JANUARY 21, 2014 V1.2

AN-B-003
DA14580 Software Patching over the Air (SPotA)

Abstract

The DA14580 is intended for Bluetooth applications that have a duty cycle during which the system only briefly transmits data and is in deep sleep mode for most of the time. The DA14580 contains embedded SW in the ROM (Bluetooth Smart protocol functions) and the One-Time-Programmable (OTP) memory (Bluetooth Smart profile and application). Amending this code on a small or large scale can be quite a challenge. The system based on the DA14580 might not be accessible in the field. Moreover, even if the system is accessible, a special procedure and infrastructure are required to make software patches (see Application Note AN-B-002). This application note describes a method that allows a software patch to be applied to an existing system over the air.

Table of Contents

1.0 Introduction

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 a 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. This is quite a flexible mechanism that allows radically changing code or replacing parameter values with a granularity of a single byte.

The programming of the patching requires physical access to the system based on the DA14580 (using a JTAG or UART interface) and manually programming the OTP with the patch code, i.e. a sequence of steps that requires the system to be carried into the lab.

A proprietary profile enables SW patching over the air. Patching code is downloaded using the Bluetooth Smart link, stored into the internal RAM or an external non-volatile memory and is eventually executed as described in AN-B-002. There are of course certain constraints with respect to the physical storage of the patching code. Furthermore, the system should be aware of the patching storage configuration in advance. Software Patching over the Air (SPotA) is described in this document in adequate detail for application developers.

1.1 TERMS AND ABBREVIATIONS

EEPROM	Electrically Eraseable Programmable Read Only Memory
GPIO	General Purpose Input Output
JTAG	Joint Test Action Group (test interface)
OTP	One Time Programmable (memory)
(S)RAM	(Static) Random Access Memory
ROM	Read Only Memory
SPOTA	Software Patching Over The Air
SW	SoftWare
UART	Universal Asynchronous Receiver Transmitter
UUID	Universal Unique ID

1.2 REFERENCES

1. DA14580, Data sheet Dialog Semiconductor
2. AN-B-002, Application and ROM code patching

1.3 HISTORY

May 9, 2013 Initial version

December 30, 2013 Added SPOTA abort command in Table 1 and SPOTA status encoding.

January 21, 2014 Corrected Figure 2.

2.0 SPotA Profile Architecture

2.1 CONFIGURATION

2.1.1 Roles

The SPOTA Profile defines 2 roles:

a. The “SPOTA Initiator” is the endpoint which transmits the patch payload.

b. The “SPOTA Receiver” is the endpoint which receives and applies the patch payload.

2.2 SERVICES AND ATTRIBUTES

The SPOTA profile realizes a single service with a number of characteristics required to be written by the initiator. The overall Attributes Database structure is shown in the following figure:

Handle	Type (Hex)	Type (Define)	Value	GATT Server Permissions	Notes
0x00	0x2800	GATT_PRIMARY_SERVICE_UUID	SPOTA UUID (UUID_0)	R	Start of SPOPTA service
0x01	0x2803	GATT_CHARACTER_UUID	0x0A/0x12 (Read/Write Permissions) handle 0x02 UUID UUID_1	R/W	SPOTA Memory Device characteristic declaration
0x02	UUID_1	SPOTA_MEM_DEV_UUID	4 Bytes **Byte 3 (Most Significant):** 0x00=SysRAM, 0x01=RetRAM, 0x02=I2C EEPROM, 0x03=SPI FLASH **Byte [2:0]**: PATCH_BASE_ADDRESS	R/W	Physical memory device info. Defines what is the target physical memory to be used for the Patch storage as well as the actuall base address of the Patch area within this memory.
0x03	0x2901	GATT_CHAR_USER_DESCRIPTION_UUID	"Mem Dev: B3=Mem, B[2:0]=Addr"	R	User Description
0x04	0x2803	GATT_CHARACTER_UUID	0x0A/0x12 (Read/Write Permissions) handle 0x05 UUID_2	R/W	SPOTA Memory Device characteristic declaration
0x05	UUID_2	SPOTA_GPIO_MAP_UUID	4 Bytes **Byte 3**: Dev. Address (MSB) / MISO **Byte 2**: Dev. Address / MOSI GPIO **Byte 1**: SCL/ CS GPIO position **Byte 0**: SDA/ SCK GPIO position	R/W	**Byte Mapping:** Bits[3:0]= Pin Number Bits[7:4]= Port Number i.e. 0x26 = Port2, Pin 6 => P2_6 pin Device Address applies only I2C EEPROMs
0x06	0x2901	GATT_CHAR_USER_DESCRIPTION_UUID	"GPIO Map: 1 Byte per pin mapping"	R	User Description
0x07	0x2803	GATT_CHARACTER_UUID	0x0A (Read Permissions) handle 0x08 UUID_3	R	SPOTA Memory Information characteristic declaration
0x08	UUID_3	SPOTA_MEM_INFO_UUID	4 Bytes **Byte[3:2] (Most Significant)**: Num_of_Patches **Byte[1:0]**: Entire_Patch_Length (32b words)	R	Memory information for the initiator. Displayes the current number of Patches already applied, and the entire Patch area length in # of words
0x09	0x2901	GATT_CHAR_USER_DESCRIPTION_UUID	"Mem Info: B[3:2]=#Patches, B[1:0]=Entire_Length"	R	User Description
0x0A	0x2803	GATT_CHARACTER_UUID	0x0A/0x12 (Read/Write Permissions) handle 0x0B UUID_4	R/W	SPOTA Patch Length characteristic declaration
0x0B	UUID_4	SPOTA_PATCH_LEN_UUID	2 Bytes	R/W	Defines the length of the new patch to be applied
0x0C	0x2901	GATT_CHAR_USER_DESCRIPTION_UUID	"Patch_Len: 2 Bytes"	R	User Description
0x0D	0x2803	GATT_CHARACTER_UUID	0x0A/0x12 (Read/Write Permissions) handle 0x0E UUID_5	R/W	SPOTA Patch Data characteristic declaration
0x0E	UUID_5	SPOTA_PATCH_DATA_UUID	20 Bytes	R/W	20 Bytes of SPOTA data, word aligned. MSByte first
0x0F	0x2901	GATT_CHAR_USER_DESCRIPTION_UUID	"Patch_Data: 20 Bytes"	R	User Description
0x10	0x2803	GATT_CHARACTER_UUID	0x0A/0x12 (Read/Notify Permissions) handle 0x11 UUID_6	R/Notify	SPOTA Service Status characteristic declaration
0x11	UUID_6	SPOTA_SERV_STATUS_UUID	1 Byte	R	SPOTA Status
0x12	0x2901	GATT_CHAR_USER_DESCRIPTION_UUID	"Patch Status Val: 8 bits"	R	User Description
0x13	0x2902	GATT_CLIENT_CHAR_CFG_UUID	2 bytes	R/W	0x0000: Disable notifications (default) 0x0001: Enable notifications

Figure 1 SPOTA Attributes Database

The SPOTA profile defines one service only (in yellow), and six characteristics (in light blue). Their values are described in the rows following each characteristic (grey). The profile is described in general by the following fields:

1. **Handle**: a pointer to the structure describing the characteristic

2. **Type**: Each type is defined by the Universal Unique ID (UUID) number and the description. Custom UUIDs are 128 bits long.
3. The **Value** of the specific Service/Characteristic.
4. The **GATT Server Permissions**. Defines if the characteristic's value can be read or written by the Client.

Each characteristic value is followed by a protocol defined GATT_CHAR_USER_DESCRIPTION which basically prints a text message for user notification.

The characteristics of the profile have a UUID which is designated by UUID_x within Figure 1 and are described in detail in the following sections.

2.2.1 SPOTA_MEM_DEV

This characteristic defines the actual physical devices where the Patch will stored upon reception over the air. This must be written by the SPOTA initiator. It contains 4 bytes. The encoding of these 4 bytes is depicted in the following table:

Any valid Byte 3 value, as defined in Table 1, will enable or disable (Byte 3=0xFF) the SPOTA mode of the system accordingly. During SPOTA mode, Extended or Deep Sleep modes are disabled.

