



DfuSe Application Programming Interface

Introduction

This document describes the available interfaces in the DfuSe library, composed of three DLLs: STDFU, STDFUPRT and STDFUFiles. The objective of the DfuSe library is to ease the development of PC Device firmware upgrade applications under Microsoft Windows (98SE/2000/XP/VISTA) connecting with STMicroelectronics targets supporting DFU mode as described in the Dfu STMicroelectronics Extension Specification.

Contents

1	Device Firmware Upgrade STMicroelectronics Extension	5
2	Library architecture	6
3	STDFU library (STDFU.DLL)	6
3.1	Library description	6
3.2	Constants	6
3.2.1	DFU error codes	6
3.2.2	Request codes	7
3.2.3	DFU State codes	8
3.2.4	SFU Status codes	9
3.3	Types	10
3.3.1	USB Device descriptor	10
3.3.2	USB Interface descriptor	11
3.3.3	USB Configuration descriptor	11
3.3.4	DFU Functional descriptor	12
3.3.5	DFU Status packet	13
3.4	APIs	14
3.4.1	STDFU_Abort	14
3.4.2	STDFU_Close	14
3.4.3	STDFU_Clrstatus	14
3.4.4	STDFU_Detach	15
3.4.5	STDFU_Dnload	15
3.4.6	STDFU_GetConfigurationDescriptor	15
3.4.7	STDFU_GetDFUDescriptor	16
3.4.8	STDFU_GetDeviceDescriptor	16
3.4.9	STDFU_GetInterfaceDescriptor	17
3.4.10	STDFU_GetNbOfAlternates	17
3.4.11	STDFU_GetNbOfConfigurations	17
3.4.12	STDFU_GetNbOfInterfaces	18
3.4.13	STDFU_GetStringDescriptor	18
3.4.14	STDFU_Getstate	18
3.4.15	STDFU_Getstatus	19
3.4.16	STDFU_Open	19

	3.4.17	STDFU_SelectCurrentConfiguration	19
	3.4.18	STDFU_Upload	20
4		STDFUPRT library (STDFUPRT.DLL)	21
	4.1	Library description	21
	4.2	Constants	21
	4.3	Types	21
	4.3.1	DFUIMAGEELEMENT	21
	4.3.2	MAPPINGSECTOR	21
	4.3.3	MAPPING	22
	4.3.4	DFUThreadContext	22
	4.4	APIs	23
	4.4.1	STDFUPRT_CreateMappingFromDevice	23
	4.4.2	STDFUPRT_DestroyMapping	23
	4.4.3	STDFUPRT_GetOperationStatus	23
	4.4.4	STDFUPRT_LaunchOperation	24
	4.4.5	STDFUPRT_StopOperation	24
5		STDFUFiles library (STDFUFiles.DLL)	25
	5.1	Library description	25
	5.2	Constants	25
	5.3	Types	26
	5.3.1	DFUPREFIX	26
	5.3.2	DFUSUFFIX	26
	5.3.3	TARGETPREFIX	27
	5.3.4	ELEMENT	27
	5.4	APIs	27
	5.4.1	STDFUFILES_AppendImageToDFUFile	27
	5.4.2	STDFUFILES_CloseDFUFile	27
	5.4.3	STDFUFILES_CreateImage	28
	5.4.4	STDFUFILES_CreateImageFromMapping	28
	5.4.5	STDFUFILES_CreateNewDFUFile	28
	5.4.6	STDFUFILES_DestroyImage	28
	5.4.7	STDFUFILES_DestroyImageElement	29
	5.4.8	STDFUFILES_DuplicateImage	29
	5.4.9	STDFUFILES_FilterImageForOperation	29

5.4.10	STDFUFILES_GetImageAlternate	29
5.4.11	STDFUFILES_GetImageElement	30
5.4.12	STDFUFILES_GetImageName	30
5.4.13	STDFUFILES_GetImageNbElement	30
5.4.14	STDFUFILES_ImageFromFile	30
5.4.15	STDFUFILES_ImageToFile	31
5.4.16	STDFUFILES_OpenExistingDFUFile	31
5.4.17	STDFUFILES_ReadImageFromDFUFile	31
5.4.18	STDFUFILES_SetImageElement	32
5.4.19	STDFUFILES_SetImageName	32
6	Document references	33
7	Revision history	34

