UM0516
User manual

Windows API for STMicroelectronics microcontroller bootloaders

Introduction

This user manual describes an application programming interface (API) for the STMicroelectronics bootloader protocol. The API is composed of two DLLs: `STBLLIB.DLL` and `Files.DLL`.

The purpose of the API is to make it easy to develop Microsoft® Windows® applications that load microcontroller code from a host PC via a serial communications interface into the embedded Flash memory of an STMicroelectronics microcontroller system. The API is compatible with Windows 98/2000/XP/Vista/7.

It interfaces with the bootloader functions embedded in the system memory of STMicroelectronics microcontrollers.

Contents

List of tables

1 Library contents

This library consists of two DLLs, `STBLLIB.DLL` and `Files.DLL`:

- **`STBLLIB.DLL`** provides a function set to execute basic system memory bootloader commands.
- **`Files.DLL`** provides a function set to manage Intel hexadecimal, Motorola S19 and binary files.

This user manual describes the available functions in detail.

2 STBLLIB library

2.1 Constants

Request codes are used to identify current requests. The supported commands are listed in *Table 1* below. Error codes are shown in *Table 2*. Each command is further described in this section.

Table 1. Embedded bootloader commands

Name	Value	AN2606	AN2430	Description
GET_CMD	0x00	X		Gets the version and the allowed commands supported by the current version of the bootloader
GET_VER_ROPS_CMD	0x01	X	X	Gets the bootloader version and the read protection status of the Flash memory
GET_ID_CMD	0x02	X		Gets the chip ID
READ_CMD	0x11	X	X	Reads up to 256 bytes of memory starting from an address specified by the user
GO_CMD	0x21	X	X	Jumps to an address specified by the user to execute (a loaded) code
WRITE_CMD	0x31	X	X	Write up to 256 bytes to the RAM or the Flash memory starting from an address specified by the user
ERASE_CMD	0x43	X	X	Erases from one to all the NVM sectors
WRITE_PROTECT_CMD	0x63	X	X	Enables the write protection for some sectors (permanently in the STR75x)
WRITE_TEMP_UNPROTECT_CMD	0x71		X	Disables the write protection temporarily for all NVM sectors
WRITE_PERM_UNPROTECT_CMD	0x73	X		Disables the write protection for all Flash memory sectors
READOUT_PROTECT_CMD	0x82	X	X	Enables the readout protection (permanently in the STR75x)
READOUT_TEMP_UNPROTECT_CMD	0x91		X	Disables the readout protection temporarily
READOUT_PERM_UNPROTECT_CMD	0x92	X	X	Disables the readout protection (permanently in the STR75x)

Table 2. Error codes

Name	Value	Description
SUCCESS	0x00	No error.
ERROR_OFFSET	0x00	Error offset.
COM_ERROR_OFFSET	ERROR_OFFSET + 0x00	Serial communication error offset.
NO_CON_AVAILABLE	COM_ERROR_OFFSET + 0x01	No serial communication opened.
COM_ALREADY_OPENED	COM_ERROR_OFFSET + 0x02	The serial communication over the selected port is already opened.
CANT_OPEN_COM	COM_ERROR_OFFSET + 0x03	Error while opening the serial communication.
SEND_FAIL	COM_ERROR_OFFSET + 0x04	Error while sending data over the serial port.
READ_FAIL	COM_ERROR_OFFSET + 0x05	Error while reading data from the serial port.
SYS_MEM_ERROR_OFFSET	ERROR_OFFSET + 0x10	System memory error offset.
CANT_INIT_BL	SYS_MEM_ERROR_OFFSET + 0x01	No response from the target, the system memory bootloader was not started.
UNREOGNIZED_DEVICE	SYS_MEM_ERROR_OFFSET + 0x02	The device was not recognized.
CMD_NOT_ALLOWED	SYS_MEM_ERROR_OFFSET + 0x03	The command is not allowed.
CMD_FAIL	SYS_MEM_ERROR_OFFSET + 0x04	The command failed.
PROGRAM_ERROR_OFFSET	ERROR_OFFSET + 0x20	Program error offset.
INPUT_PARAMS_ERROR	PROGRAM_ERROR_OFFSET + 0x01	One of the input parameters is out of range.
INPUT_PARAMS_MEMORY_ALLOCATION_ERROR	PROGRAM_ERROR_OFFSET + 0x02	One of the input parameters is not allocated.