Table 1: SPOTA_MEM_DEV definition

Byte	Description
3	This is the Most Significant Byte. Values: **0x00**: Patch is stored in System RAM **0x01**: Patch is stored in Retention RAM **0x02**: Patch is stored in I2C EEPROM **0x03**: Patch is stored in SPI FLASH **0x04-0xFE**: Reserved **0xFF**: SPOTA abort command. Return to normal application.
2	Byte #2 of the base address of the Patch in the external Non Volatile memory (SPI FLASH)
1	Byte #1 of the base address of the Patch in the external Non Volatile memory (SPI FLASH or I2C EEPROM)
0	Byte #0 of the base address of the Patch in the external Non Volatile memory (SPI FLASH or I2C EEPROM)

Note: Only SPI FLASH devices need 3 bytes for the Address. I2C EEPROMs only need 2 bytes.

2.2.2 SPOTA_GPIO_MAP

This characteristic defines the mapping of the interfaces on various GPIO pins. This is defined by the SPOTA initiator so that the system knows which pins are connected to the external Non Volatile memory. The reason of the Initiator instructing the Receiver of the GPIO mapping is that the low level functions that will take care of storing and executing the patch, reside in the ROM and are totally configurable.

This characteristic value contains 4 bytes. The meaning of each byte is depicted in the following table:

Table 2: SPOTA_GPIO_MAP definition

Byte	Description
3	This byte can be: 1. Byte #2 of **Device Address (MSB)** if I2C EEPROM 2. **MISO** position on the I/O ports if SPI FLASH
2	This byte can be: 1. Byte #1 of **Device Address**, if I2C EEPROM 2. **MOSI** position on the I/O ports if SPI FLASH
1	This byte can be: 1. **SCL** position on the GPIO ports if I2C EPROM 2. **CS** position on the I/O ports if SPI FLASH
0	This byte can be: 1. **SDA** position on the GPIO ports if I2C EPROM 2. **SCK** position on the I/O ports if SPI FLASH

Every byte is clearly indicating which I/O pin is used for the specific interface signal. Bits[3:0] defines the Pin Number while Bits[7:4] defines the Port Number. For example, a byte value of 0x26 means that the specific interface signal (e.g. MOSI) is mapped on Port 2 and pin 6 (i.e. P2_6).

2.2.3 SPOTA_MEM_INFO

This characteristic provides information to the SPOTA Initiator about the already applied number of Patches as well as the overall Patch area length as a number of 32-bit words. It can be read by the Initiator and contains 4 bytes which are described in the table below:

Table 3: SPOTA_MEM_INFO definition

Byte	Description
3-2	This value defines the number of Patches that have been applied to this specific device to date.
1-0	This value defines the entire size of the Patch related area in 32-bit words residing either in an external Non Volatile memory or in the System/Retention RAM.

2.2.4 SPOTA_PATCH_LEN

This characteristic defines the length of the new Patch which is to be applied during the current SPOTA session. It must be written from the Initiator to the Receiver and contains 2 bytes. The length is defined by this value in bytes.

Note: The length value should be 32-bit word aligned i.e. always be a multiple of 4.

2.2.5 SPOTA_PATCH_DATA

This characteristic contains 20 bytes which are the actual Patch payload data. This data is written by the Initiator to the Receiver. The Initiator should send the data in the following order:

Most significant word first, least significant byte first.

For example, if the patching data are 0x00077000, 0x12345678 then the Initiator should send:

0x00, 0x70, 0x07, 0x00, 0x78, 0x56, 0x34, 0x12

Note: The Initiator always sends the least significant byte first in all cases.

2.2.6 SPOTA_SERV_STATUS

This characteristic has two attributes. The first is the service status indicator, which is read-only and consists of 1 byte. The second is the protocol defined GATT_CLIENT_CHAR_CFG which can be written, consists of 2 bytes and enables the service status to be sent as an indication or a notification.

The attribute values are explained in the following tables:

Table 4: SPOTA_SERV_STATUS definition

Value	Description
0x00	**Reserved**
0x01	**SPOTAR_SRV_STARTED**. Valid memory device has been configured by initiator. While in this mode, system is kept at active mode.
0x02	**SPOTAR_CMP_OK**. SPOTA process completed successfully.
0x03	**SPOTAR_SRV_EXIT.** Forced exit of SPOTAR service. See Table 1.
0x04	**SPOTAR_CRC_ERR.** Patch Data CRC mismatch.
0x05	**SPOTAR_PATCH_LEN_ERR.** Received patch Length not equal to PATCH_LEN characteristic value.
0x06	**SPOTAR_EXT_MEM_ERR.** External Memory Error. Writing to external device failed.

