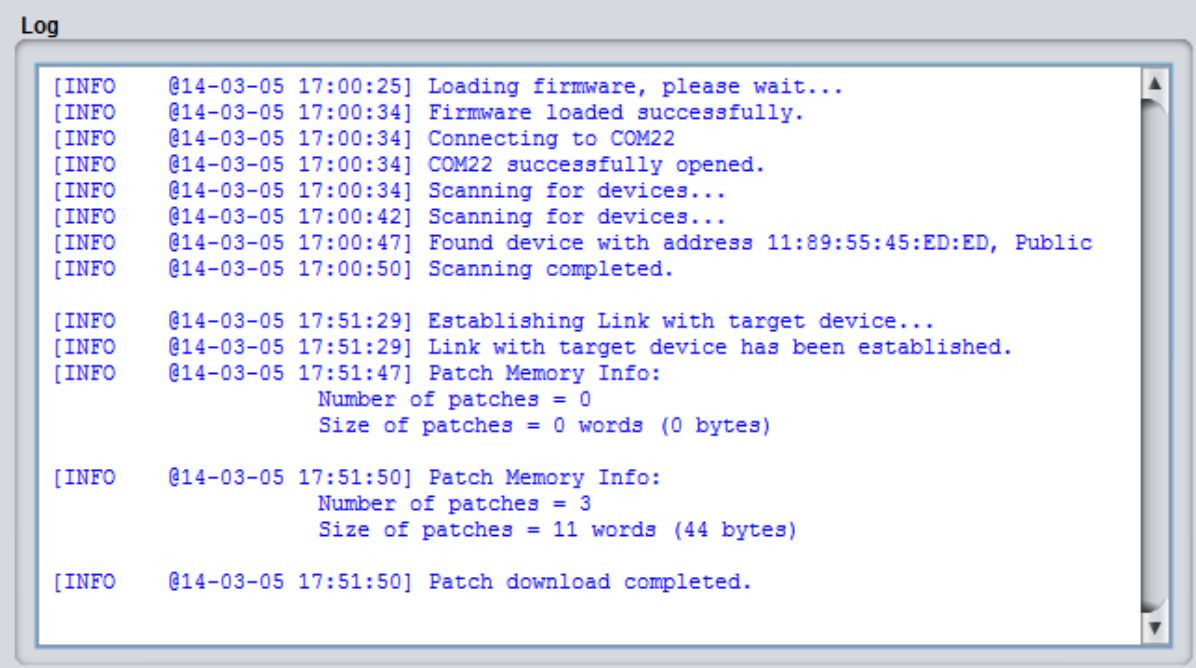
If at any point, user wants to terminate an established link between the dongle and the DK, **Terminate Link** button can be pressed.

Patch Download

The user first selects the file that contains the patch to be applied. Then the memory that the patch will be applied to has to be selected. The available options are:

1. SYSTEM RAM
2. RETENTION RAM
3. SPI (Flash memory)
4. I2C (EEPROM memory)

Depending on the memory type, the user may need to provide the memory base address and the GPIO mapping. If SYSTEM RAM or RETENTION RAM are selected, no extra fields have to be specified and the user can press the **Download Patch** button. In case SPI is selected, the user has to specify the base memory address and the 4 GPIOs. Finally, if I2C is selected, the user has to specify the base memory address, the I2C device address and 2 GPIOs. The maximum transfer unit (MTU) over the Bluetooth Smart Link is 20 bytes, so the patch file is divided and transmitted in packets of size 20 bytes (maximum). If the patch has been successfully downloaded, the number of patches in memory increases, as indicated in the following image:



```
Log
[INFO @14-03-05 17:00:25] Loading firmware, please wait...
[INFO @14-03-05 17:00:34] Firmware loaded successfully.
[INFO @14-03-05 17:00:34] Connecting to COM22
[INFO @14-03-05 17:00:34] COM22 successfully opened.
[INFO @14-03-05 17:00:34] Scanning for devices...
[INFO @14-03-05 17:00:42] Scanning for devices...
[INFO @14-03-05 17:00:47] Found device with address 11:89:55:45:ED:ED, Public
[INFO @14-03-05 17:00:50] Scanning completed.

[INFO @14-03-05 17:51:29] Establishing Link with target device...
[INFO @14-03-05 17:51:29] Link with target device has been established.
[INFO @14-03-05 17:51:47] Patch Memory Info:
                          Number of patches = 0
                          Size of patches = 0 words (0 bytes)

[INFO @14-03-05 17:51:50] Patch Memory Info:
                          Number of patches = 3
                          Size of patches = 11 words (44 bytes)

[INFO @14-03-05 17:51:50] Patch download completed.
```