1 **Device Firmware Upgrade STMicroelectronics Extension**

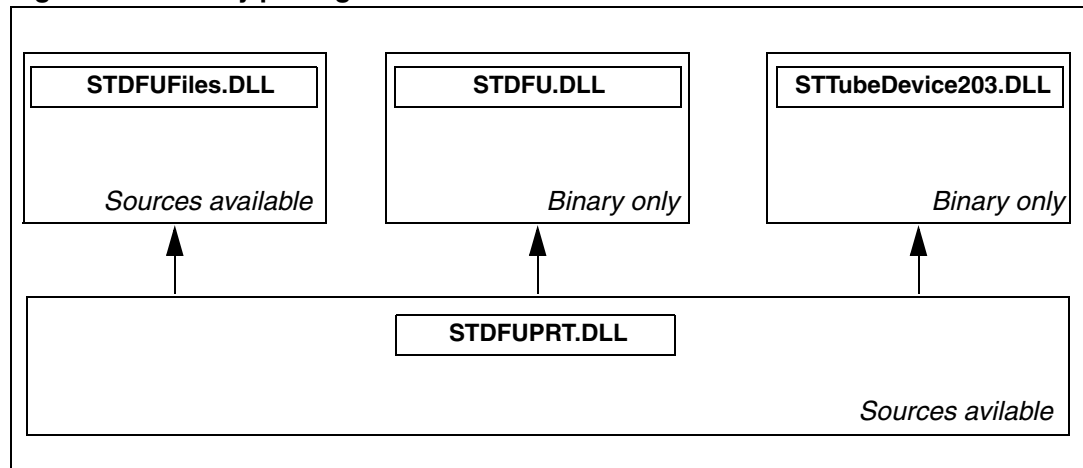
The reason for creating this DFU extension is that the standard device firmware upgrade protocol is too specialized in terms of protocol versus the target or the application.

This extension makes it easy to use DFU with all 8 or 32-bit microcontrollers, simply letting the PC side know the target memory mapping. In addition, the DFU file format has been updated, to be able to build DFU files from standard 8 or 32-bit s19, hex or bin formats. The result is a new revision of the standard DFU revision 1.1, called DfuSe for Device firmware upgrade STMicroelectronics Extension.

2 Library architecture

The complete library package consists of several layers. Some of these layers are given with sources, some others are only available as binaries. The following diagram presents the different layers.

Figure 1. Library package



3 STDFU library (STDFU.DLL)

3.1 Library description

The STDFU library implements basic DFU commands, as described in the standard DFU specification. Two sets of API are available: one to access device interfaces in both DFU and application mode and to get information about them, the second to issue standard DFU requests in DFU mode.

3.2 Constants

3.2.1 DFU error codes

Table 1. DFU error codes

Name	Value	Description
STDFU_ERROR_OFFSET	0x12340000	Error offset.
STDFU_NOERROR	0x12340000	No error or success.
STDFU_MEMORY	0x12340001	Cannot access memory.
STDFU_BADPARAMETER	0x12340002	Passed parameter is incorrect.
STDFU_NOTIMPLEMENTED	0x12340003	Function not implemented yet.

Table 1. DFU error codes

Name	Value	Description
STDFU_ENUMFINISHED	0x12340004	Device enumeration phase finished.
STDFU_OPENDRIVERERROR	0x12340005	The system cannot open the specified driver.
STDFU_ERRORDESCRIPTORBUILDING	0x12340006	Cannot get device descriptor or device configuration for the specified device.
STDFU_PIPECREATIONERROR	0x12340007	Cannot access endpoint.
STDFU_PIPERESETERROR	0x12340008	Pipe reset error.
STDFU_PIPEABORTERROR	0x12340009	Pipe abort error.
STDFU_STRINGDESCRIPTORERROR	0x1234000A	Cannot access string descriptor
STDFU_DRIVERISCLOSED	0x1234000B	Driver is closed.
STDFU_VENDOR_RQ_PB	0x1234000C	Cannot get the vendor id.
STDFU_ERRORWHILEREADING	0x1234000D	Error while reading on the pipe.
STDFU_ERRORBEFOREREADING	0x1234000E	Error before reading on the pipe.
STDFU_ERRORWHILEWRITING	0x1234000F	Error while writing on the pipe.
STDFU_ERRORBEFOREWRITING	0x12340010	Error before writing on the pipe.
STDFU_DEVICERESETERROR	0x12340011	Error while resetting the device.
STDFU_CANTUSEUNPLUGEVENT	0x12340012	Cannot use the unplug event.
STDFU_INCORRECTBUFFERSIZE	0x12340013	Insufficient Buffer size.
STDFU_DESCRIPTORNOTFOUND	0x12340014	Device descriptor not found.
STDFU_PIPESARECLOSED	0x12340015	The pipe is closed.
STDFU_PIPESAREOPEN	0x12340016	The pipe is already opened.
STDFU_TIMEOUTWAITINGFORRESET	0x12340017	reserved.

