Application Note

DA14580 Interfacing with External Memory

AN-B-023

Abstract

This Application Note describes how to use an external memory (I2C EEPROM or SPI Flash) with the DA14580. The main differences between the internal OTP and an external memory are also presented.

Contents

Figures

Tables

1 Terms and definitions

BLE	Bluetooth® Low Energy
CLI	Command Line Interface
DK	Development Kit
GUI	Graphical User Interface
I2C	Inter-Integrated Circuit
OTP	One-Time Programmable (memory)
SDK	Software Development Kit
SPI	Serial Peripheral Interface
SW	Software
UART	Universal Asynchronous Receiver/Transceiver

2 References

1. AN-B-001, DA14580 Booting from Serial Interfaces, Dialog Semiconductor
2. DA14580, Data sheet, Dialog Semiconductor
3. UM-B-004, DA14580 Peripheral Drivers, Dialog Semiconductor
4. UM-B-012, DA14580 Secondary Boot Loader, Dialog Semiconductor
5. UM-B-025, DA14580 Bluetooth Smart Development Kit-Basic, Dialog Semiconductor
6. UM-B-034, DA14580 Bluetooth Smart Development Kit-PRO, Dialog Semiconductor

3 Introduction

The DA14580 includes user memory of 32kB of OTP (One Time programmable), 42kB of SRAM and additional blocks of retention RAM. The flexibility of the device allows the DA14580 to also use external SPI/I2C FLASH or EEPROM in a variety of different use cases. This includes booting from external memory, using external memory to store critical data parameters, provide flexibility for the DA14580 to support complete Over the Air image update, or booting from memory within an external MCU (externally hosted application). OTP memory gives optimum performance in terms of low power and cost , and coupled with the ability to use external FLASH or EEPROM, the DA14580 can support the most basic of configurations highly optimised for system cost right through to more feature rich configurations requiring image updates or multiple images. Please refer to [1] and [4], for the booting sequence of DA14580 from an external SPI or I2C memory. The following table sums up the pros and cons of using the internal OTP or an external memory (SPI or I2C).

Table 1: Pros and cons of using internal or external memory

	Advantages	Disadvantages
Internal memory (OTP)	No added cost	Can be programmed only once
	Minimum energy needed to load application into SRAM	Needs an extra pin of 6.8V to be programmed
	Already integrated in the DA14580. No extra board space	
		The memory size is fixed to 32kB
External memory (I2C or SPI)	The memory size can be increased. Flexible memory configuration	Added cost ($). Typically, a SPI Flash is cheaper than the I2C EEPROM.
	Can be programmed many times, which can be handy using SW design. Software updates can be downloaded.	Added PCB surface + added PCB design time.
		Impact on coin cell battery life due to the high erase current
		Consume more energy
		Mirror time (= loading SW from external memory to SRAM) can be very long

4 Internal OTP vs External memory usage

4.1 Comparison between the internal OTP and external memories

The table below shows the main differences between the internal OTP and an external memory (SPI & I2C).

Table 2: Comparison between external memory and OTP

	DA14580 with OTP usage	DA14580 without OTP usage	
	OTP	**SPI Flash (W25X10CL)**	**I2C EEPROM (M24M01-RMN)**
Supply voltage	0.9V – 3.3V	2.3 – 3.6V	1.8 – 5.5V
Memory size	32 kB	128 kB	128 kB
Peak read current (@ 1 MHz)	0.6 mA	2 mA	1.5 mA
Programming / Erase current	1.5 mA (only once during production)	15 mA	2 mA
Time to mirror to SRAM (with 32 kB of data)	1.2 ms *(1)*	202.2 ms *(2)*	3.18 sec *(3)*
Energy consumption during mirroring to SRAM	2.3 µC	385 µC (using default 2 MHz clock speed for SPI bus)	5655 µC (using default 100 kbit/s speed for I2C bus)
Deep power down current	0µA (switched off during the sleep time)	1µA	3µA
Operating range	-40ºC to +85ºC	-40ºC to +85ºC	-40ºC to +85ºC

(1) To copy the OTP content into the SRAM, a word of 32 bits (= 4 bytes) is loaded every 2 clock cycles. The frequency of the parallel bus is 14.5 MHz, so here is the formula:

$$t = \frac{1}{14\,500\,000\ Hz} \times \frac{32 \times 1024}{4} \times 2 = 1.13\ ms$$

- It has to be taken into account that the OTP cell remains active for some time before the starting of the mirroring and for some time after the ending of the mirroring. Thus it consumes power for a bit more time than the real duration of the mirroring.

(2) Theoretical calculation for the mirroring time using an external SPI memory:

$$t = (\text{Number of bytes}) \times \left(\frac{1}{SPI\ Clock\ speed} \times 8 + 2.62\ \mu s\right)$$

- With a **2MHz** SPI clock speed, 8 clock cycles are followed by a gap of **2.62 µs** which is due to the polling of the SPI registers from the DA14580 processor.
- With an **8MHz** SPI clock speed, 8 clock cycles are followed by a gap of **0.19 µs** which is due to the polling of the SPI registers from the DA14580 processor.

(3) Theoretical calculation for the mirroring time using an external I2C memory:

$$t = \frac{1}{100\ kBit/s} \times \frac{Image\ size\ (in\ byte)}{32\ bytes\ block} \times (297 + 27)$$

The DA14580 reads the data from the external I2C memory in packets of 32 bytes.
To read the packet it needs to issue a write command and then a read command to read the 32 bytes plus ACK.

- Write command 3 bytes (slave address + memory address) + ACK bit: total 27 clocks.
- Read command 1 byte (slave address) + 32 bytes data (33 * 9clocks): total 297 clocks.

5 Hardware configuration of the external memory

The application code can be stored into an external SPI or I2C memory which is limited to address of 3 address bytes (that is a memory size of 16MB).

The DA14580 can boot from an external memory when the OTP is blank (see [1]). The schematics are shown on the next page according to the type of external memory which has to be connected to the DA14580. Note that for isolation purposes, series of 1kΩ may be used between the Tx_DA14580/Rx_DA14580 pins to UART-TTL board in production or bench.

Important note:

If the OTP is blank, the assignment for the booting port of the SPI memory cannot be changed (refer to [1]). However, by burning the secondary boot loader into the OTP, the pinout for the booting (from SPI or I2C memory) can be arbitrary modified. Please refer to [4]. In fact, the DA14580 operates in two modes [1, 2], namely the “Normal Mode” and the “Development/Calibration Mode” hereafter addressed as “DevMode”. At power up or reset of the DA14580 the primary boot code (ROM code) will check if the OTP memory is programmed. In this case the DA14580 enters “Normal Mode” and proceeds with mirroring the OTP contents to System RAM and program execution. Otherwise, it enters “DevMode” and it scans a predefined number of pins to communicate with external devices using the three interfaces available on chip: UART, SPI and I2C. The SPI port assignment is shown in the Table 4. An option for using different pin assignment for the SPI flash is given using the secondary bootloader burned in OTP [4].

What’s more, the following table must be taken into account when UART and SPI ports are used at the same time.