12.2 Software update over the air (SUOTA)

Similarly to SPOTA, SUOTA refers to a software update that is distributed over Bluetooth Smart link. This functionality can be achieved by using the SUOTA service, which is based on the SPOTA service that has been already described. SUOTA also defines two roles:

1. The “SUOTA Initiator” that transmits the new software image
2. The “SUOTA Receiver” that is responsible for:
3. Receiving a new software image that is sent by the Initiator over the Bluetooth Smart link.

4. Validating the new image and send informative status updates to the Initiator.
5. Storing the new image into an external non-volatile memory (FLASH/EEPROM device)
6. Configuring the bootloader to run the new image.

The screenshot shows the SPOTA software interface with the following sections:

- Please select Over The Air Service:**
 - SUOTA
 - SPOTA
- Establish Connection using DA14580-01**
 - Port:
 - Devices:
- Download**
 - File:
 - Memory Type:
 - Image Bank:
 - I2C Device Address (HEX):
 - Block size (bytes):
 - GPIOs: SCL: SDA:
 -
- Log**

```
[INFO @14-07-16 17:04:30] Firmware loaded successfully.
[INFO @14-07-16 17:04:30] Connecting to COM31
[INFO @14-07-16 17:04:31] COM31 successfully opened.
[INFO @14-07-16 17:04:31] Scanning for devices...
[INFO @14-07-16 17:04:31] Found device with address 78:56:34:12:CD:AB, Public
[INFO @14-07-16 17:04:39] Scanning completed.

[INFO @14-07-16 17:04:39] Establishing Link with target device...
[INFO @14-07-16 17:04:40] Link with target device has been established.
[INFO @14-07-16 17:04:40] Characteristics discovery started.
[INFO @14-07-16 17:04:40] Characteristics discovery finished.
[INFO @14-07-16 17:05:01] Firmware download started...
[INFO @14-07-16 17:05:01] Memory Info:
                        0 bytes

[INFO @14-07-16 17:05:09] Memory Info:
                        26357 bytes

[INFO @14-07-16 17:05:09] Firmware download completed.
[INFO @14-07-16 17:05:14] Target device has been disconnected. Please try to re-establish link.
```

Unlike SPOTA, SUOTA applies only to devices that have an external FLASH or EEPROM memory. Additionally, SUOTA transmits the image file in blocks. The user is able to specify the block size in bytes. Each block will in turn be divided in 20 byte packets, so the block size cannot be less than 20 bytes. Image bank field specifies where and which image file will be loaded. It accepts only 3 values: 0, 1 or 2

0: Update external memory with the oldest image. 1: Put image in bank no. 1, as specified in product header. 2: Put image in bank no. 2, as specified in product header.

After a successful image update, suota receiver reboots the target device and the connection to it is lost. User has to download again the suota receiver firmware via the **Booter** tool, and press the **Connect** button at the OTA services tool in order to discover again the device. If the user tries to download the same firmware file twice, an error like the following will appear:

```
[ERROR [14-07-16 17:10:04] error: SPOTA_SERV_STATUS = 0x15 (SPOTAR_SAME_IMG_ERR) after writing to SPOTA_PATCH_DATA
```

13 Data Rate Monitor

Data Rate Monitor is used in order to monitor the overall receive and transmit rate over bluetooth.