2.2 Types

The command record is provided in *Table 3*.

Table 3. Command record

Offset	Field	Size	Value	Description
0	GET_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
2	GET_VER_ROPS_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
4	GET_ID_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
6	READ_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
8	GO_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
10	WRITE_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
12	ERASE_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
14	WRITE_PROTECT_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
16	WRITE_TEMP_UNPROTECT_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
18	WRITE_PERM_UNPROTECT_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
20	READOUT_PROTECT_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
22	READOUT_TEMP_UNPROTECT_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).
24	READOUT_PERM_UNPROTECT_CMD	2	BOOL	Command supported (TRUE), not support ed (FALSE).

2.3 APIs

2.3.1 Introduction

The aim of this manual being to get started easily with minimum development effort, this section only fully describes the most important and common API functions. Please refer to the self-documented source codes and headers inside the “Src” subfolder for the latest updates and the descriptions of all implemented functions.

2.3.2 COM_Close

`BYTE COM_Close()`

Closes the current serial connection.

Table 4. COM_Close description

	Field	Type	Description
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.3 COM_Open

`BYTE COM_Open()`

Opens the selected serial port. Use `SetCOMSettings` before selecting and configuring the serial port to be used, otherwise default settings (serial communication over COM1 at 115 200 baud, 8 bits, with no parity and nbStopBit = 1) are applied.

Table 5. COM_Open description

	Field	Type	Description
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.4 COM_is_Open

`BOOL COM_is_Open()`

Gets the status of the selected serial port.

Table 6. COM_is_Open description

	Field	Type	Description
Output		BOOL	Returns TRUE if the serial communication is opened, FALSE otherwise

2.3.5 GetPaketSize

`BYTE GetPaketSize(LPBYTE size)`

Gets the maximum packet size to be transmitted to the device, the packet size is set to 0x100 as default and updated when calling *STBL_Init_BL* method.

Table 7. GetPacketSize description

	Field	Type	Description
Input	size	LPBYTE	Pointer to the maximum packet size to be transmitted to the device.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.6 GetAckValue

```
ACKS GetAckValue()
```

Gets the ACK value, returned by the device after each transmitted packet, the ACK value is set to ST79 as default and updated with the received value after calling the *STBL_Init_BL* function.

Table 8. GetAckValue description

	Field	Type	Description
Output		ACKS	UNDEFINED=0x00, ST75=0x75, ST79=0x79

2.3.7 STBL_DNLOAD

```
BYTE STBL_DNLOAD(DWORD Address, LPBYTE pData, DWORD Length,BOOL
bTruncateLeadFFForDnLoad)
```

Issues a Download request to the device.

Table 9. STBL_DNLOAD description

	Field	Type	Description
Input	Address	DWORD	Address to which the data are to be written.
	pData	LPBYTE	Pointer to the data buffer.
	Length	DWORD	Length of the data buffer in bytes.
	bTruncateLeadFFForDnLoad	BOOL	Set to TRUE to ignore FF packets.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.8 STBL_ERASE

```
BYTE STBL_ERASE(BYTE NbSectors, LPBYTE pSsectors)
```

Issues an Erase request to the device, the erase operation is performed on a sector/page basis.

Table 10. STBL_ERASE description

	Field	Type	Description
Input	NbSectors	BYTE	Number of sectors/pages to be erased.
	pSsectors	LPBYTE	Pointer to sector code buffer.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.9 STBL_GET

```
BYTE STBL_GET(LPBYTE Version, LPCommands pCmds)
```

Issues a Get request to the device to get the firmware version and the allowed commands.

