UM2330
User manual

ST8500 boot

Introduction

This user manual describes ST8500 bootloader functionalities and operations to be done for a correct device boot and the firmware images download.

The following sections are included in this document

- *Section 2 on page 4* covers the bootloader and the firmware image format
- *Section 3 on page 7* covers the boot from an external SPI Flash
- *Section 4 on page 10* covers the boot from an external host processor and the boot communication protocol
- *Section 5 on page 20* explains the usage of the image generator tool

This document does not include the description of the ST8500 device itself that can be found on www.st.com. Detailed information on the various PLC protocols running on the ST8500 and the device tools can be found in specific software packages, separately delivered under Software license agreement by contacting your local ST sales office.

Contents

1 **Document conventions 3**

2 **Description and bootloader 4**

2.1 Boot modes 4

2.2 Image format 5

2.3 User security configuration 6

3 **Boot from external SPI Flash 7**

3.1 Small configuration 8

3.2 Large configuration 9

4 **Boot from host interfaces 10**

4.1 SPI interface 10

4.2 UART interface 11

4.3 Command message flows 12

4.3.1 Setup of UART configuration and MIB request flow 13

4.3.2 Write image flow 13

4.4 Command message description 15

4.4.1 COM Init frame 15

4.4.2 MIB GET frame 16

4.4.3 IMG Init Frame 17

4.4.4 IMG write frame 18

4.4.5 IMG start frame 18

4.4.6 SYS reset frame 19

5 **Image generator tool 20**

6 **Revision history 22**

1 Document conventions

Table 1. List of abbreviations

Abbreviation	Description
Cortex	Cortex®-M4
AES	Advanced Encryption Standard
NVM	Non-volatile memory
OTP	One Time Programming
OTW	One time writable
PE	Protocol engine
RAM	Random access memory
ROM	Read-only memory
SoC	System on Chip
SPI	Serial peripheral interface
UART	Universal asynchronous receiver/transmitter

2 Description and bootloader

The ST8500 is a power line communication modem device supporting several narrow-band PLC standards. It is based on two different computing subsystems: an ARM® Cortex-M4 (called also PE) dedicated to the non-real time firmware and a real time engine (RTE) dedicated to the real time firmware.

The ST8500 boot procedure is managed by the Cortex-M4 core. The bootloader code is stored in an embedded ROM during the silicon manufacturing process. The Cortex-M4 core starts executing the instructions present in the embedded ROM after the power on reset (POR) or after a system reset.

The basic function implemented by the boot process is to decrypt, validate, load images from various boot sources into ST8500 RAMs and to transfer control to the target cores running the images. Once the control is transferred to the loaded code, it is no more possible to run the bootloader code without performing a system reset.

2.1 Boot modes

The bootloader is able to support up to 8 different boot modes. The configuration is done through the 3 BOOTx pins. Only 4 boot modes are currently used as shown in *Table 2*.

Table 2. Boot modes

Boot2	Boot1	Boot0	Boot ID	Boot mode
0	0	0	0x0	Boot from UART host interface
0	0	1	0x1	Boot from SPI host interface
0	1	0	0x2	Boot from SPI external Flash (large configuration)
0	1	1	0x3	Boot from SPI external Flash (small configuration)
1	0	0	0x4	Reserved
1	0	1	0x5	Reserved
1	1	0	0x6	Reserved
1	1	1	0x7	Reserved

The bootloader mandates to download the RTE image first and then the Cortex-M4 image. If only the Cortex-M4 image is provided, the RTE is not booted. This means that the RTE is not enabled and that it is not possible to load any further RTE image without performing a complete reset to deal with a new boot procedure.

The ST8500 is able to load new or different images only after the POR or a complete reset.

2.2 Image format

The bootloader is able to load the firmware into the proper core only if it is provided in an image format. The image contains a clear-text header, one or more section headers and a payload that is the binary firmware. The format of the image is the same for all the boot modes.

The firmware can be divided into different sections of payload to be stored at different RAM addresses. The whole firmware payload can be either clear-text or encrypted.

The image payload and all the header data following the authentication tag are authenticated.

The format of the firmware image is detailed in *Table 3*.