Timestamp	Time Span	Packets Tx	Bytes Tx	Packets Rx	Bytes Rx	Errors
2014-07-17 11:57:56	502.5	0	0	0	0	34
2014-07-17 11:57:57	502.5	0	0	0	0	41
2014-07-17 11:57:57	502.5	0	0	0	0	42
2014-07-17 11:57:58	502.5	0	0	0	0	38
2014-07-17 11:57:58	502.5	0	0	0	0	45
2014-07-17 11:57:59	502.5	0	0	0	0	44
2014-07-17 11:57:59	502.5	0	0	0	0	41
2014-07-17 11:58:00	502.5	0	0	0	0	47
2014-07-17 11:58:00	502.5	0	0	0	0	36
2014-07-17 11:58:01	502.5	0	0	0	0	42
2014-07-17 11:58:01	502.5	0	0	0	0	41
2014-07-17 11:58:02	502.5	0	0	0	0	39
2014-07-17 11:58:02	502.5	0	0	0	0	42
2014-07-17 11:58:03	502.5	0	0	0	0	47
2014-07-17 11:58:03	502.5	0	0	0	0	35
2014-07-17 11:58:04	502.5	0	0	0	0	39
2014-07-17 11:58:04	502.5	0	0	0	0	45
2014-07-17 11:58:05	502.5	0	0	0	0	41
2014-07-17 11:58:05	502.5	0	0	0	0	42
2014-07-17 11:58:06	502.5	0	0	0	0	39

In order to use the Max Data Rate monitor tool, the user needs two DA14580 DKs connected via usb and two SmartSnippets instances. The one DK will play the **Central** role and the other DK will play the **Peripheral** role. The central device can initiate a scan and send connect request to the peripheral device, while the peripheral device can start advertising and accept incoming connection requests. Supposing we have two DKs, *DK_central* and *DK_peripheral*, and two SmartSnippets instances, *central_app* and *peripheral_app*, the following steps have to be followed:

1. Firmware Download

- User starts SmartSnippets *peripheral_app* and selects the ports/port in UART/SPI mode or UART mode that correspond to one of the DKs. From now on this will be referred as *DK_peripheral*. Selects Max Data Rate Monitor tool and presses the Start Peripheral button. Follows the instructions at the log in order to download the firmware.
- User starts *central_app* and selects the ports/port in UART/SPI mode or UART mode that correspond to the second DK, which will be *DK_central*. Selects Max Data Rate Monitor tool and presses the Start Central button.

2. Scan

- From *central_app*, user presses the Scan button and waits for the discovered devices to be listed and for the scanning process to complete.

3. Connect

- User enters connection interval in millisecond as a multiple of 1.25 at *central_app* and presses the Connect button. If the connection to *DK_peripheral* succeeds, connection status panel will be updated with the respective information. Once connected, the stats table will be continuously updated with data, indicating the following:

- Timestamp: Actual time when the packet was received
- Time span: Duration of the connection event
- Packets Tx: Transmitted packets
- Bytes Tx: Transmitted bytes
- Packets Rx: Received packets
- Bytes Rx: Received bytes
- Errors: Packet errors, including both packets that have been erroneously received and packets for which an acknowledgement has not been received.

Counters at Control Panel will also be updated while the two devices are connected. Note that most of the counter values and the values at the stats table are zero at this point.

4. Start Transmission

- By pressing the Start Transmission button from *central_app*, *DK_central* starts sending packets to *DK_peripheral*. Note that Packets Tx and Bytes Tx are now greater than zero at *central_app* while Packets Rx and Bytes Rx are greater than zero at *peripheral_app*. The opposite can be observed if the Start Transmission button is pressed from the *peripheral_app*.

The screenshot shows the 'Data Rate Monitor' application window. It features three main control panels: 'Central control', 'Connection status', and 'Test control'. The 'Central control' panel includes a 'Scan list' with the address '77:00:00:CA:EA:80' and a 'Scan' button. The 'Connection status' panel shows the device '77:00:00:CA:EA:80' is 'Connected'. The 'Test control' panel has buttons for 'Start', 'Stop', 'Reset counters', 'Clear Stats', and 'Export Stats', along with a table of counters and values.