Table 3: Connections of DA14580 and SPI/UART

DA14580 Port	SPI assignment	UART assignment
P0_0	SCK	**NOT POSSIBLE**
P0_1	-	**NOT POSSIBLE**
P0_2	-	Tx1
P0_3	CS	Rx1
P0_4	-	Tx2
P0_5	MISO	Rx2
P0_6	MOSI	Tx3
P0_7		Rx3

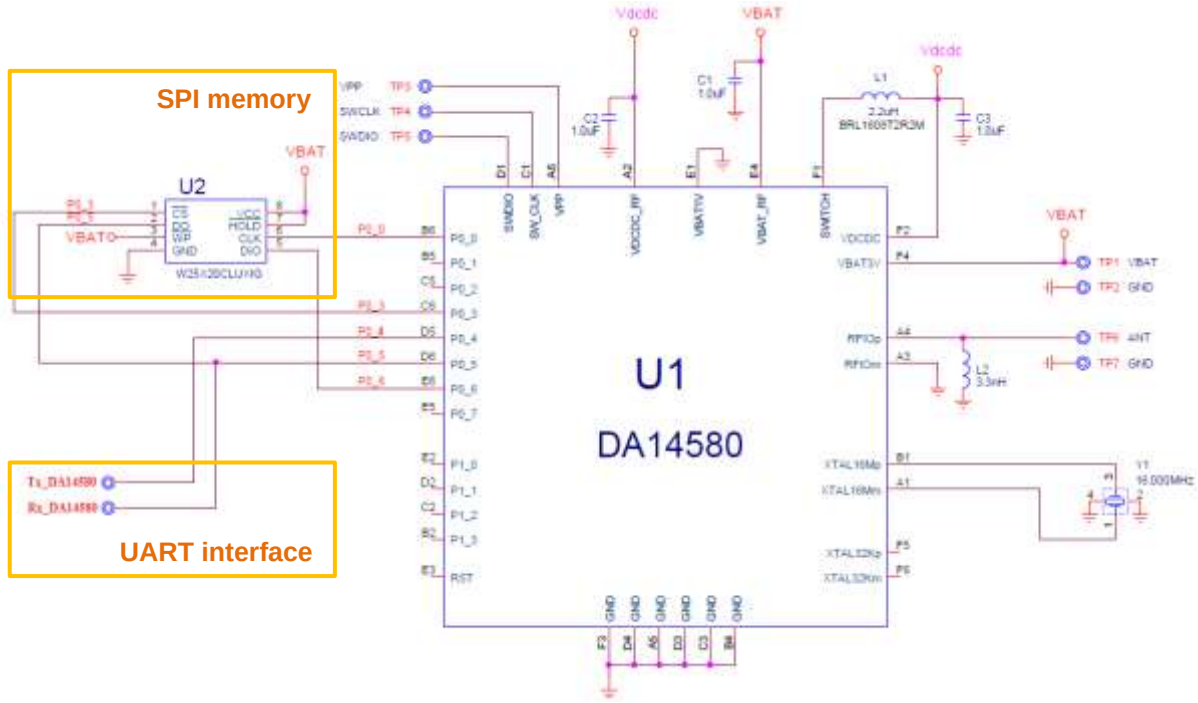
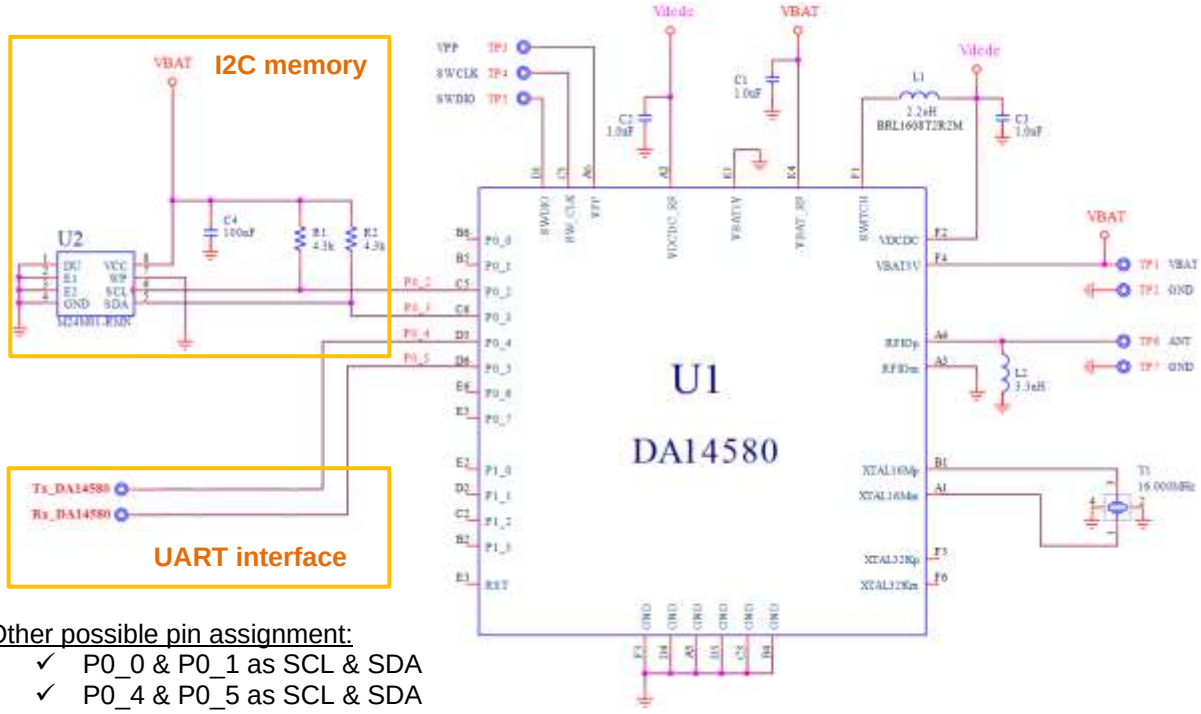


Figure 1: Booting from external SPI Flash memory with the DA14580 (blank OTP)



Other possible pin assignment:

- ✓ P0_0 & P0_1 as SCL & SDA
- ✓ P0_4 & P0_5 as SCL & SDA
- ✓ P0_6 & P0_7 as SCL & SDA

Figure 2: Booting from external I2C EEPROM memory with the DA14580 (blank OTPP)

5.1 With the use of the DA14580 PRO Development kit

The DA14580 Development kit-PRO is populated with the W25X20 CL SPI memory IC (Memory size: 256 kB, Page size: 256 Bytes). Figure 5 shows the required jumper settings for accessing the SPI memory. However, on this board the application is downloaded from the PC to the SPI memory via the JTAG interface. Please see [6] for more details about how to use this Kit-PRO.