Table 3. Firmware image format description

-	Data type	Values/meaning	Data Bytes offset	Data Bytes size
Clear-text FIRMWARE IMAGE HEADER	IMAGE_TYPE	FW_PE_IMAGE = 0x0000F1F1	0	4
		FW_RTE_IMAGE = 0x0000A3A3	-	-
	VALIDITY STATUS	Validity of the image can change without impact on the authentication tag. Managed by the application during the FW upgrade on the external Flash (0x2 and 0x3 BOOT MODES only).	4	4
	APPLICATION PARAMETERS	Application specific data (32 bytes), can change without impact on the authentication code.	8	32
	FIRMWARE SIZE	The size of the firmware payload in bytes.	40	4
	SECURITY MODE	Specifies the image encryption mode, the encryption B seed (PE only) or no encryption.	44	4
	AES IV	The AES initialization vector	48	16
	AUTHENTICATION TAG	The authentication tag, generated during AES authentication/encryption.	64	16
	FIRMWARE ENTRY ADDRESS	Address where the execution of the code contained in this image shall start.	80	4
	NUMBER OF SECTIONS	The number of sections (n) of the firmware image. The number of sections has to be greater than zero and minor than 25.	84	4
Clear-text FIRMWARE IMAGE n-th SECTION HEADER	FIRMWARE DESTINATION ADDRESS	The internal load address of the section image	88 + 8 * (n - 1)	4
	FIRMWARE n-th SECTION SIZE	The size of the n-th section in firmware payload	92 + 8 * (n - 1)	4
HEADER PADDING	Optional			8
FIRMWARE PAYLOAD	-	Variable length firmware payload (one or more sections), either clear-text or AES encrypted	-	Variable length
PAYLOAD PADDING	-			

The “number of sections (n)” valid range is 1 to 24.

The image must be padded (up to 8 bytes) in order to keep the authenticated header size multiple of an AES block (16 bytes).

2.3 User security configuration

The ST8500 allows the user to define security rules that the bootloader applies at the startup by a stored configuration into an OTP memory area. Refer to the ST8500 datasheet for details on the OTP data structure and functions. The OTP area includes the “PE key” (Cortex-M4F image decryption key) used for the (optional) decryption and authentication of the PE firmware image. The image is encrypted and authenticated using the AES-GCM algorithm.

3 Boot from external SPI Flash

Booting from the SPI Flash requires having at least one firmware image already present in the external Flash. The SPI1 peripheral is used to control the SPI Flash at a nominal frequency of 12 MHz.

Two kinds of Flash size are supported

- Small size Flash
- Large size Flash

The firmware images are searched at fixed offsets according to the memory organization as showed in *Figure 1* and *Figure 2*.

The external SPI Flash can contain two versions of each image to support upgrading and rollback to the previous firmware version in case of firmware malfunctioning. The selection of the images to be loaded is based on the “Validity” field inside the image header. The meaning of the field “Validity” is detailed in *Table 4*.

Table 4. Firmware image validity coding

Validity binary value (4 bits)	Image validity status
1110b0	Updated image, not yet functionally validated from the application code
1100b0	First choice image, validated from the application code
1000b0	Second choice image, validated but superseded by a newer one
0000b0	Functionally invalid image, invalidated from the application code
All other values	Invalid value, corrupted image - header check will fail

In particular, if two images satisfy the integrity checks based on the header and authentication code, the selection process is defined by following rules

- A new, not validated image needs validation (or invalidation), so it will get the priority
- If no new images exist, a first-choice image is selected
- If no new or first-choice image exists, a second-choice image is selected
- If no new, first-choice or second-choice image exists, a functionally invalid image is selected and an error condition is flagged
- If the validity flag value is the same for both images, then the image #1 (PE#1 and/or RTE#1) has the preference

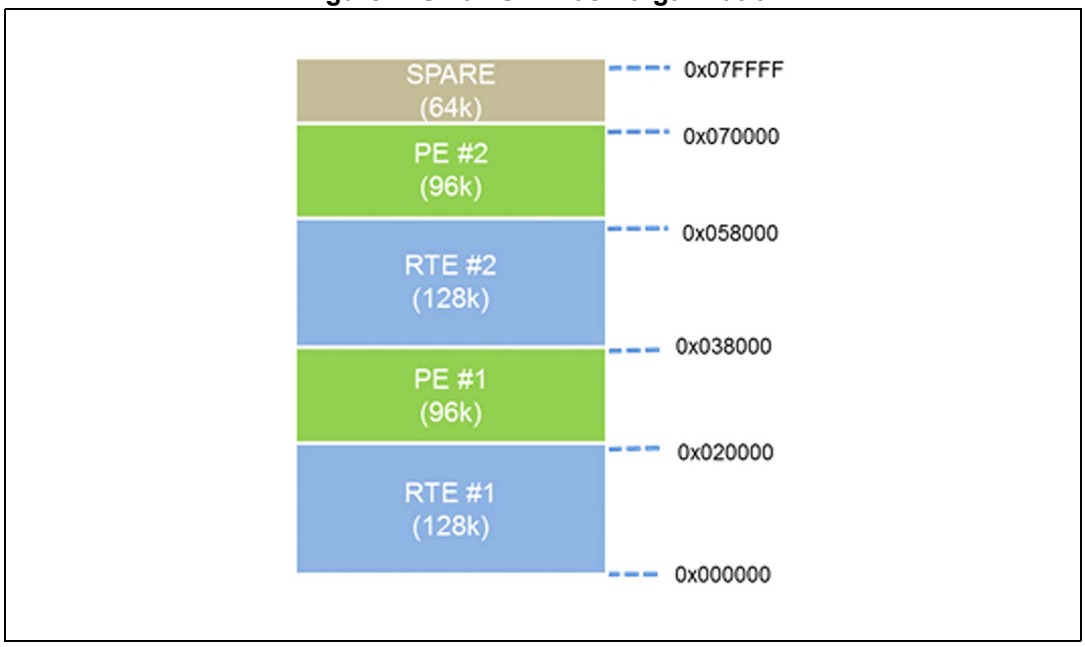
According to the coding of *Table 4*, the image with the higher valid value of the validity flag is selected. If only one image satisfies the integrity checks based on the header and authentication code, this image is loaded regardless to the value of the field “Validity”.