Table 4: SPOTA_SERV_STATUS definition

Value	Description
0x07	**SPOTAR_INT_MEM_ERR.** Internal Memory Error. Not enough internal memory space for patch.
0x08	**SPOTAR_INVAL_MEM_TYPE.** Invalid memory device.
0x09	**SPOTAR_APP_ERROR.** Application error.
0x0A to 0xFF	**Reserved**

Table 5: GATT CLIENT CHAR CFG definition

Value	Description
0x0001	**Enable Notifications**. The SPOTA_SERV_STATUS value will be sent as a notification.
0x0002	**Enable Indications**. The SPOTA_SERV_STATUS value will be sent as an indication.

Note: The difference between notification and indication is that the indication requires an acknowledgement from the Client. For further information please refer to the BLE specification (Vol. 3, Part G, Section 4.10, 4.11).

3.0 ROM functions

This section describes the low level firmware functions used for the implementation of the SPOTA flow. These functions are stored in the ROM and can be called from the application if SPOTA is supported by the final product. The functions are presented in the following tables:

Table 6: Get_Patching_SPOTA_Length function

Name	get_patching_spota_length	
Number of Arguments	2	
	Type	**Description**
Argument #1	32-bit word	*mem_dev*: defines the physical memory as described in Table 1

Table 6: Get_Patching_SPOTA_Length function

Argument #2	32-bit word	*gpio_map*: defines the mapping of the interface signals to I/O pins as described at Table 2
Returns	32-bit word	One word encoded as in Table 3

Example.

WORD get_patching_spota_length(0x020000F0, 0xA2A12225);

is translated into the following text.

Memory Device: 0x02, i.e. an external I2C EEPROM

Patch Base Address: 0xF0, i.e. Patches are placed at this address onwards.

Device Address: 0xA2A1.

SCL location: 0x22, i.e. P2_2.

SDA location: 0x25, i.e. P2_5

Table 7: Exec_Patching_SPOTA function

Name	exec_patching_spota	
Number of Arguments	4	
	Type	**Description**
Argument #1	32-bit word	*mem_dev*: defines the physical memory as described in Table 1
Argument #2	32-bit word	*gpio_map*: defines the mapping of the interface signals to IO pins as described at Table 2
Argument #3	pointer	Points to the intermediate buffer address which will be used to temporary store the patch payload fetched from external Non Volatile memories. Only valid if I2C or SPI memories used.
Argument #4	16-bit half-word	*patch_length*: defines the size of the patch to be applied as described in Section 2.2.4
Returns	Void	

Example.

VOID exec_patching_spota(0x020000F0, 0xA2A12225, *buffer, 0x000A);

is translated into the following text.

Memory Device: 0x02, i.e. an external I2C EEPROM

Patch Base Address: 0xF0, i.e. Patches will be placed at this address onwards.

Device Address: 0xA2A1.

SCL location: 0x22, i.e. P2_2.

SDA location: 0x25, i.e. P2_5

**buffer*: points to the buffer where the Patch will be fetched in the System RAM before being executed

Patch Length: 0x000A, i.e. 10 32-bit words. This is the Patch size including the Patch Header and the Patch Payload.

Note: For further information on the structure of a Patch Area as well as the execution of the patch, please refer to the Application Note AN-B-002.pdf

4.0 SPotA Flow

The process of the SPOTA flow is illustrated in the following flow diagram:

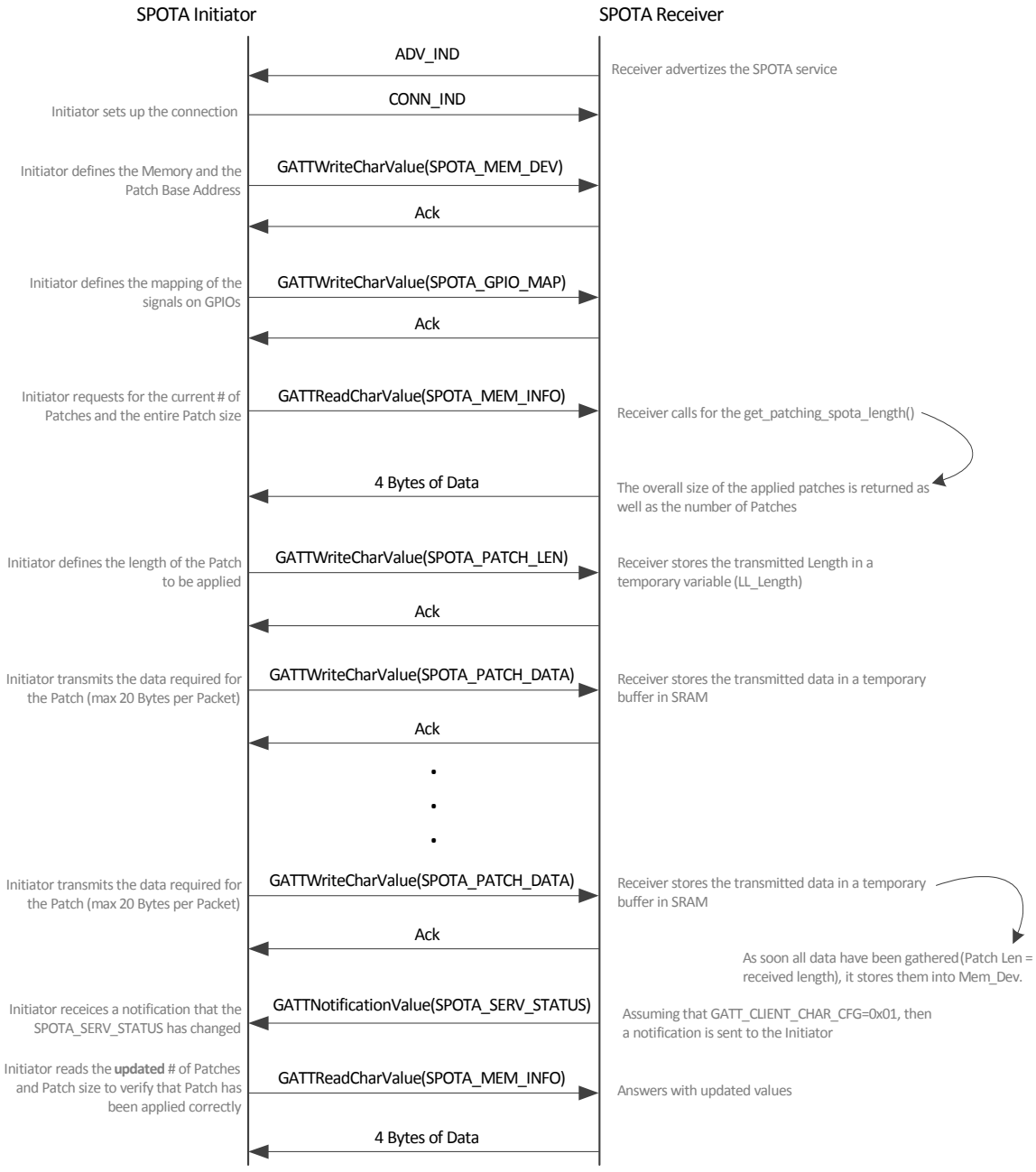

Figure 2 SPOTA flow diagram

5.0 UUIDs

The random 128-bit UUIDs assigned to the characteristics are summarized in the following table:

Table 8: Proprietary Characteristics UUIDs

UUID	Value
UUID_0	e9ef3514-5993-43bc-86c7-6b114cb061ee
UUID_1	8082caa8-41a6-4021-91c6-56f9b954cc34
UUID_2	724249f0-5ec3-4b5f-8804-42345af08651
UUID_3	6c53db25-47a1-45fe-a022-7c92fb334fd4
UUID_4	9d84b9a3-000c-49d8-9183-855b673fda31
UUID_5	457871e8-d516-4ca1-9116-57d0b17b9cb2
UUID_6	5f78df94-798c-46f5-990a-b3eb6a065c88