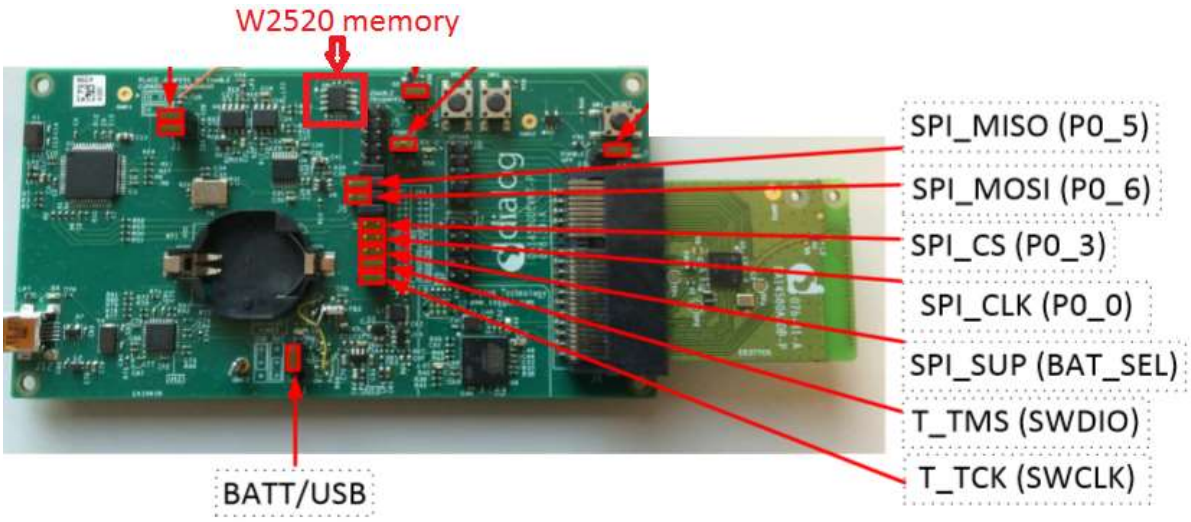


Figure 3: Development kit-PRO

5.2 With the use of the DA14580 BASIC Development kit

The DA14580 Development kit-Basic is populated with the W25X20 CL SPI memory IC (Memory size: 256 kB, Page size: 256 Bytes). Figure 5 shows the required jumper settings for accessing the SPI memory. However, on this board the application is downloaded from the PC to the SPI memory via the JTAG interface. Please see [5] for more details about how to use this Kit-Basic.

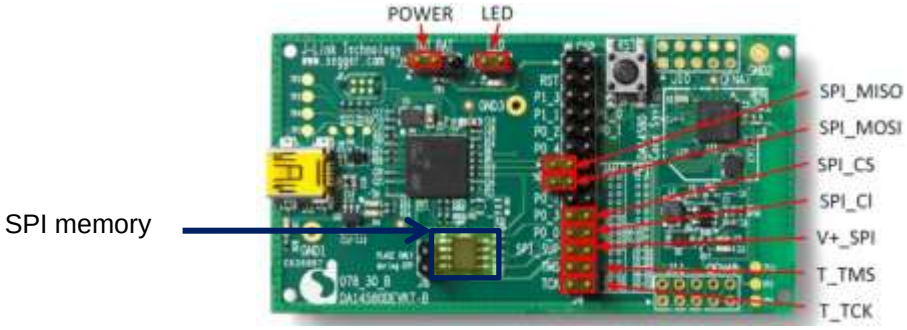


Figure 4: Development kit-Basic

6 Tools & Support for programming OTP and external memory

SmartSnippets is the appropriated tool to download your application into the external memory very easily. It can be downloaded from http://support.dialog-semiconductor.com/software-downloads. Using the development kit, there are two possibilities to download the image into the external memory (by either choosing UART (see figure 2 for the setting of the jumper) or JTAG) as shown in figure 6. The SmartSnippets tool is targeting the main activities of programming and reading the power performance of the DA14580. It enables:

- programming the internal OTP with the actual application compiled image
- downloading an application binary file to the external SPI/I2C memory over UART or JTAG

- accurate examination of the power profile and how it is affected by application software (can be only used with the development kit)
- downloading a SW image to SRAM over UART or JTAG and execute (for development kit-Basic, only the JTAG is used)

The SmartSnippets tool 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. All the tests have been carried out with the **SmartSnippets v3.1**. For more information, please go to the **HELP** tab in the SmartSnippets GUI.

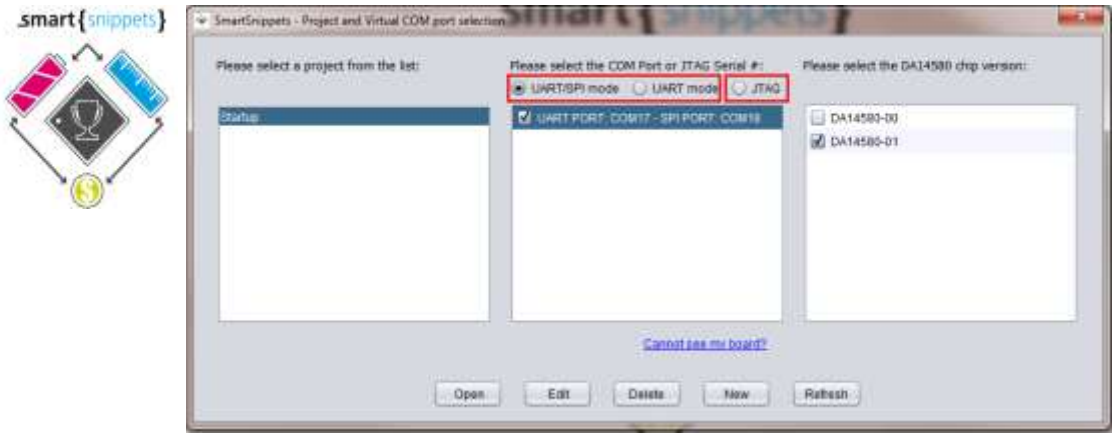


Figure 5: Mode to download the application

By default, the UART port must be connected to P04/P05 as Tx/Rx to download the application. However, this can be changed as mentioned in the next chapter. If the case of UART mode is chosen to download the application, make sure that the UART pins are connected according to the board setup as follow:

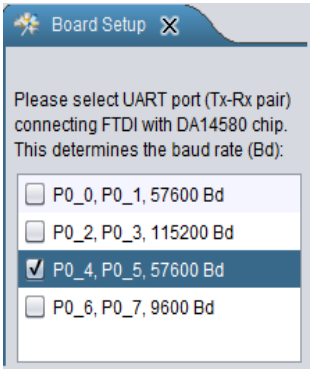


Figure 6: Board setup if UART is used

Having chosen the appropriated mode to download the application, then the type of memory must be selected from the left side of the GUI (SPI Flash Programmer or EEPROM programmer tab).