Table 11. STBL_GET description

	Field	Type	Description
Input	Version	LPBYTE	Firmware version.
	pCmds	LPCommands	Pointer to the Supported command structure (see *Table 3: Command record*).
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.10 STBL_GET_ID

```
BYTE STBL_GET_ID(LPBYTE size, LPBYTE pID)
```

Issues a Get ID request to the device.

Table 12. STBL_GET_ID description

	Field	Type	Description
Input	size	LPBYTE	Size of the PID buffer in bytes.
	pID	LPBYTE	Pointer to the PID buffer.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.11 STBL_GET_VER_ROPS

```
BYTE STBL_GET_VER_ROPS(LPBYTE Version, LPBYTE ROPEnabled, LPBYTE
ROPDisabled)
```

Issues a Get Version and Read Out Protection status request to the device.

Table 13. STBL_GET_VER_ROPS description

	Field	Type	Description
Input	Version	LPBYTE	Firmware version.
	ROPEnabled	LPBYTE	Number of times the readout protection was enabled.
	ROPDisabled	LPBYTE	Number of times the readout protection was disabled.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.12 STBL_GO

```
BYTE STBL_GO(DWORD Address)
```

Issues a Go request to the device.

Table 14. STBL_GO description

	Field	Type	Description
Input	Address	DWORD	Start address of the code to be run.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.13 STBL_Init_BL

```
BYTE STBL_Init_BL()
```

Starts the system memory bootloader configuration by sending a 0x7F value. The ACK value is then updated.

Table 15. STBL_Init_BL description

	Field	Type	Description
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.14 STBL_READ

```
BYTE STBL_READ(DWORD Address, BYTE Size, LPBYTE pData)
```

Issues a Read request to the device. This function can read up to 256 bytes from the memory starting from the given address. To read a larger amount of data, use the *STBL_UPLOAD* function.

Table 16. STBL_READ description

	Field	Type	Description
Input	Address	DWORD	Start address from which data will be written.
	Size	BYTE	Size of the data in byte.
	pData	LPBYTE	Pointer to the data buffer.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.15 STBL_READOUT_PROTECT

```
BYTE STBL_READOUT_PROTECT()
```

Enables the memory readout protection (permanently in the case of the STR75x).

Table 17. STBL_READOUT_PROTECT description

	Field	Type	Description
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.16 STBL_READOUT_PERM_UNPROTECT

```
BYTE STBL_READOUT_PERM_UNPROTECT()
```

Disables the memory readout protection (permanently in the case of the STR75x).

Table 18. STBL_READOUT_PERM_UNPROTECT description

	Field	Type	Description
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.17 STBL_READOUT_TEMP_UNPROTECT

```
BYTE STBL_READOUT_TEMP_UNPROTECT()
```

Disables the memory readout protection temporarily.

Note: This command is not available in "STM32F101xx and STM32F103xx system memory boot mode" (AN2606).

Table 19. STBL_READOUT_TEMP_UNPROTECT description

	Field	Type	Description
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.18 STBL_UPLOAD

```
BYTE STBL_UPLOAD(DWORD Address, LPBYTE pData, DWORD Length)
```

Issues an Upload request to the device.

Table 20. STBL_UPLOAD description

	Field	Type	Description
Input	Address	DWORD	Address the data will be read from.
	pData	LPBYTE	Pointer to the data buffer.
	Length	DWORD	Length of the data buffer in bytes.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.19 STBL_VERIFY

```
BYTE STBL_VERIFY(DWORD Address, LPBYTE pData, DWORD Length,BOOL
bTruncateLeadFFForDnLoad)
```

Issues a Verify request to the device. the given data buffer will be compared to the data stored at the given memory address.