3.2.2 Request codes

The request codes are used to identify the current request sent by the active DFUThread context, the request code is stored in the CurrentRequest attribute.

Table 2.

Name	Value	Description
STDFU_RQ_GET_DEVICE_DESCRIPTOR	0x02000000	Get device descriptor request.
STDFU_RQ_GET_DFU_DESCRIPTOR	0x03000000	Get DFU descriptor request.
STDFU_RQ_GET_STRING_DESCRIPTOR	0x04000000	Get String descriptor request.
STDFU_RQ_GET_NB_OF_CONFIGURATIONS	0x05000000	Get number of configuration request.
STDFU_RQ_GET_CONFIGURATION_DESCRIPTOR	0x06000000	Get configuration descriptor request.

Table 2.

Name	Value	Description
STDFU_RQ_GET_NB_OF_INTERFACES	0x07000000	Get number of interfaces request.
STDFU_RQ_GET_NB_OF_ALTERNATES	0x08000000	Get number of alternats request.
STDFU_RQ_GET_INTERFACE_DESCRIPTOR	0x09000000	Get interface descriptor request.
STDFU_RQ_OPEN	0x0A000000	Open device request.
STDFU_RQ_CLOSE	0x0B000000	Close device request.
STDFU_RQ_DETACH	0x0C000000	DFU Detach request.
STDFU_RQ_DOWNLOAD	0x0D000000	DFU Download request.
STDFU_RQ_UPLOAD	0x0E000000	DFU Upload request.
STDFU_RQ_GET_STATUS	0x0F000000	DFU Get status request.
STDFU_RQ_CLR_STATUS	0x10000000	DFU Clear status request.
STDFU_RQ_GET_STATE	0x11000000	DFU Get state request.
STDFU_RQ_ABORT	0x12000000	DFU Abort request.
STDFU_RQ_SELECT_ALTERNATE	0x13000000	Select alternate/target request.
STDFU_RQ_AWAITINGPNPUNPLUGEVENT	0x14000000	Awaiting UNPLUG EVENT request.
STDFU_RQ_AWAITINGPNPPLUGEVENT	0x15000000	Awaiting PLUG EVENT request.
STDFU_RQ_IDENTIFYINGDEVICE	0x16000000	Identifying device after reenumeration request.

3.2.3 DFU State codes

The device state is defined as follows:

Table 3. DFU State codes

State	Value	Meaning
STATE_IDLE	0x00	Device is running its normal application.
STATE_DETACH	0x01	Device is running its normal application, has received the DFU_DETACH request, and is waiting for a USB reset.
STATE_DFU_IDLE	0x02	Device is operating in the DFU mode and is waiting for requests.
STATE_DFU_DOWNLOAD_SYNC	0x03	Device has received a block and is waiting for the host to solicit the status via DFU_GETSTATUS.
STATE_DFU_DOWNLOAD_BUZY	0x04	Device is programming a control-write block into its nonvolatile memories.