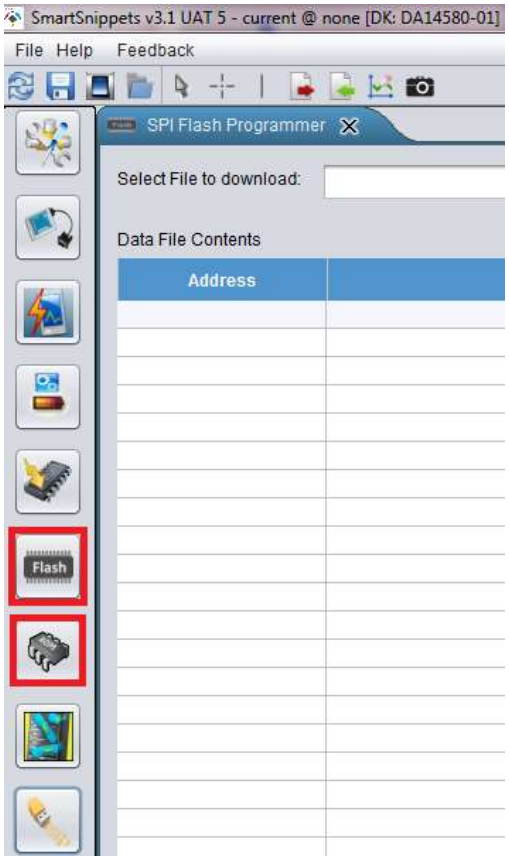


Figure 7: Memory programmers tab

The following steps show how to download an image into an external SPI memory over the UART. The figure 10 shows the steps mentioned below.

1) Browse for the HEX file of the image needed to load into external memory,
2) Press the CONNECT button to establish a connection with the DA14580,

   ⇨ ***"Successfully downloaded firmware file to DA14580"*** is displayed in the log windows.
3) Specify in hexadecimal the memory size, by default use a 128 kB memory size (0x20000 ⇔ 131 072 Bytes => 128kBytes).
4) Set the starting address where the image has to be stored. Blank means starting address 0.
5) Press the BURN button.
6) If the BURN operation has been chosen, choose YES (to boot from the external memory) when the pop-up window appears.

   ⇨ ***"Memory burning completed successfully"*** is displayed in the log windows.

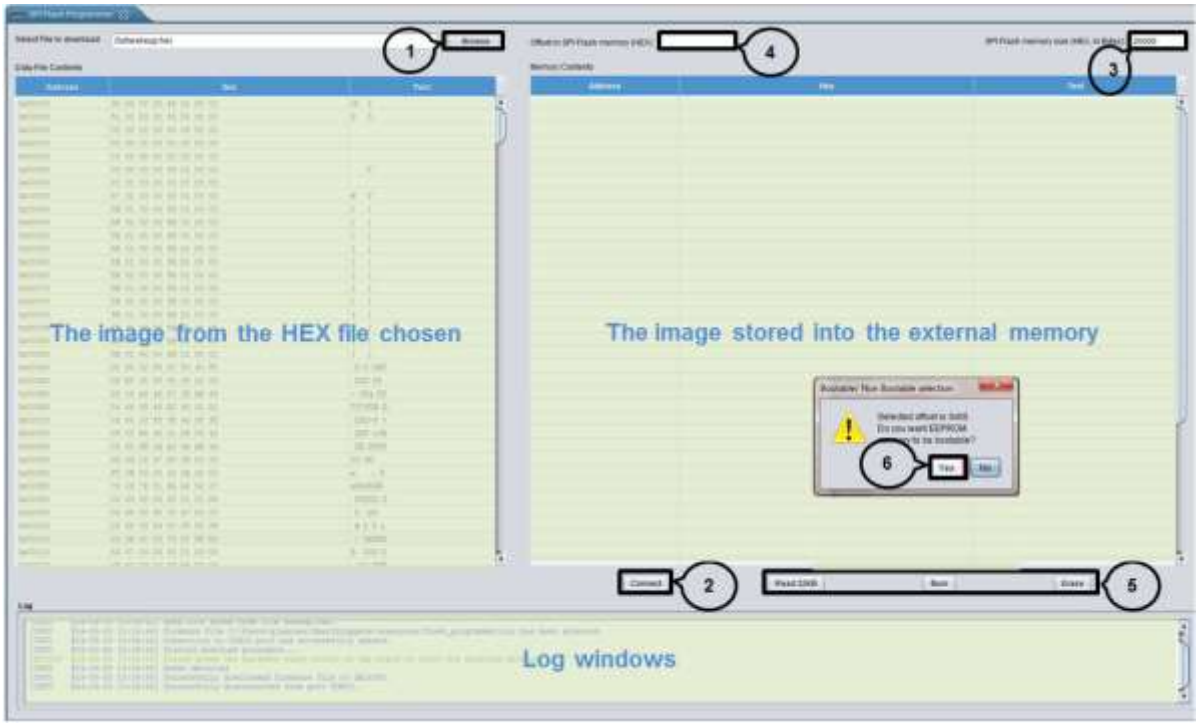


Figure 8: External memory tab

Note:

Before writing any data into the SPI memory, make sure to ERASE the memory first. In flash, data cannot be overwritten; an ERASE operation must be done beforehand.
When doing the programming, production or design parameters can be stored into the external memory, so some specific sectors can be erased by clicking on the Erase sector button has shown below.

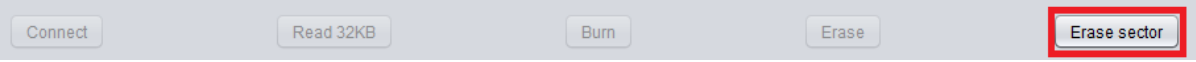


Figure 9: ERASE sector button

6.1 Procedure to change the configuration of SmartSnippets

This section shows how to modify the default SmartSnippets settings. The default parameters are the followings:

The booting port for the I2C memory can have the following combination (refer to [1)]:

- ✓ P0_0 & P0_1 as SCL & SDA
- ✓ P0_2 & P0_3 as SCL & SDA
- ✓ P0_4 & P0_5 as SCL & SDA
- ✓ P0_6 & P0_7 as SCL & SDA

The only booting port for the SPI memory (without the use of the secondary boot loader) is:

- ✓ P0_0, P0_3, P0_5, P0_6 as SCK, CS, MISO & MOSI

The following parameters can be changed from the flash_programmer project:

- ✓ Memory port and UART port
- ✓ Clock speed
- ✓ Memory size
- ✓ Memory page size

Here are the steps to change the default parameters:

1) Open & compile the flash programmer project found in the path: *DA14580_SDK_version\tools\flash_programmer\ flashprogrammer.uvproj.*