If no image satisfies the integrity checks based on the authentication code, the boot does not load the firmware for the related computing subsystem regardless to the value of the field “Validity” and enters in the “Boot from UART HOST interface” mode. The image validity is managed by the application code and not by the bootloader code.

3.1 Small configuration

In the small configuration, the minimum SPI Flash size is 512 kB. The boot loader uses only the first 512 kB ignoring the other part of the SPI Flash if bigger. *Figure 1* shows the Flash organization for the small size SPI Flash.

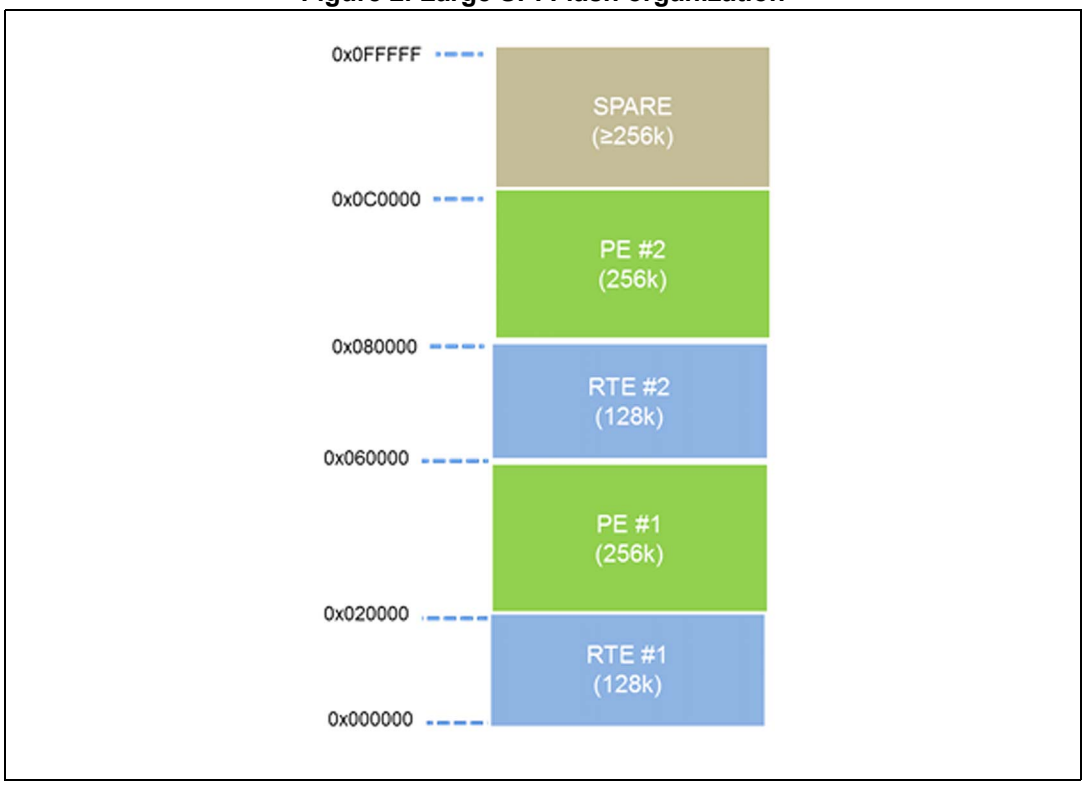
Figure 1. Small SPI Flash organization



3.2 Large configuration

If the large configuration is selected, the minimum SPI Flash size must be 1 MB. The boot loader uses only the first 1 MB ignoring the other part of the SPI Flash if bigger that is left to application purposes. The large configuration is the preferred one. *Figure 2* shows the Flash organization for the large size SPI Flash.