Table 3. DFU State codes

State	Value	Meaning
STATE_DFU_DOWNLOAD_IDLE	0x05	Device is processing a download operation. Expecting DFU_DNLOAD request.
STATE_DFU_MANIFEST_SYNC	0x06	Device has received the final block of firmware from the host and is waiting for receipt of DFU_GETSTATUS to begin the Manifestation phase, or device has completed the Manifestation phase and is waiting for receipt of DFU_GETSTATUS. (Devices that can enter this state after the Manifestation phase set <i>bmAttributes</i> bit <i>bitManifestationTolerant</i> to 1).
STATE_DFU_MANIFEST	0x07	Device is in the Manifestation phase. (Not all devices will be able to respond to DFU_GETSTATUS when in this state).
STATE_DFU_MANIFEST_WAIT_RESET	0x08	Device has programmed its memories and is waiting for a USB reset or a power on reset. (Devices that must enter this state clear <i>bitManifestationTolerant</i> to 0).
STATE_DFU_UPLOAD_IDLE	0x09	The device is processing an upload operation. Expecting DFU_UPLOAD request.
STATE_DFU_ERROR	0x0A	An error has occurred. Waiting for the DFU_CLRSTATUS request.
STATE_DFU_UPLOAD_SYNC	0x91	Reserved.
STATE_DFU_UPLOAD_BUZY	0x92	Reserved.

3.2.4 SFU Status codes

The device status is defined as follows:

Table 4. DFU Status codes

Name	Value	Suggested String
STATUS_OK	0x00	No error condition is present.
STATUS_errTARGET	0x01	File is not targeted for use by this device.
STATUS_errFILE	0x02	File is for this device but fails some vendor-specific verification test.
STATUS_errWRITE	0x03	Device is unable to write memory.
STATUS_errERASE	0x04	Memory erase function failed.
STATUS_errCHECK_ERASE	0x05	Memory erase check failed.
STATUS_errPROG	0x06	Program memory function failed.
STATUS_errVERIFY	0x07	Programmed memory failed verification.
STATUS_errADDRESS	0x08	Cannot program memory due to received address that is out of range.

Table 4. DFU Status codes

Name	Value	Suggested String
STATUS_errNOTDONE	0x09	Received DFU_DNLOAD with <i>wLength</i> = 0, but device does not think it has all of the data yet.
STATUS_errFIRMWARE	0x0A	Device's firmware is corrupt. It cannot return to run-time (non-DFU) operations.
STATUS_errVENDOR	0x0B	<i>iString</i> indicates a vendor-specific error.
STATUS_errUSBR	0x0C	Device detected unexpected USB reset signaling.
STATUS_errPOR	0x0D	Device detected unexpected power on reset.
STATUS_errUNKNOWN	0x0E	Something went wrong, but the device does not know what it was.
STATUS_errSTALLEDPKT	0x0F	Device stalled an unexpected request.

3.3 Types

3.3.1 USB Device descriptor

A device descriptor describes general information about a USB device. It includes information that applies globally to the device and all of the device's configurations. A USB device has only one device descriptor.

Table 5. USB Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	12h	Size of this descriptor, in bytes
1	bDescriptorType	1	01h	Device descriptor type
2	bcdUSB	2	0100h	USB specification release number in binary coded decimal
4	bDeviceClass	1	00h	See interface
5	bDeviceSubClass	1	00h	See interface
6	bDeviceProtocol	1	00h	See interface
7	bMaxPacketSize0	1	8,16,32,64	Maximum packet size for endpoint zero
8	idVendor	2	ID	Vendor ID. Assigned by the USB-IF
10	idProduct	2	ID	Product ID. Assigned by manufacturer
12	bcdDevice	2	BCD	Device release number in binary coded decimal
14	iManufacturer	1	Index	Index of string descriptor
15	iProduct	1	Index	Index of string descriptor
16	iSerialNumber	1	Index	Index of string descriptor
17	bNumConfigurations	1	01h	One configuration only for DFU

3.3.2 USB Interface descriptor

The interface descriptor describes a specific interface within a configuration. A configuration provides one or more interfaces, each with zero or more endpoint descriptors describing a unique set of endpoints within the configuration.

Table 6. USB Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	09h	Size of this descriptor, in bytes
1	bDescriptorType	1	04h	Interface descriptor type
2	bInterfaceNumber	1	00h	Number of this interface
3	bAlternateSetting	1	Number	Number of Alternate settings in this interface
4	bNumEndpoints	1	00h	Only the control pipe is used
5	bInterfaceClass	1	FEh	Application specific Class Code
6	bInterfaceSubClass	1	01h	Device Firmware Upgrade Code
7	bInterfaceProtocol	1	02h	DFU mode protocol
8	iInterface	1	Index	Index of string descriptor for this interface

3.3.3 USB Configuration descriptor

The configuration descriptor describes information about a specific device configuration. The descriptor contains a bConfiguration Value that, when used as a parameter to the SetConfiguration request, causes the device to assume the described configuration. This descriptor is identical to the standard configuration descriptor described in the USB specification v1.0, with the exception that the bNumInterfaces field must contain the value 01h.