2) Search for the *periph_setup.h* file. From this file, the port, clock frequency, memory size and memory page size can be changed as shown:

```
// Select EEPROM characteristics
#define I2C_EEPROM_SIZE    0x20000          // EEPROM size in bytes
#define I2C_EEPROM_PAGE    256              // EEPROM's page size in bytes
#define I2C_SLAVE_ADDRESS 0x50              // Set slave device address
#define I2C_SPEED_MODE     I2C_FAST         // 1: standard mode (100 kbits/s), 2: fast mode (400 kbits/s)
#define I2C_ADDRESS_MODE   I2C_7BIT_ADDR    // 0: 7-bit addressing, 1: 10-bit addressing
#define I2C_ADDRESS_SIZE   I2C_2BYTES_ADDR // 0: 8-bit memory address, 1: 16-bit memory address, 3: 24-bit memory address

// SPI Flash options
#define SPI_FLASH_SIZE 131072               // SPI Flash memory size in bytes
#define SPI_FLASH_PAGE 256                  // SPI Flash memory page size in bytes
//SPI initialization parameters
#define SPI_WORD_MODE   SPI_8BIT_MODE       // 0: 8-bit, 1: 16-bit, 2: 32-bit, 3: 9-bit
#define SPI_SMN_MODE    SPI_MASTER_MODE     // 0: Master mode, 1: Slave mode
#define SPI_POL_MODE    SPI_CLK_INIT_HIGH   // 0: SPI clk initially low, 1: SPI clk initially high
#define SPI_PHA_MODE    SPI_PHASE_1         // If same with SPI_POL_MODE, data are valid on clk high edge, else on low
#define SPI_MINT_EN     SPI_NO_MINT         // (SPI interrupt to the ICU) 0: Disabled, 1: Enabled
#define SPI_CLK_DIV     SPI_XTAL_DIV_2      // (SPI clock divider) 0: 8, 1: 4, 2: 2, 3: 14
// UART
#define UART_GPIO_PORT  GPIO_PORT_0
#define UART_TX_PIN     GPIO_PIN_4
#define UART_RX_PIN     GPIO_PIN_5
#define UART_ENABLED

// SPI
#define SPI_GPIO_PORT   GPIO_PORT_0
#define SPI_CS_PIN      GPIO_PIN_3
#define SPI_CLK_PIN     GPIO_PIN_0
#define SPI_DO_PIN      GPIO_PIN_6
#define SPI_DI_PIN      GPIO_PIN_5
// EEPROM
#define I2C_GPIO_PORT   GPIO_PORT_0
#define I2C_SCL_PIN     GPIO_PIN_2
#define I2C_SDA_PIN     GPIO_PIN_3
```

3) Press the Rebuild all target files button in order to generate the updated hex file,

4) Convert the *flashprogrammer.hex* into *flashprogrammer.bin* (by downloading the file converter from http://www.keil.com/download/docs/7.asp).

5) Replace this new binary file into the **resources** folder from the directory path where SmartSnippets has been installed.

7 Power consumption

After the application image has been copied in the SRAM of DA14580, a system with an external flash or EEPROM will have almost the same power consumption as a system running with the internal OTP because the external memory is not needed any more. The external memory can be put into ultra-deep power down mode (=200nA). Thus, the main difference of the energy consumption between the internal OTP and an external memory is caused by copying the image into the SRAM of the DA14580.

All the tests have been carried out with the HID keyboard (Integrated Processor Solution) application. The code size of this application is **30.56 kB with a power supply of 3V**.

7.1 Booting from external SPI memory in development mode

In this mode, the OTP is blank. The energy consumption of a READING operation is going to be measured.

- Default clock frequency of the SPI bus: 2 MHz, Power supply = 3V.

From theoretical point of view, the time needed to load 30.56 kB of data from the external SPI memory to the SRAM would be:

$$t = (\text{Number of bytes}) \times (\frac{1}{2\ MHz} \times 8 + 2.62\ \mu s)$$

$$t = (30560) \times (500\ \text{ns} \times 8 + 2.62\ \mu s)$$

$$t = 202.3\ ms$$

In practise, the time needed is 202.2 ms.
The amount of charge required during the mirroring is about 385 μC.

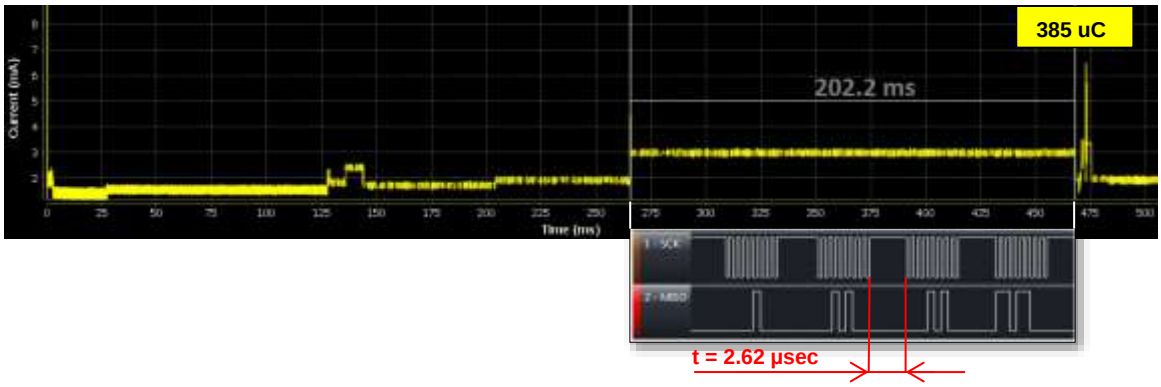


Figure 10: SPI memory mirroring into SRAM

7.2 Booting from external SPI memory with the secondary bootloader

In this mode, the secondary bootloader has been created and programmed in OTP to change the default boot sequence and improve the SPI speed up to 8MHz. However, the speed remains the same of the I2C bus. The energy consumption of a READING operation is going to be measured.

The secondary bootloader has been burnt into OTP for the purpose of having a SPI boot up as fast as possible.

- Default clock frequency of the SPI bus: 8 MHz, Power supply = 3V.

From theoretical point of view, the time needed to load 30.56 kB of data from the external SPI memory to the SRAM would be:

$$t = (\text{Number of bytes}) \times (128\ \text{ns} \times 8 \times +0.19\ \mu s)$$

$$t = (30560) \times (128\ \text{ns} \times 8 \times +0.19\ \mu s)$$

$$t = 36.4\ ms$$

In practise, the time needed is 36.4 ms.
The amount of charge required during the mirroring is about 116.5 μC

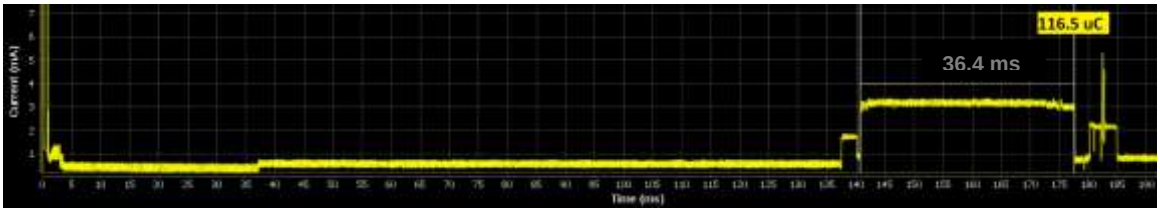


Figure 11: SPI memory mirroring into SRAM (with secondary bootloader)

7.3 Booting from external I2C memory in development mode

In this mode, the OTP is blank. The energy consumption of a READING operation is going to be measured.

- Default speed of the I2C bus: 100 kbit/s, Power supply = 3V.

From theoretical point of view, the time needed to load 30.56 kB of data from the external I²C memory to the SRAM would be:

$$t = \frac{1}{100\ kBit/s} \times \frac{Image\ size\ (in\ byte)}{32\ bytes\ block} \times (297 + 27)$$

$$t = 10\ \mu s \times \frac{30\ 560}{32} \times (297 + 27)$$

$$t = 3.1\ sec$$

In practise, the time needed is 3.18 sec.
The amount of charge required during the mirroring is about 5655 μC.