Figure 2. Large SPI Flash organization



4 Boot from host interfaces

Boot modes from the host interface require an external host controller connected to the ST8500 via UART or SPI. The host controller is able to download the firmware images according to the protocol specified in the following paragraphs. The host controller acts as a master, while the ST8500 is a slave.

The frame format and the protocol commands are common to both UART and SPI interfaces. The frame format specified in *Table 5* is the same for both requests and responses. All fields are in the little endian format.

It is always the master (host controller) that initiates the communication by sending a request. The ST8500 answers with a response using the same command ID (CMD_ID).

Table 5. Request/response message format

Request/response message format									
Synchro field		Control field		State field		Message payload field			Check
0x16	0x16	CMD_ID	MSG_LEN	MODE = 0xFE	STATE	MSG0	...	MSGLEN-1	CRC
2 Bytes		1 Byte	2 Bytes	1 Byte	4 Bytes	N Bytes			2 Bytes

4.1 SPI interface

The SPI0 peripheral is used as the interface with the following configuration

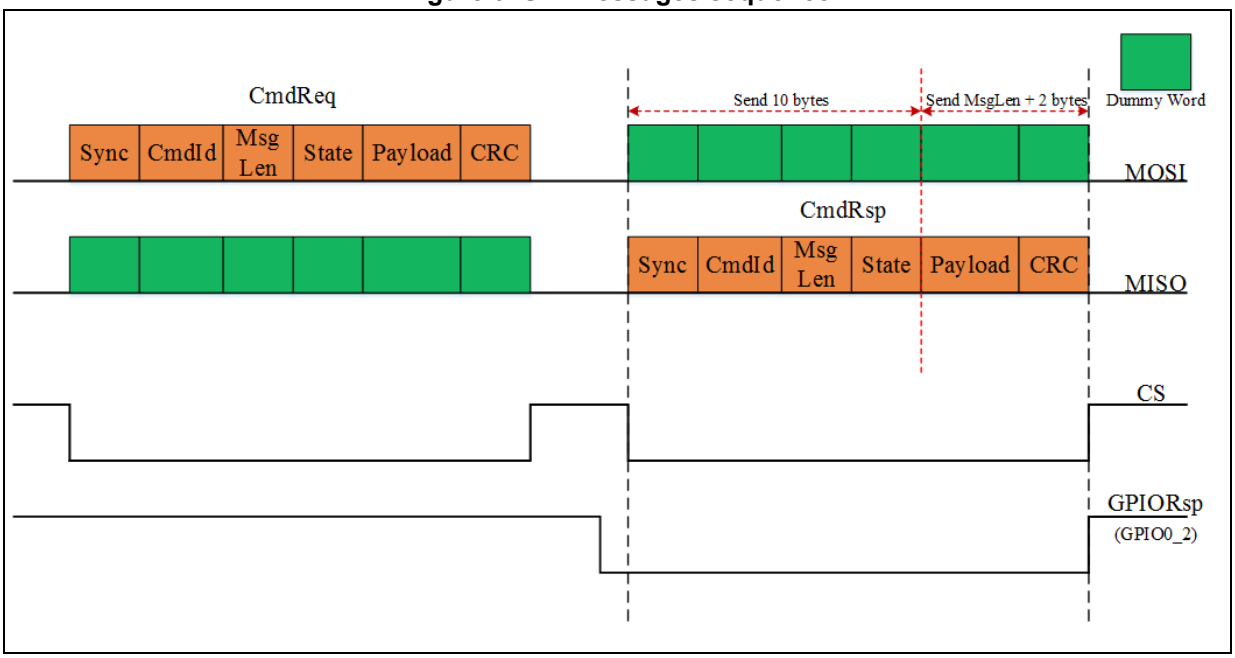
- CPHA = 0 (the first clock transition is the first data capture edge)
- CPOL = 0 (clock to 0 when idle)
- Maximum frequency is 15 MHz

Since the SPI is a synchronous protocol and the ST8500 is the slave of this communication, the response management needs and extra signal to indicate to the host that a response is available. The ST8500 signals the availability of the response setting GPIORsp low that is mapped on GPIO0_2.