Table 7. USB Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes
1	bDescriptorType	1	04h	Configuration descriptor type
2	wTotalLength	2	Number	Total length of data returned for this configuration. Includes the combined length of all descriptors (configuration, interface, endpoint, and class- or vendor-specific) returned for this configuration.
4	bNumInterfaces	1	01h	Number of interfaces supported by this configuration
5	bConfigurationValue	1	Number	Value to use as an argument to the SetConfiguration() request to select this configuration
6	iConfiguration	1	Number	Index of string descriptor describing this configuration

Table 7. USB Configuration Descriptor

Offset	Field	Size	Value	Description
7	bmAttributes	1	Number	<p>Configuration characteristics</p> <p>D7: Reserved (set to one)</p> <p>D6: Self-powered</p> <p>D5: Remote Wakeup</p> <p>D4...0: Reserved (reset to zero)</p> <p>D7 is reserved and must be set to one for historical reasons.</p> <p>A device configuration that uses power from the bus and a local source reports a non-zero value in bMaxPower to indicate the amount of bus power required and sets D6. The actual power source at runtime may be determined using the GetStatus(DEVICE) request. If a device configuration supports remote wakeup, D5 is set to one.</p>
8	MaxPower	1	Index	<p>Maximum power consumption of the USB device from the bus in this specific configuration when the device is fully operational. Expressed in 2 mA units (i.e., 50 = 100 mA).</p> <p>Note: A device configuration reports whether the configuration is bus-powered or selfpowered. Device status reports whether the device is currently self-powered. If a device is disconnected from its external power source, it updates device status to indicate that it is no longer self-powered.</p> <p>A device may not increase its power draw from the bus, when it loses its external power source, beyond the amount reported by its configuration. If a device can continue to operate when disconnected from its external power source, it continues to do so. If the device cannot continue to operate, it fails operations it can no longer support.</p> <p>The USB System Software may determine the cause of the failure by checking the status and noting the loss of the device's power source.</p>

3.3.4 DFU Functional descriptor

The descriptor is identical for both the run-time and the DFU mode descriptor sets.

Table 8. DFU Functional Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	07h	Size of the descriptor, in bytes.
1	bDescriptorType	1	21h	DFU Functional descriptor type

Table 8. DFU Functional Descriptor

Offset	Field	Size	Value	Description
2	bmAttributes	1	Bit Mask	DFU attributes Bit 7..3: reserved Bit 2: device is able to communicate via USB after Manifestation phase (Manifestation Tolerant bit) 0 = no, must see bus reset 1 = yes Bit 1: upload capable (CanUpload bit) 0 = no 1 = yes Bit 0: download capable (CanDnload bit) 0 = no 1 = yes
3	wDetachTimeOut	2	Number	Time, in milliseconds, that the device will wait after receipt of the DFU_DETACH request. If this time elapses without a USB reset, then the device will terminate the Reconfiguration phase and revert back to normal operation. This represents the maximum time that the device can wait (depending on its timers, etc.). The host may specify a shorter timeout in the DFU_DETACH request.
5	wTransfertSize	2	Number	Maximum number of bytes that the device can accept per control write transaction.
7	bcdDFUVersion	2	Number	Numeric expression identifying the version of the DFU Specification release.

3.3.5 DFU Status packet

A packet sent by the device as a response to the DFU_GETSTATUS request

Table 9. Status packet format

Offset	Field	Size	Value	Description
1	bStatus	1	Number	An indication of the status resulting from the execution of the most recent request.
2	bwPollTimeout	3	Number	Minimum time, in milliseconds, that the host should wait before sending a subsequent DFU_GETSTATUS request.
5	bState	1	Number	An indication of the state that the device is going to enter immediately following transmission of this response. (By the time the host receives this information, this is the current state of the device).
6	iString	1	Index	Index of the status description in string table.