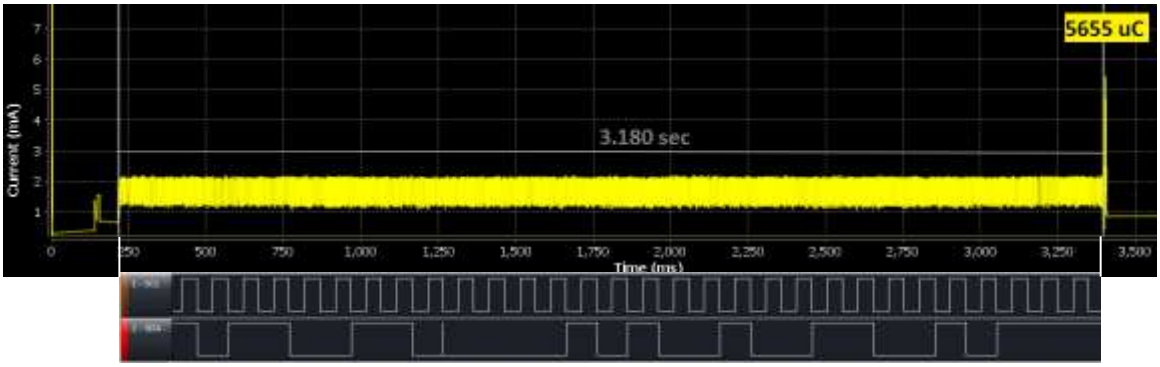


Figure 12: I2C memory mirroring into SRAM

7.4 Summary

Depending on the memory type from where the DA14580 has to boot, it can have quite some impact on the battery. The following setup summarizes the measurements.

Table 4: Energy vs time vs memory type

Setup	Time needed	Energy consumption (@ 3V)
Application of 30.56 kB loaded from OTP to SRAM	1.2 ms	2.3 μC

Application of 30.56 kB copied from SPI memory to SRAM	With blank OTP	202.2 ms	385 µC
	With secondary bootloader burnt into OTP	36.8 ms	116.5 µC
Application of 30.56 kB copied from I2C memory to SRAM	With blank OTP	3180 ms	5655 µC

8 Energy consumption during the memory programming

The energy consumption of a WRITING operation is going to be measured. The HID keyboard application (30.56 kB) has been downloaded via JTAG using the SmartSnippets GUI for the three following cases.

8.1.1 External SPI memory

- SPI port: P0_0 = CLK, P0_3 = CS, P0_5 = MISO, P0_6 = MOSI

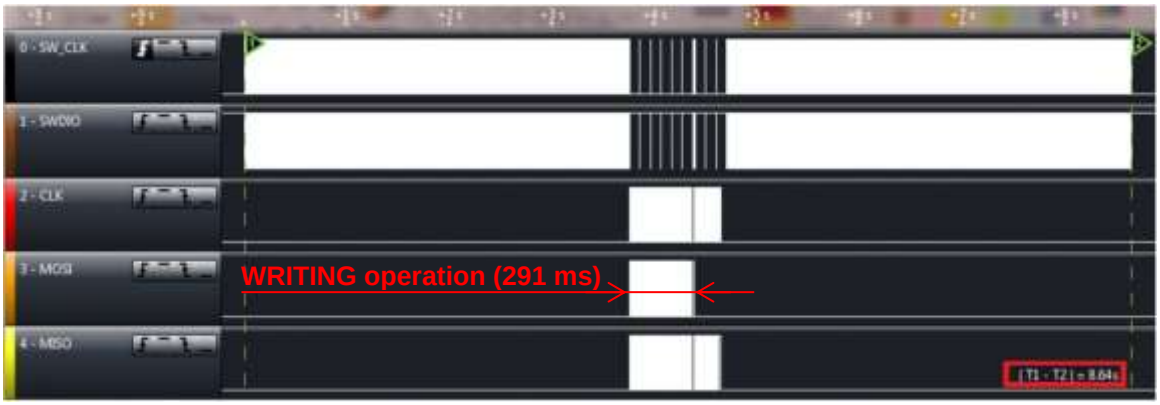


Figure 13: Writing operation time of the SPI memory

The total time to download the application into an external SPI memory is equal to 8.64 seconds. The charge needed for the WRITING operation (MOSI line) into the external SPI memory is 1128.5 µC as shown below. Peaks of current for the writing operation are equal to 11 mA (JTAG consumes 1 mA). Measurements are shown below.

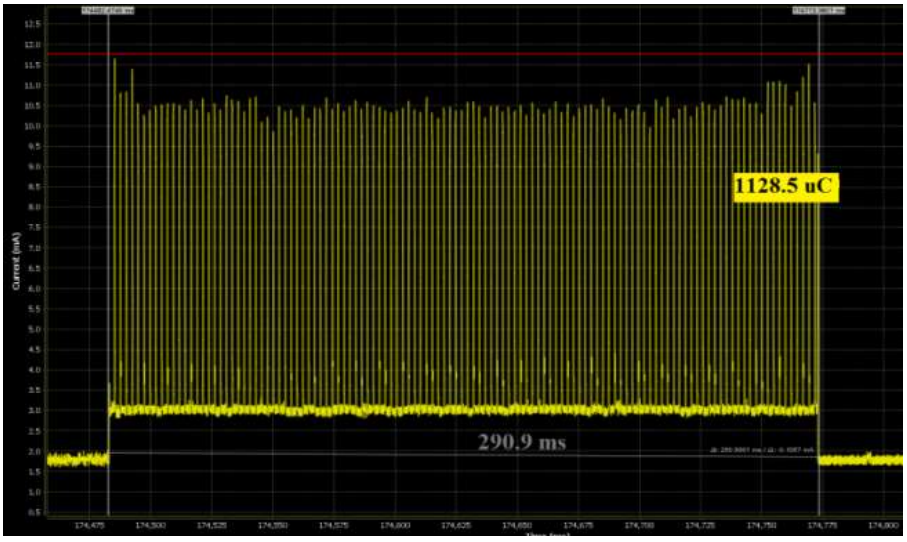


Figure 14: Energy consumption during the writing operation into SPI memory

8.1.2 External I2C memory

- I2C port: P0_2 = SCL, P0_3 = SDA

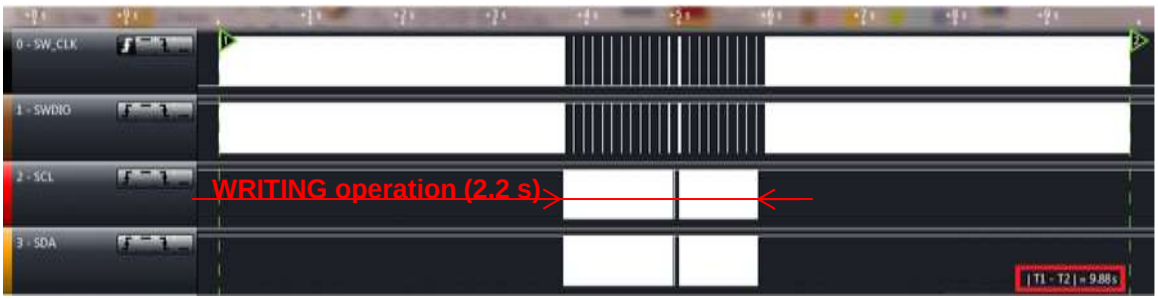


Figure 15: Writing operation time of the I2C memory

The total time to download the application into an external I2C memory is equal to 9.88 seconds from the figure above. The charge needed for the WRITING operation (so during activities of SCL line) into the external I2C memory is 6096.3 μC as shown below. Peaks of current for the writing operation are equal to 2 mA (JTAG consumes 1 mA)