The master sends the request commands message on the MOSI line ignoring the received bytes on the MISO line. When the request has been processed, the ST8500 signals the availability of the response by setting the GPIORsp (GPIO0_2) line in a low state. At this point the master sends dummy bytes on the MOSI line in order to receive the response. When all the response bytes are transmitted, the ST8500 changes the GPIORsp (GPIO0_2) line status back to the original status. At this point, the master may issue a new command - a response transmission cycle may start again.

Figure 3 shows a complete request - response sequence.

Figure 3. SPI messages sequence



4.2 UART interface

The UART0 is used as an interface with the following initial configuration

- 9600 baud
- 1 start bit
- 1 stop bit
- No parity
- No flow control

Since the UART is an asynchronous interface, both the host controller and the ST8500 may send messages without any additional hardware handshake. The UART configuration can be modified by a specific command to be sent at the beginning.

4.3 Command message flows

The following paragraphs describe the host interface messages and sequence flows.

In particular, the command IDs and the STATE values are described respectively in *Table 6* and *Table 7*. The common fields like SYNCH, MODE and CRC are not reported.

Table 6. Command ID description

CMD_ID		Description
System commands		
0x01	COM_Init	To change the UART configuration
0x02	SYS_Reset	To perform a system reset
Image download commands		
0x11	IMG_Init	To signal the start of an image download
0x12	IMG_Write	To download an image block
0x13	IMG_Start	To signal the end of the image and the request of the start the related computing subsystem
Management information base		
0x21	MIB_Get	To request the value of a specific MIB

Table 7. State description

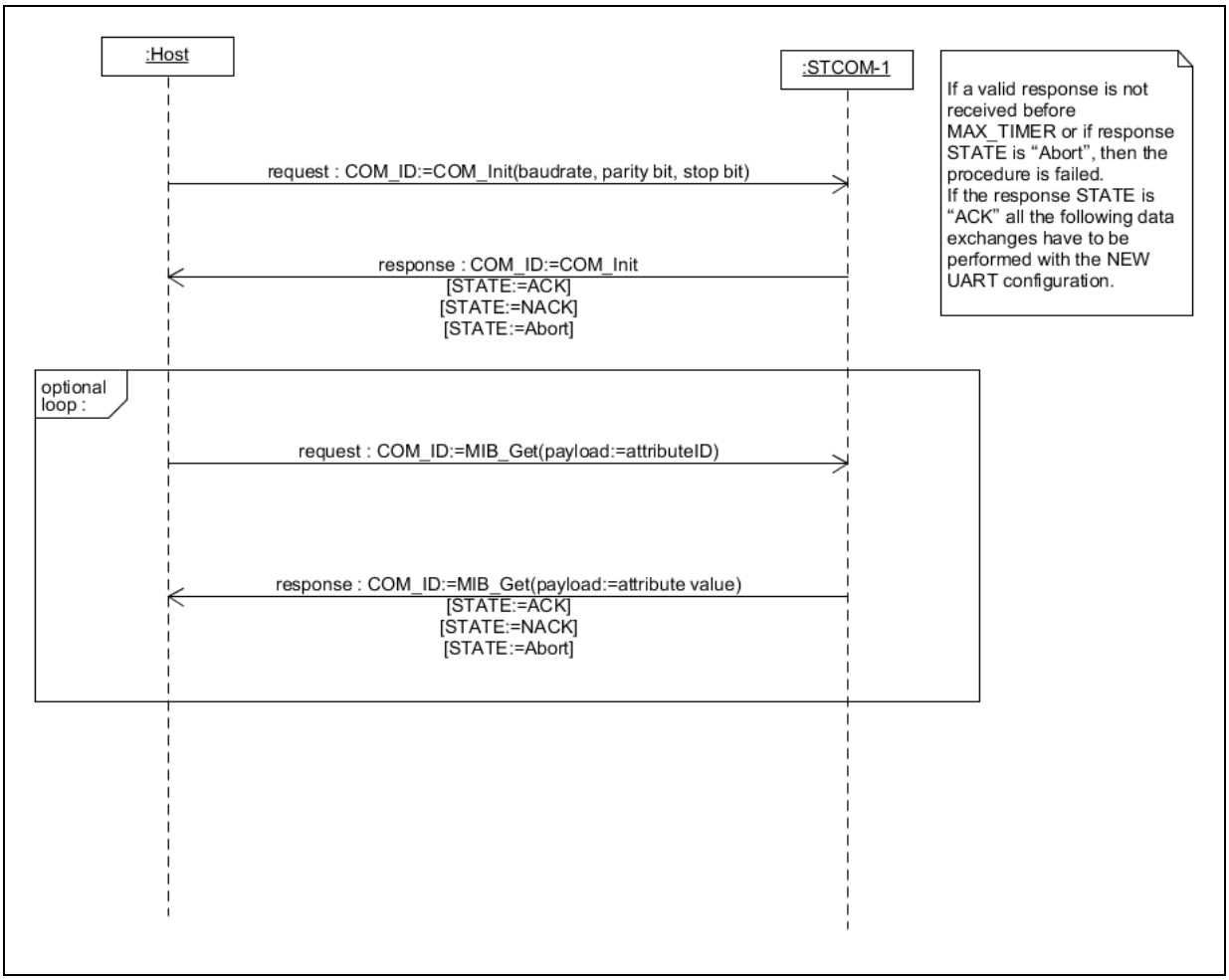
State				Description
0x00	0x00	0x00	0x00	ACK - request received and correctly performed
0x01	0x00	0x00	0x00	NACK - request received with errors, ask for retransmission
0x02	0x00	0x00	0x00	Abort - request correctly received, but the related operation can't be performed