Table 21. STBL_VERIFY description

	Field	Type	Description
Input	Address	DWORD	Address from which data will be read.
	pData	LPBYTE	Pointer to the data buffer to be verified.
	Length	DWORD	Length of the data buffer in bytes.
	bTruncateLeadFFForDnLoad	BOOL	Set to TRUE to ignore FF packets.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.20 STBL_WRITE

```
BYTE STBL_WRITE(DWORD address, BYTE size, LPBYTE pData)
```

Issues a Write request to the device. This function can write up to 256 bytes in memory starting from the given address. To write a larger amount of data, use the *STBL_DNLOAD* function.

Table 22. STBL_WRITE description

	Field	Type	Description
Input	Address	DWORD	Start address, data will be read from.
	Size	BYTE	Size of the read data in byte.
	pData	LPBYTE	Pointer to the read data buffer.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.21 STBL_WRITE_PERM_UNPROTECT

```
BYTE STBL_WRITE_PERM_UNPROTECT()
```

Disables the write protection.

Note: This command is not available in the "STR75x SystemMemory boot mode" (AN2430)

Table 23. STBL_WRITE_PERM_UNPROTECT description

	Field	Type	Description
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.22 STBL_WRITE_PROTECT

```
BYTE STBL_WRITE_PROTECT(BYTE NbSectors, LPBYTE pSectors)
```

Enable write protection of given sectors/pages.

Table 24. STBL_WRITE_PROTECT description

	Field	Type	Description
Input	NbSectors	BYTE	Number of sectors to be write-protected.
	pSectors	LPBYTE	Pointer to sector code buffer.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.23 STBL_WRITE_TEMP_UNPROTECT

```
BYTE STBL_WRITE_TEMP_UNPROTECT()
```

Disables the write protection temporarily.

Note: This command is not available in the "STM32F101xx and STM32F103xx system memory boot mode" (AN2606).

Table 25. STBL_WRITE_TEMP_UNPROTECT description

	Field	Type	Description
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.24 SetCOMSettings

```
BYTE SetCOMSettings(int numPort, long speedInBaud, int nbBit, int
parity, float nbStopBit)
```

Configures the serial port.

Table 26. `SetCOMSettings` description

	Field	Type	Description
Input	numPort	int	Serial port number, the number should be in the range of the available port numbers in the used workstation.
	speedInBaud	long	Maximum rate, in bits per second (bps), at which you want data to be transmitted through the selected port. Usually, this is set to the maximum rate supported by the computer or device used for communication in the application.
	nbBit	int	Number of data bits you want to use for each character that is transmitted and received. The device used in the application should have the same setting as the one chosen here.
	parity	int	Type of error checking you want to use for the selected port. The device used for communication in the application must have the same setting as the one chosen here. **0: None** means that no parity bit is added to the data bits sent from this port. Disables error checking. **1: Odd** means that a parity bit is added if the number of 1's in the data bits has to be made odd. Enables error checking. **2: Even** means that a parity bit is set to 1 if the number of 1's in the data bits has to be made even. Enables error checking. **3: Mark** means that a parity bit is added, but it is always reset to 0. **4: Space** means that a parity bit is added, but it is always set to 1.
	nbStopBit	float	Time between each character being transmitted.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.25 `SetPacketSize`

```
BYTE SetPacketSize(BYTE size)
```

Sets the packet size to be transmitted during download or upload operations.

Table 27. `SetPacketSize` description

	Field	Type	Description
Input	size	BYTE	Size of the packet transmitted over the serial communication.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

2.3.26 SetTimeOut

```
BYTE SetTimeOut(DWORD vms)
```

Sets the timeout in milliseconds, specifying the period of time after which the read operation from the serial port is aborted.

Table 28. SetTimeOut description

	Field	Type	Description
Input	vms	DWORD	Period of time after which the read from the serial port is aborted.
Output		BYTE	SUCCESS, Error otherwise (see *Error codes.*).