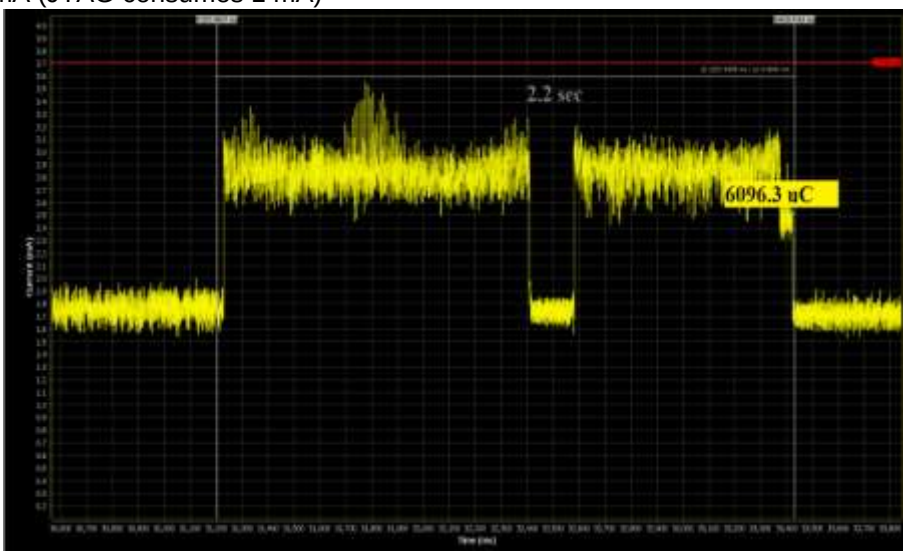


Figure 16: Energy consumption during the writing operation into I2C memory

8.1.3 Internal OTP

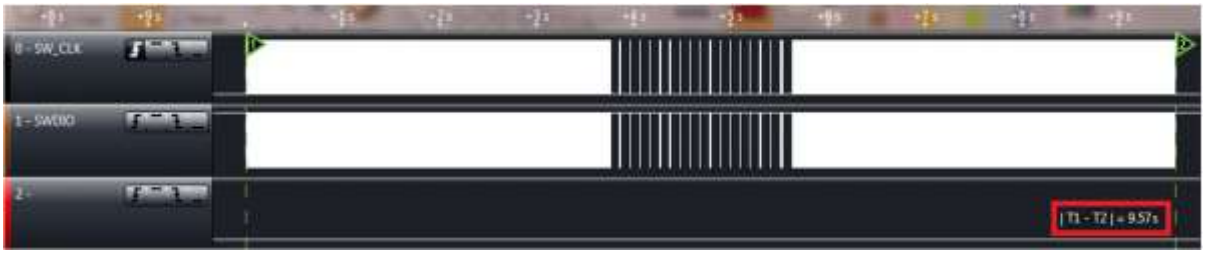


Figure 17: Time needed to laod application into OTP

⇨ The total time is equal to 9.57 seconds.

8.1.4 Summary

Table 5: Time and charge during the writing operation

Memory type	Time needed to load a 30.56 kB application with JTAG	Charge during the writing operation (@ 3V)
External SPI flash	8.6 seconds	1128.5 µC during 291 msec
External I2C EEPROM	9.9 seconds	6096.3 µC during 2.2 sec
Internal OTP	8.6 seconds	-

9 How to handle the Crystal trimming and Bluetooth address into an external memory

In this chapter, the steps needed to write/read values (non-volatile parameters) into/from an external memory (I2C or SPI) are described. Non-volatile parameters can be for instance Bluetooth address, XTAL trim value. Reading the non-volatile parameters should be defined in the application layer. This is done by calling a specific function to read out the value stored at a specific address. SmartSnippets is equipped with a function to write a non-volatile parameter into external memory. Note that the explanation for an external SPI memory case only is mentioned in the example in the next section. A similar procedure can be used for an external I2C memory.

9.1 External SPI flash memory case

9.1.1 Reading operation:

It is assumed that the user knows how to handle the SPI drivers from the Peripheral Drivers [3]. The SPI driver must be included in your application.
The function which has to be used to read data at a specific address from the SPI memory is:
uint32_t spi_flash_read_data (uint8_t *rd_data_ptr, uint32_t address, uint32_t size);

Below is an example which describes how to use this function:


```
uint8_t rd_data[10];   // The data read from the address given will be stored in this buffer
uint32_t address=0x10; // The data will be read from the address 0x10 of the SPI flash
uint32_t size=10;      // 10 bytes are going to be read
uint32_t read_byte;    // Returns how many bytes have been read

read_byte = spi_flash_read_data(rd_data, address, size);
```

Figure 18: Read bytes from external SPI memory

9.1.2 Writing operation:

9.1.2.1 Using the SmartSnippets CLI

All the information/syntaxes about the CLI can be found in the **HELP** tab in the SmartSnippets GUI or by executing **Smartsnippets –help** in the CLI. In this example, the SPI connections are SCK (P0_0), CS (P0_3), MISO (P0_5) and MOSI (P0_6). However, if another pin assignment has to be used, some modifications have to be done in the flash programmer project (see chapter 6.1).

Step 1, in order to use the CLI, a binary file has to be downloaded into the SRAM of the DA14580 beforehand. There are two different bin files as shown below according to the way that the DA14580 is going to be connected by the PC (via UART or JTAG) using the CLI.

- *when using UART: flash_programmer.bin*

- *when using JTAG: jtag_programmer.bin*

Step 2, open the CLI by pushing the Shift button and right click on the ‘bin’ folder of the SmartSnippets and select Open command window here as follow:

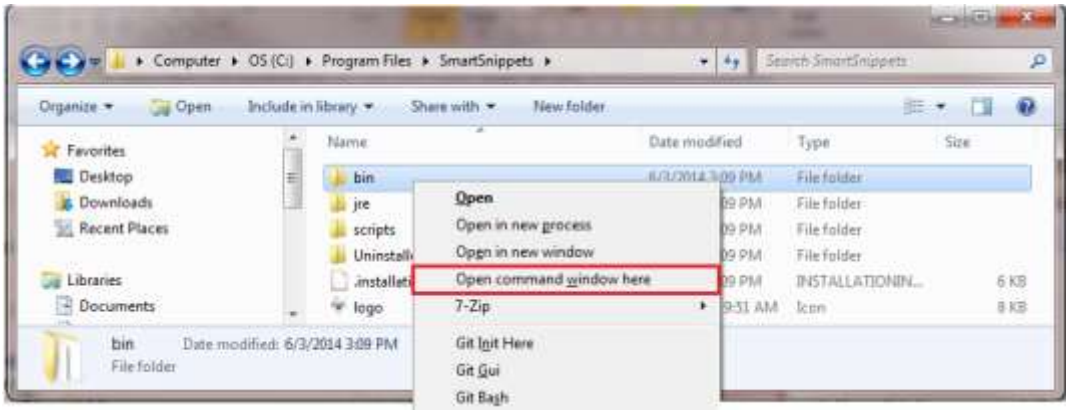


Figure 19: Describe how to open the CLI of SmartSnippets

Example: in order to write e.g. value 0x1347 (example of a bluetooth device address) at the address 0x93, the following command line can be written:

SmartSnippets.exe -type spi -chip DA14580-01 -jtag 228202458 -cmd write_field -offset 0x93 -data 1347 -firmware "D:\SmartSnippets\resources\jtag_programmer.bin"

The answers should be as follow:

```
Found SWD-DP with ID 0x0BB11477
FPUnit: 4 code (BP) slots and 0 literal slots
Found Cortex-M0 r0p0, Little endian.
BTLE device DA14580 selected.
Using default GPIO pin Id: P1_2.
Using default baudrate: 57600 Bd.
Burned 2 bytes to address 0x00093.
```

Figure 20: Answer from the DA14580 after receiving a command

9.1.2.2 Using the SmartSnippets GUI

From the Memory Header/NVDS programmer tab as shown below, it is possible to write a specific data (i.e Bluetooth address, crystal trimming) into the external memory.

Figure 21: Memory header tab

Then, a text file has to be created in order to specify at which address a value has to be stored. In this text file, three parameters must be added and separated with tabulation:

- ✓ Size (bytes)
- ✓ Type
- ✓ Parameter
- ✓ Description

Here is an example:

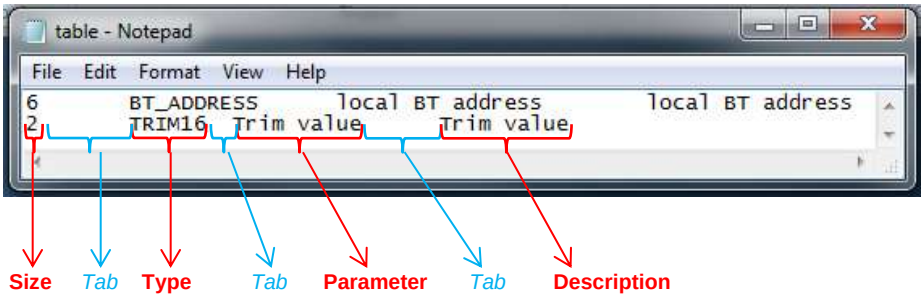


Figure 22: Text file format to write a customer header

Then, the text file has to be browsed and both address (Memory offset in hexadecimal) and data (Value) must be written manually in the GUI as follow:

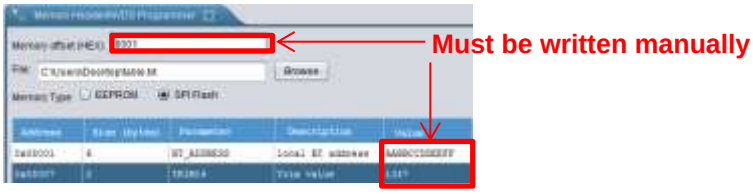


Figure 23: Memory header tab fulfilled

At this point, the content of the header can be burnt into the external memory.

10 List of supported FLASH/EEPROM memories

The memories in Table 7 are currently supported by DIALOG.

Table 6: Memories supported

Memory type	Name	Memory size	Page size
SPI	M25P16	2 MB	256B
SPI	GD25VQ41B	512 kB	256B
SPI	GD25VQ21B	256 kB	256B
SPI	W25X20CL	256 kB	256B
SPI	AT25XE011	128 kB	256B
SPI	W25X10CL	128 kB	256B
SPI	MX25V1006E	128 kB	256B
I2C	M24M01-R	128 kB	256B

Table 7: Performance of the SPI memories supported

		W25X10CL winbond	MX25V1006E MXIC	AT25XE011 adesto TECHNOLOGIES
Supply voltage		2.3V – 3.6V	2.35V – 3.6V	1.65V – 3.6V
Deep power down current		1 µA	2 µA	0.4 µA
		Measurements done @ 3.0V		
OTP BLANK	READING 30.56kB	385 µC	312 µC	941 µC
	Time to read 30.56 kB	202.3 ms	205 ms	205 ms
	Average current	2 mA	1.5 mA	4.8 mA
Secondary bootloader burnt	READING 30.56kB	116.5 µC	86 µC	193 µC
	Time to read 30.56 kB	36.4 ms	36.4 ms	36.4 ms
	WRITING 30.56kB	1 128.5 µC	1350 µC	5 038 µC
	Time to write 30.56 kB	291 ms	286 ms	500 ms

OTP BLANK	Peak current	9.5 mA	13.5 mA	10.25 mA

11 Layout view of WLCSP + USON Flash

The following layout proposal for a DA14580 solution with an external Windbond USON SPI Flash is shown. The overall size is 10.5 mm by 8 mm (0.413 inch by 0.315 inch). It includes the size of the USON memory which is 2x3mm. The schematic can be found in appendix 12.2.

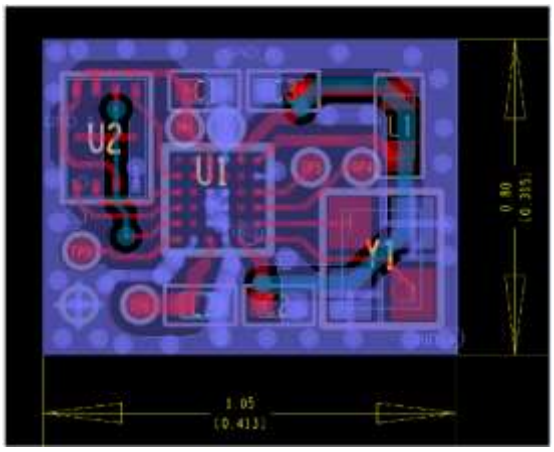


Figure 24: Layout of the DA1450 + external SPI memory

12 Revision history

Revision	Date	Description
1.0	23-June-2014	Initial version.
1.1	29-September-2014	Another SPI memory (AT25XE11) is supported (see Chapter 10)
1.2	16-October-2014	Another SPI memory (MX25V1006EWG) is supported (see Chapter 10)
1.3	21-November-2014	Standard function to read data from external SPI memory (see Chapter 9)
1.4	01-December-2014	Correction of reference from MX25V1006EWG to MX25V1006E
1.5	16-January-2015	Correction for the Writing operation measurement of the MX25V1006E
1.6	27-January-2015	The power supply of the external memory cannot be driven from the DA14580 via a FET. The schematics have been removed.
1.7	19-February-2015	7.2 sections: The secondary bootloader supports the I2C memory too. The speed of the I2C remains the same.
1.8	10-March-2015	Figure 23 updated
1.9	04-May-2015	Table 7 updated. Added GD25VQ41B & GD25VQ41B memories.
2.0	18-May-2015	Table 7 updated. Added M25P16 memory.
2.1	08-May-2016	Remove the section about the EXPERT board
2.2	18-Jan-2022	Updated logo, disclaimer, copyright.

Status definitions

Status	Definition
APPROVED	The content of this document has been approved for publication.

RoHS Compliance

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