4.3.1 Setup of UART configuration and MIB request flow

The following flowchart in *Figure 4* shows the starting communication from the host to the ST8500.

The first step shows the UART configuration setup (i.e. in order to increase the UART speed), then the flowchart shows the optional loop for multiple MIB get requests.

Figure 4. Setup of UART configuration and MIB request flow



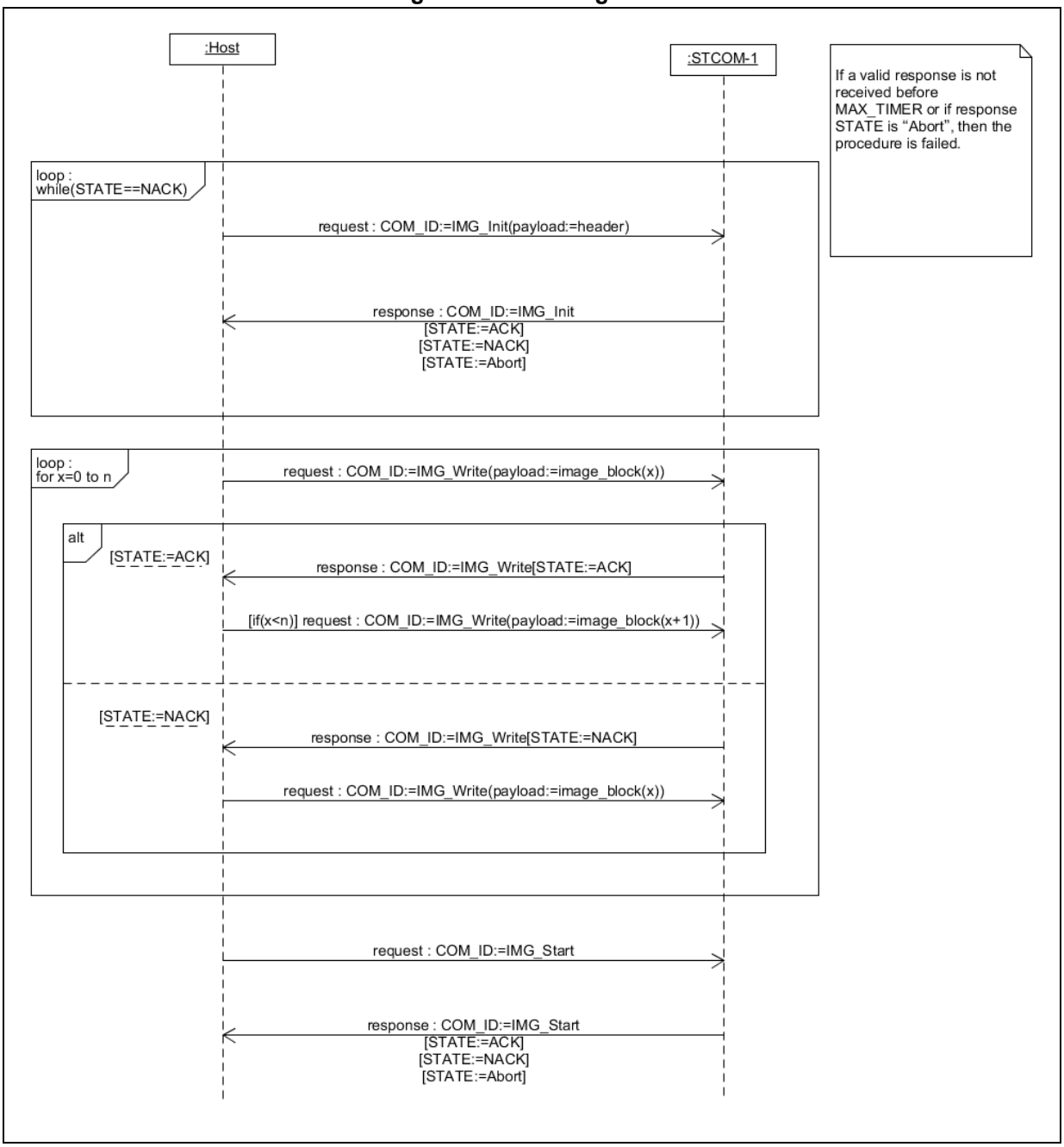
4.3.2 Write image flow

The following flowchart in *Figure 5* shows the write image procedure.