Counter	Value
Total TX packets	68761
Total TX bytes	130036
Total Rx packets	2158
Total Rx bytes	43160
Total packet errors	9432
Tx Packet Rate (pkt/sec)	43.7811
Tx Rate (kbits/sec)	7.0050
Rx Packet Rate (pkt/sec)	43.7811
Rx Rate (kbits/sec)	7.0050

The 'Stats' table at the bottom provides a detailed log of data points:

Timestamp	Time Span	Packets Tx	Bytes Tx	Packets Rx	Bytes Rx	Errors
2014-07-17 12:11:39	502.5	44	660	0	0	76
2014-07-17 12:11:40	502.5	33	660	0	0	76
2014-07-17 12:11:40	502.5	38	760	0	0	76
2014-07-17 12:11:41	502.5	28	560	0	0	74
2014-07-17 12:11:41	502.5	29	580	9	180	78
2014-07-17 12:11:42	502.5	31	620	31	620	84
2014-07-17 12:11:42	502.5	37	740	37	740	77
2014-07-17 12:11:43	502.5	21	420	21	420	76
2014-07-17 12:11:43	502.5	30	600	30	600	81
2014-07-17 12:11:44	502.5	25	500	25	500	79
2014-07-17 12:11:44	502.5	53	1060	53	1060	77
2014-07-17 12:11:45	502.5	40	800	40	800	80
2014-07-17 12:11:45	502.5	28	560	28	560	79
2014-07-17 12:11:46	502.5	23	460	23	460	77
2014-07-17 12:11:46	502.5	34	680	34	680	83
2014-07-17 12:11:47	502.5	21	420	21	420	85
2014-07-17 12:11:47	502.5	27	540	27	540	83
2014-07-17 12:11:48	502.5	31	620	31	620	74
2014-07-17 12:11:48	502.5	29	580	29	580	73
2014-07-17 12:11:49	502.5	23	460	23	460	85

5. Stop Transmission

- This button stops the transmission from the one device to the other.

6. Disconnect

- By pressing the Disconnect button the connection between the two devices closes.

7. Stop Data Rate Monitor


- This button initially tries to close the connection between the two devices, if not already closed via the Disconnect button. Then closes the port that listens for incoming messages from the other DK. This is necessary in order to use another tool with the same com port, like the Booter or the OTP Programmer.

User has the option to reset the counters of the aggregated statistics by pressing the Reset counters button. Moreover, stats table can be cleared by pressing the Clear Stats button. Finally, stats table data can be exported to csv format by pressing the Export Stats button.

14 Working with multiple tools

SmartSnippets framework has been designed to help user achieve maximum productivity by allowing multiple tools be displayed on the screen at the same time. When working with multiple tools, user can:

1. Modify the size of each tool frame
2. Determine the exact location of each tool frame, by dragging the tool frame titlebar and placing it at the upper, lower, left or right side of another tool frame
3. Stack and un-stack tool frames, by dragging the tool frame titlebar and placing it at the center of another tool frame
4. Minimize a tool frame. User can make it visible again by clicking on the corresponding left-hand side toolbox button
5. Maximize a tool frame and restore it to its previous size and location

Every time the user closes the application, all layout-related data is stored and restored next time the application starts. To restore layout to default, user can press the  button of the horizontal toolbar.

15 Working with multiple projects

SmartSnippets framework allows the user work on multiple DA14580 Development Kits on the same laptop or PC by opening multiple SmartSnippets applications, one for each DK. The following assumptions are made:

1. Each SmartSnippets application instance should be opened for a different project
2. Each project should be associated with a different DK board connected to the laptop or PC via its own Virtual COM port pair.

Since more than one SmartSnippets application instance may be running at the same time, the titlebar has been adjusted to provide useful information about the project and the Virtual COM port pair that is currently running on each application instance. E.g. in the image above, the project name is 'TEST' and is currently connected to COM3 and COM4. If 'none' is displayed for the Virtual COM port pair, it means that currently the project is not connected to any Virtual COM port pair.

16 Logs

Almost all tools have their own 'Log' panel at the bottom of their frame. The log messages that are written to these panels help the user understand the status of the tool and the tasks that it performs under the hood (e.g. validation checks in the case of the 'OTP Header' tool before burning the header).