3 Files library

3.1 Constants

3.1.1 Error codes

Table 29. Error codes

Name	Value	Description
FILES_ERROR_OFFSET	0x12346000	File access error offset.
FILES_NOERROR	0x12340000	No error.
FILES_UNABLETOOPENFILE	0x12346003	Unable to open file.
FILES_UNABLETOOPENTEMPFILE	0x12346004	Unable to open temporary file.
FILES_BADFORMAT	0x12346005	Bad format.
FILES_BADADDRESSRANGE	0x12346006	Bad address range.
FILES_BADPARAMETER	0x12346008	Bad parameter.
FILES_UNEXPECTEDERROR	0x1234600A	Unexpected error.
FILES_FILEGENERALERROR	0x1234600D	File general error.

3.2 Types

3.2.1 ELEMENT

Table 30. ELEMENT record

Offset	Field	Size	Value	Description
0	dwElementAddress	4	Address	Starting address of element data.
4	dwElementSize	4	Number	Data size in bytes.
8	Data	1		Data buffer.

3.2.2 MAPPINGSECTOR

Table 31. MAPPINGSECTOR record

Offset	Field	Size	Value	Description
0	Name	4	Pointer (char*)	Name of the sector
4	dwStartAddress	4	DWORD	Start address
8	dwAliasedAddress	4	DWORD	Aliased address
12	dwSectorIndex	4	DWORD	Index of the sector
16	dwSectorSize	4	DWORD	Size of the sector
20	bSectorType;	1	BYTE	Type of the sector
21	UseForOperation	2	BOOL	Equal to TRUE if the sector is used for the current operation.
23	UseForErase;	2	BOOL	Equal to TRUE if the sector is used for the Erase operation.
25	UseForUpload;	2	BOOL	Equal to TRUE if the sector is used for the Upload operation.
27	UseForWriteProtect	2	BOOL	Equal to TRUE if the sector is used for the Write operation.

3.2.3 MAPPING

Table 32. MAPPING record

Offset	Field	Size	Value	Description
0	nAlternate	1	BYTE	Index of the mapping list.
1	Name	260	Array of char	Name of the mapping list.
261	NbSectors	4	DWORD	Number of sectors in the mapping list.
265	pSectors	4	PMAPPINGSECTOR	Pointer to the first sector record.

3.3 APIs

3.3.1 FILES_CreateImage

`FILES_CreateImage(PHANDLE pHandle, BYTE nAlternate)`

Creates an empty Image for a given index number to a given handle pointer.

Table 33. FILES_CreateImage description

	Field	Type	Description
Input	pHandle	PHANDLE	Pointer to a handle.
	nAlternate	BYTE	Alternate number.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.2 FILES_CreateImageFromMapping

`FILES_CreateImageFromMapping (PHANDLE pHandle, PMAPPING pMapping)`

Creates an empty image according to the given mapping.

Table 34. FILES_CreateImageFromMapping description

	Field	Type	Description
Input	pHandle	PHANDLE	Pointer to a handle
	pMapping	PMAPPING	Pointer to a mapping
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.3 FILES_DestroyImage

`FILES_DestroyImage (PHANDLE pHandle)`

Destroys the given image and frees the allocated memory.

Table 35. FILES_DestroyImage description

	Field	Type	Description
Input	pHandle	PHANDLE	Pointer to the Image handle
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.4 FILES_DestroyImageElement

```
FILES_DestroyImageElement (HANDLE Handle, DWORD dwRank)
```

Removes the ELEMENT located on the dwRank index in the given image.

Table 36. FILES_DestroyImageElement description

	Field	Type	Description
	pHandle	PHANDLE	Image handle
Input	DWORD	dwRank	Element index
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.5 FILES_DuplicateImage

```
FILES_DuplicateImage (HANDLE hSource, PHANDLE pDest)
```

Creates a copy of the given hSource image in the pDest image.

Table 37. FILES_DuplicateImage description

	Field	Type	Description
Input	hSource	HANDLE	Original image handle
	pDest	PHANDLE	Pointer to the duplicated image handle
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.6 FILES_FilterImageForOperation

```
FILES_FilterImageForOperation (HANDLE Handle, PMAPPING pMapping,
DWORD Operation, BOOL bTruncateLeadFFForUpgrade)
```

Filters an image for the given operation. bTruncateLeadFFForUpgrade is used for UPGRADE and ERASE operations to remove all 256-byte packets that contain only FFh data, if needed.