On the first step the host sends a request to the ST8500 to start the download. This message must include only the plaintext header of the binary image. After that, a loop is used to write the whole image payload. At the end, the host controller sends the start message asking the core to boot the loaded image.

The procedure is the same for both RTE and PE images, and it must be performed first for the RTE image download and then for the PE image download. If only the PE procedure is done, the boot process ends without loading the RTE image.

Figure 5. Write image flow



4.4 Command message description

4.4.1 COM Init frame

This command is used to initialize the communication between the host and the ST8500 specifying a new UART configuration: baud rate, parity bit and stop bit. The possible responses are ACK, NACK and Abort.

Table 8. COM_Init.Request command frame

COM_Init.Request command frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 … 4	MSG_LEN	6	Length of message payload
10 …13	Baudrate	See *Table 9*	Baudrate
14	Parity	0….2	Parity: 0x00 = None, 0x01 = Odd, 0x02 = Even
15	Stop	0….2	Stop bits: 0x1= 1 bit, 0x2 = 2 bits

Table 9. Baudrate configuration

COM_Init PAYLOAD - baudrate
9600
14400
19200
38400
57600
115200
230400
921600

Table 10. COM_Init.Response frame

COM_Init.Response frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 … 4	MSG_LEN	0	Length of message payload
6 … 9	STATE	See *Table 9*	STATE: 0x00 = ACK, 0x01 = NACK, 0x02 = Abort

4.4.2 MIB GET frame

This command is used to get the value of a specific ST8500 MIB (see *Table 13*). The possible responses are ACK with the asked value, NACK and Abort.

Table 11. MIB_GET.Request command frame

MIB_GET.Request command frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 …4	MSG_LEN	1	Length of message payload
10	AttributeId	-	The AttributeId (see *Table 7*)

Table 12. MIB_GET.Response frame

MIB_GET.Response frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 … 4	MSG_LEN	2 …16	Length of message payload
6 … 9	STATE	See *Table 7*	STATE: 0x00 = ACK, 0x01 = NACK, 0x02 = Abort
10 …10 + (MSG_LEN-1)	AttributeValue	-	Attribute value

The available MIB are listed in *Table 13*.

Table 13. MIB list

ID	MIB	Size (bytes)	Access
0x00	RTE boot status (see *Table 14*)	2	RO
0x01	PE boot status (see *Table 14*)	2	RO
0x02	Security bits	2	RO
0x03	BOOT mode	2	RO
0x04	BOOT version	4	RO
0x05	Reserved	4	RO
0x06	Reserved	4	RO
0x07	EUI64	8	RO
0x08	Reserved	8	RO
0x09	Reserved	16	RO
0x0A	Reserved	16	RO

RTE/PE boot status bits are listed in *Table 14*.

Table 14. RTE/PE status bits description

Bit	Status value
0	Image transfer initiated
1	Image transfer completed
2	Image transfer successful
3	RTE load error
4	PE load error
5	RTE IMG_Init error
6	PE IMG_Init error
7	RTE IMG_Write error
8	PE IMG_Write error
9	No valid PE image error
10	No valid RTE image error
11	OTP read error
12	FW image size error
13	Image start address error
14	RTE start timeout error
15	Functional invalid image error
16	Loaded image number (only for SPI Flash boot modes)

4.4.3 IMG init frame

This command is used to start the loading of an image on the ST8500. The message payload includes the plaintext header of the binary image. The possible responses are ACK, NACK and Abort.

Table 15. IMG_Init.Request command frame

IMG_Init.Request command frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 ...4	MSG_LEN	Global header length + length of section headers	Length of message payload: image header size in bytes
10 ...10 + MSG_LEN	Img Header	-	Image header

Table 16. IMG_Init.Response frame

IMG_Init.Response Frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 … 4	MSG_LEN	0	Length of message payload
6 … 9	STATE	See *Table 7*	STATE: 0x00 = ACK, 0x01 = NACK, 0x02 = Abort

4.4.4 IMG write frame

This command is used to load an image payload block on the ST8500. The block size must be a multiple of 16 bytes (a single AES block). The maximum allowed size is 1 kB. The possible responses are ACK, NACK and Abort.