Each log message has the following attributes:

1. **Message Type:** each message belongs to one of the following types: INFO, ACTION, WARNING, ERROR. The message type also determines the message colour
2. **Timestamp:** in YY-MM-DD format
3. **Description:** the actual info, action, warning or error message

For each project, there is a 'log.txt' file located under the project's working directory (i.e. under %SMARTSNIPPETS_WORK%\Projects\<<project name> folder). All log messages that refer to this project are stored in this file, together with a short description of the tool that generated each message.

17 Command-line implementation

SmartSnippets framework also provides a command line implementation of Uart Booter and OTP Programmer. In order to run the command-line version of SmartSnippets, the user has to open the command prompt, change directory to %SMARTSNIPPETS_HOME%\bin directory and execute one of the available commands listed below:

1. **SmartSnippets.exe -help** Displays the available commands and examples
2. **SmartSnippets.exe -type booter -chip chip_version [-jtag jtag_serialNumber | -com_port portNumber] [-sys_ctrl_reg b1] [-gp_ctrl_reg b2] [-close] [-gpio pinId] [-uart TxId] -file filename [-y] [-nomessages messageFilter]** Downloads file to DA14580 using UART Booter tool. If a 'CRC does not match' shows up, please run the command again and then press the hardware reset button on the board to restart the download process.
3. **SmartSnippets.exe -type otp -chip chip_version [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd write_header -file filename [-y] [-nomessages messageFilter]** Burns file to OTP Header. Makes the assumption that a valid firmware file has already been downloaded (e.g. through use of the uartBooter command above). In order to disable validations that stop the header writing process waiting for user confirmation, the following line should be added to the "%SMARTSNIPPETS_WORK%\properties.txt" file: DISABLEVALIDATIONS = true
4. **SmartSnippets.exe -type otp -chip chip_version [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd write_nvds -file filename [-offset ofst] [-y] [-nomessages messageFilter]** Burns file to OTP NVDS. Makes the assumption that a valid firmware file has already been downloaded (e.g. through use of the uartBooter command above).
5. **SmartSnippets.exe -type otp -chip chip_version [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd write_custom_code -file filename [-offset ofst] [-y] [-nomessages messageFilter]** Burns file to OTP memory. Makes the assumption that a valid firmware file has already been downloaded (e.g. through use of the uartBooter command above).
6. **SmartSnippets.exe -type otp -chip chip_version [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd read_header -file filename [-y] [-nomessages messageFilter]** Reads OTP Header and writes its contents to a file. Makes the assumption that a valid firmware file has already been downloaded (e.g. through use of the uartBooter command above).
7. **SmartSnippets.exe -type otp -chip chip_version [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd read_nvds -file filename [-offset ofst] [-y] [-nomessages messageFilter]** Reads OTP NVDS and writes its contents to a file. Makes the assumption that a valid firmware file has already been downloaded (e.g. through use of the uartBooter command above).
8. **SmartSnippets.exe -type otp -chip chip_version [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd read_custom_code -file filename [-offset ofst] [-length byteLength] [-y] [-nomessages messageFilter]** Reads OTP memory contents and writes its contents to a file. Makes the assumption that a valid firmware file has already been downloaded (e.g. through use of the uartBooter command above).
9. **SmartSnippets.exe -type power -com_port portNumber [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd capture_data -duration duration [-time_offset time_offset] [-threshold threshold] -data_file data_file -stat_file stat_file [-y] [-nomessages messageFilter]** Captures Current data measured by Power Profiler, writes them to a file and exports final statistics. Makes the assumption that a valid firmware file has already been downloaded (e.g. through use of the uartBooter command above).
10. **SmartSnippets.exe -type power -com_port portNumber [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd export_stats -duration duration -sampling_interval sampling_interval -stat_file stat_file [-y] [-nomessages messageFilter]** Captures statistics for a period of time and exports to a file statistic samples every sampling interval. Makes the assumption that a valid firmware file has already been downloaded (e.g. through use of the uartBooter command above).
11. **SmartSnippets.exe -type spi -chip chip_version [-clk clk] [-cs cs] [-miso miso] [-mosi mosi] [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd read -file filename [-offset ofst] [-length byteLength] [-y] [-nomessages messageFilter]** Reads SPI FLASH memory contents and writes them to a file.