Table 38. FILES_FilterImageForOperation description

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	pMapping	PMAPPING	Pointer to the mapping.
	Operation	DWORD	Operation code. The operation code can be set to the following values: OPERATION_UPLOAD (02h), OPERATION_ERASE (03h), OPERATION_DNLOAD (04h)
	bTruncateLeadFFForUpgrade	BOOL	Truncate if TRUE.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.7 FILES_GetImageAlternate

```
FILES_GetImageAlternate (HANDLE Handle, PBYTE pAlternate)
```

Gets the alternates setting stored in the given Image.

Table 39. FILES_GetImageAlternate description

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	pAlternat	PBYTE	Buffer to which the alternates are copied.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.8 FILES_GetImageElement

```
FILES_GetImageElement (HANDLE Handle, DWORD dwRank, PIMAGEELMENT
pElement)
```

Retrieves an element from the given image handle at the given dwRank index.

Table 40. FILES_GetImageElement description

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	dwRank	DWORD	Element index.
	pElement	PIMAGEELMENT	Pointer to the retrieved IMAGEELEMENT.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.9 FILES_GetImageName

```
FILES_GetImageName (HANDLE Handle, PSTR Name)
```

Gets the image name.

Table 41. FILES_GetImageName description

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	Name	PSTR	image name.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.10 FILES_GetImageNbElement

```
FILES_GetImageNbElement (HANDLE Handle, PDWORD pNbElements)
```

Gets the number of elements in the given image.

Table 42. FILES_GetImageNbElement description

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	pNbElements	PDWORD	Number of elements.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.11 FILES_ImageFromFile

```
FILES_ImageFromFile (PSTR pPathFile, PHANDLE pImage, BYTE
nAlternate)
```

Loads the image from the given S19 or Hex file specified by **pPathFile**. The file format is recognized by its extension (*.s19 or *.Hex).

Table 43. FILES_ImageFromFile description

	Field	Type	Description
Input	pPathFile	PSTR	Path of the file to be loaded, can be S19 or Hex file.
	pImage	PHANDLE	Pointer to an image handle to be created.
	nAlternate	BYTE	Index to be associated with the image.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.12 FILES_ImageToFile

```
FILES_ImageToFile (PSTR pPathFile, HANDLE Image)
```

Saves the given image handle to the S19 or Hex file specified by **pPathFile**. The file format is recognized by its extension (*.s19 or *.Hex).

Table 44. FILES_ImageToFile description

	Field	Type	Description
Input	pPathFile	PSTR	Path of the file, can be S19 or Hex file.
	Image	HANDLE	Pointer to image handle
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.13 FILES_SetImageElement

```
FILES_SetImageElement (HANDLE Handle, DWORD dwRank, BOOL bInsert,
ELEMENT Element)
```

Inserts or replaces an ELEMENT in the given image at the given index.

Table 45. FILES_SetImageElement description

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	dwRank	DWORD	Index at which the ELEMENT is inserted.
	bInsert	BOOL	Insert if TRUE. Replace if FALSE.
	Element	ELEMENT	Pointer to ELEMENT.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

3.3.14 FILES_SetImageName

```
FILES_SetImageName (HANDLE Handle, PSTR Name)
```

Sets the image name.

Table 46. FILES_SetImageName description

	Field	Type	Description
Input	Handle	HANDLE	Pointer to image handle.
	Name	PSTR	Image name.
Output		DWORD	SUCCESS, Error otherwise (see *Error codes*).

4 Revision history

Table 47. Document revision history

Date	Revision	Changes
05-Jun-2008	1	Initial release.
12-Nov-2009	2	*Introduction* modified. References to application notes removed from `STBLLIB.DLL` description *on page 5*. *Section 2.3.1: Introduction* added. Small text changes in function names in *Section 2.3: APIs*.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com