Table 17. IMG_Write.Request command frame

IMG_Write.Request command frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 … 4	MSG_LEN	16xN N = 1 … Max	Length of message payload
10 …10 + MSG_LEN	Img Data Block	-	Image data block

Table 18. IMG_Write.Response frame

IMG_Write.Response frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 … 4	MSG_LEN	0	Length of message payload
6 … 9	STATE	See *Table 7*	STATE: 0x00 = ACK, 0x01 = NACK, 0x02 = Abort

4.4.5 IMG start frame

This command is used to send to the ST8500 the restart signal after the image loading. The possible responses are ACK, NACK and Abort.

Table 19. IMG_Start.Request frame

IMG_Start.Request command frame		
Label	**Value**	**Description**
CMD_ID	See *Table 6*	Command ID
MSG_LEN	0	Length of message payload

Table 20. IMG_Start.Response frame

IMG_Start.Response frame		
Label	**Value**	**Description**
CMD_ID	See *Table 6*	Command ID
MSG_LEN	0	Length of message payload
STATE	ACK, NACK, Abort	STATE: 0x00 = ACK, 0x01 = NACK, 0x02 = Abort

4.4.6 SYS reset frame

This command is used to send a reset request to the ST8500. Possible responses are ACK, NACK and Abort.

Table 21. SYS_Reset.Request command frame

SYS_Reset.Request command frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 … 4	MSG_LEN	0	Length of message payload

Table 22. SYS_Reset.Response frame

SYS_Reset.Response frame			
Byte	**Label**	**Value**	**Description**
2	CMD_ID	See *Table 6*	Command ID
3 … 4	MSG_LEN	0	Length of message payload
6 … 9	STATE	See *Table 7*	STATE: 0x00 = ACK, 0x01 = NACK, 0x02 = Abort

5 Image generator tool

The image generator is a tool that receives as inputs a configuration file and some sections binaries (at least one section) in order to produce as an output either a PE or an RTE firmware image composed by the specified sections, arranged and encrypted according to the configuration file. The sections sizes must be aligned to 128 bits, and the image generator tool provides for this alignment by adding pad bits if needed.

The configuration file is an *.xml file, formed by the following tags

```
<header image_type="processor_type" validity="validity_value"
entry="entry_address" fw_encryption_status="encryption_type"
fw_encryption_seed="seed_value">
<application_data value0="app_data[0]" value1=" app_data [1]" value2="
app_data [2]" value3=" app_data [3]" value4=" app_data [4]" value5="
app_data [5]" value6=" app_data [6]" value7=" app_data [7]">]">
</application_data>
   <aes_iv value0="iv[0]" value1="iv[1]" value2="iv[2]" value3="iv[3]">
</aes_iv>
   <aes_key value0="key[0]" value1="key[1]" value2="key[2]" value3="key[3]"
value4="key[4]" value5="key[5]" value6="key[6]" value7="key[7]"> </aes_key>
</header>
<section address="sect_0_address" filename="section_0_filename">
</section>
<section address="sect_1_address" filename="section_1_filename">
</section>
<section address="sect_2_address" filename="section_2_filename">
</section>
<section address="sect_3_address" filename="section_3_filename">
</section>
...
<section address="sect_n_address" filename="section_n_filename">
</section>
```

Table 23 specifies the meaning and the allowed values for every tag fields.

Table 23. Image generator fields description

Configuration field	Meaning	Allowed values	Notes
image_type	Specifies the image type to be generated	cpu rte	-
validity	Validity value	Unsigned 32-bit integer greater than 0	-
entry	Entry address from where the execution of the fw image shall start	32-bit address	The entry address must be 64-bit aligned (see sect_n_address)
fw_encryption_status	Specifies if the payload is encrypted	cleartext aes_256	An authentication tag is always generated for the whole image, also if clear-text.
fw_encryption_seed	Reserved	0	Must be set to “0”
Aes_iv value *i*, 0≤i≤3	AES-GCM Initial vector's i-th word, input for the encryption algorithm	32-bit integer	-
Aes_key value *i*, 0≤i≤7	AES-GCM key's i-th word, input for the encryption algorithm	32-bit integer	-
Section address	Memory address where the first section's word is written	32-bit address	The section address must be 64-bit aligned
Section filename	Filename containing the input hexadecimal bytes of the section	-	-

6 Revision history

Table 24. Document revision history

Date	Revision	Changes
13-Dec-2017	1	Initial release.