12. **SmartSnippets.exe -type spi -chip chip_version [-clk clk] [-cs cs] [-miso miso] [-mosi mosi] [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd write -file filename [-offset ofst] [-y] [-nomessages messageFilter]** Burns file to SPI FLASH memory.
13. **SmartSnippets.exe -type spi -chip chip_version [-clk clk] [-cs cs] [-miso miso] [-mosi mosi] [-jtag jtag_serialNumber | -com_port portNumber] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd erase [-offset ofst] [-sectors num] [-y] [-nomessages messageFilter]** Erase all the SPI FLASH memory or part of it.
14. **SmartSnippets.exe -type eeprom -chip chip_version [-scl scl] [-sda sda] [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd read -file filename [-offset ofst] [-length byteLength] [-y] [-nomessages messageFilter]** Reads EEPROM memory contents and writes them to a file.
15. **SmartSnippets.exe -type eeprom -chip chip_version [-scl scl] [-sda sda] [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd write -file filename [-offset ofst] [-y] [-nomessages messageFilter]** Burns file to EEPROM memory.
16. **SmartSnippets.exe -type otp -chip chip_version [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd write_field -offset ofst -data dataInHex [-nvds_address_offset nvdsAddrOfst] [-y] [-nomessages messageFilter]** Burns dataInHex to OTP starting from address 0x40000 + ofst.
17. **SmartSnippets.exe -type spi -chip chip_version [-clk clk] [-cs cs] [-miso miso] [-mosi mosi] [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd write_field -offset ofst -data dataInHex [-y] [-nomessages messageFilter]** Burns dataInHex to SPI FLASH starting from address ofst.
18. **SmartSnippets.exe -type eeprom -chip chip_version [-scl scl] [-sda sda] [-jtag jtag_serialNumber | -com_port portNumber] [-baudrate rate] [-gpio pinId] [-uart TxId] [-firmware firmware_file] -cmd write_field -offset ofst -data dataInHex [-y] [-nomessages messageFilter]** Burns dataInHex to EEPROM starting from address ofst.
19. **SmartSnippets.exe -bundle commands_file** Executes bundle of commands from input txt file.

Option Description:

Option	Description
-type cli_type	booter or power or otp or spi or eeprom.
-cmd command	Available commands are: write_header, write_nvds, write_custom_code, read_header, read_nvds, read_custom_code, capture_data, capture_stats, read, write, erase, write_field. Look previous section for available combinations of -type and -cmd options.
-com_port portNumber	An integer number indicating the UART COM Port. E.g. portNumber=3 for port COM3
-jtag jtag_serialNumber	If -jtag option along with the jtag's serial number has been specified the jtag interface will be used for communication with the DA14580. This option is mutually exclusive with the -com_port option.
-gpio pinId	The GPIO pin which controls the transistor enabling high voltage. Format: Pi_j, where P is character 'P', i is an integer with between 0 and 3 and j is an integer between 0 and 7. E.g.: P1_2
-uart TxId	The Tx part of the UART port Tx-Rx Pair connecting FTDI chip with DA14580 chip. Available TxIds: P0_0, P0_2, P0_4 and P0_6

Continued on next page

Table 1 – continued from previous page

Option	Description
-file filename	Input or output filename. For write commands it is an input bin or hex file, containing the data to be burned in DA14580. For read commands it is an output text, hex or bin file, where data should be stored.
-baudrate rate	Specifies the data transfer rate through UART. Available baudrates are: 57600, 9600, 115200
-offset ofst	The offset in OTP memory, from which read or write operation should start. If 'ofst' starts with '0x' it is parsed as hex, otherwise as decimal
-length lengthInBytes	The length in bytes to read from OTP memory.
-chip chip_version	The version of the chip. Acceptable values are DA14580-00 (and da14580-00) or DA14580-01 (and da14580-01)
-duration durationInmSecs	Duration of data capturing (integer number of msecs). Used in 'capture_data' Power Profiler command
-duration durationInSecs	Duration of data capturing (integer number of secs). Used in 'export_stats' Power Profiler command
-sampling_interval interval	An integer number indicating the number of seconds between the statistic samples
-time_offset time_offset	Pre trigger sampling period (double number of msecs). Optional, default 0
-threshold threshold	Threshold for considering active (in mA). If set, measured values are collected, exported and aggregated as stats only if their value is >= threshold. Optional, default 0.
-data_file data_filename	Filename of .csv file with collected measurements
-stat_file stat_filename	Filename of .csv file with collected statistics (e.g. peak current, avg current, charge)
-y	Disable all popups.
-nvds_address_offset nvdsAddrOfs	The starting offset of NVDS inside OTP memory, with respect to 0x40000. If 'nvdsAddrOfs' starts with '0x' it is parsed as hex, otherwise as decimal. If -nvds_offset option has been specified, write_otp_field command will write the input data at OTP NVDS.
-firmware firmware_file	Firmware file that should be downloaded to DA14580 before executing the command.
-nomessages messageFilter	Disables a message category. messageFilter is a string combined of '1' and '0'. The message categories are INFO, WARNING, ERROR, ACTION, ARGUMENT_PARSING. 00111 disables INFO and WARNING messages. 00111 is equal to 00, since by default all message categories are enabled.
-sectors num	Specifies the number of sectors to be erased from SPI Flash memory. If 'num' starts with '0x' it is parsed as hex, otherwise as decimal.
-bundle commands_file	Executes bundle of commands from input txt file.
-sys_ctrl_reg b1	Byte value to write at System Control Register, e.g. A4 or 0xA4.

Continued on next page

Table 1 – continued from previous page

Option	Description
-gp_ctrl_reg b2	Byte value to write at General Purpose Control Register, e.g. E2 or 0xe2.
-close:	Call JLINKARM_Close instead of JLINKARM_Go when downloading the firmware.
-cs cs	Specifies the CS pin configuration for SPI Flash memory. Available values: [P0_0:P0_7], [P1_0:P1_3], [P2_0:P2_9], [P3_0:P3_7]
-clk clk	Specifies the CLK pin configuration for SPI Flash memory. Available values: [P0_0:P0_7], [P1_0:P1_3], [P2_0:P2_9], [P3_0:P3_7]
-miso miso	Specifies the MISO pin configuration for SPI Flash memory. Available values: [P0_0:P0_7], [P1_0:P1_3], [P2_0:P2_9], [P3_0:P3_7]
-mosi mosi:	Specifies the MOSI pin configuration for SPI Flash memory. Available values: [P0_0:P0_7], [P1_0:P1_3], [P2_0:P2_9], [P3_0:P3_7]
-scl scl	Specifies the SCL pin configuration for I2C EEPROM memory. Available values: [P0_0:P0_7], [P1_0:P1_3], [P2_0:P2_9], [P3_0:P3_7]
-sda sda	Specifies the SDA pin configuration for I2C EEPROM memory. Available values: [P0_0:P0_7], [P1_0:P1_3], [P2_0:P2_9], [P3_0:P3_7]

Non-mandatory options:

In commands listed above, optional arguments are placed between square brackets (e.g. [rate]). If user doesn't set an optional argument, the following default values are used:

Option	Default Value
[-gpio pinId]	P1_2
[-uart TxId]	P0_4
[-baudrate rate]	57600
[-offset ofst]	0x0000
[-length byteLength]	32768 (0x8000)
[-time_offset time_offset]	0
[-threshold threshold]	0
[-sectors num]	1
[-sys_ctrl_reg b1]	0xA4
[-gp_ctrl_reg b2]	0x2E
[-cs cs]	DA14583: P2_3, other: P0_3
[-clk clk]	DA14583: P2_0, other: P0_0
[-miso miso]	DA14583: P2_4, other: P0_5
[-mosi mosi]	DA14583: P2_9, other: P0_6
[-scl scl]	DA14583: P0_2, other: P0_2
[-sda sda]	DA14583: P0_3, other: P0_3