3.4 APIs

3.4.1 STDFU_Abort

STDFU_Abort (PHANDLE phDevice)

Issues an DFU_ABORT request on the Control Pipe (Endpoint0). The Abort request enables the host to exit from certain states and return to the STATE_DFU_IDLE state. The device sets the OK status on receipt of this request.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.2 STDFU_Close

STDFU_Close (PHANDLE phDevice)

Closes the DFU driver.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.3 STDFU_Clrstatus

STDFU_Clrstatus (PHANDLE phDevice)

Issues a DFU_CLRSTATUS request on the Control Pipe (Endpoint0). Any time the device detects an error and reports an error indication status to the host in the response to a DFU_GETSTATUS request, it enters the STATE_DFU_ERROR state. The device cannot transition from the dfuERROR state after reporting any error status, until after it has received a DFU_CLRSTATUS request. Upon receipt of DFU_CLRSTATUS, the device sets a status of OK and transition to the STATE_DFU_IDLE state. Only then is it able to transition to other states.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.4 STDFU_Detach

STDFU_Detach (PHANDLE phDevice)

Issues a DFU_DETACH request on the Control Pipe (Endpoint0).

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open. Pointed handle will contain NULL after call
		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.5 STDFU_Dnload

STDFU_Dnload (PHANDLE phDevice, UCHAR *pBuffer, ULONG nBytes, USHORT nBlock)

Issues a DFU_DNLOAD request on the Control Pipe (Endpoint0).

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	pBuffer	UCHAR *	Buffer of Data.
	nBytes	ULONG	Size of data to be downloaded in bytes.
	nBlock	USHORT	Block number.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.6 STDFU_GetConfigurationDescriptor

STDFU_GetConfigurationDescriptor (PHANDLE phDevice, UINT nConfigIdx,

PUSB_CONFIGURATION_DESCRIPTOR pDesc)

Gets the configuration descriptor.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	nConfigIdx	UINT	Index of the selected Configuration.
	pDesc	PUSB_CONFIGURATION_DESCRIPTOR	Buffer the descriptor will be copied to.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.7 STDFU_GetDFUDescriptor

STDFU_GetDFUDescriptor (PHANDLE phDevice, PUINT pDFUInterfaceNum, PUINT pNbOfAlternates, PDFU_FUNCTIONAL_DESCRIPTOR pDesc)

Gets the DFU functional descriptor.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	pDFUInterfaceNum	PUINT	Number of the interface. Zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
	pNbOfAlternates	PUINT	Number of possible Alternate settings for selected interface.
	pDesc	PDFU_FUNCTIONAL_DESCRIPTOR	Buffer the descriptor will be copied to.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.8 STDFU_GetDeviceDescriptor

STDFU_GetDeviceDescriptor(PHANDLE phDevice, PUSB_DEVICE_DESCRIPTOR pDesc)

Gets the Device descriptor.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	pDesc	PUSB_DEVICE_DESCRIPTOR	Buffer the descriptor will be copied to.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.9 STDFU_GetInterfaceDescriptor

STDFU_GetInterfaceDescriptor (PHANDLE phDevice, UINT nConfigIdx, UINT nInterfaceIdx, UINT nAltSetIdx, PUSB_INTERFACE_DESCRIPTOR pDesc)

Gets the Interface descriptor

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	nConfigIdx	UINT	Index of the selected Configuration.
	nInterfaceIdx	UINT	Index of the selected Interface.
	nAltSetIdx	UINT	Index of the selected Alternate Setting.
	pDesc	PUSB_INTERFACE_DESCRIPTOR	Buffer the descriptor will be copied to.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.10 STDFU_GetNbOfAlternates

STDFU_GetNbOfAlternates (PHANDLE phDevice, UINT nConfigIdx, UINT nInterfaceIdx, PUINT pNbOfAltSets)

Gets the number of available Alternate settings for a given configuration and a given interface.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	nInterfaceIdx	UINT	Index of the selected Configuration.
	nConfigIdx	UINT	Index of the selected Interface.
	pNbOfAltSets	PUINT	Pointer to Alternate Setting's Number.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.11 STDFU_GetNbOfConfigurations

STDFU_GetNbOfConfigurations (PHANDLE phDevice, PUINT pNbOfConfigs)

Gets the number of available configurations.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	pNbOfConfigs	PUINT	pointer to the configuration's number.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.12 STDFU_GetNbOfInterfaces

STDFU_GetNbOfInterfaces (PHANDLE phDevice, UINT nConfigIdx, PUINT pNbOfInterfaces)

Gets the number of available interfaces, for a given configuration.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	nConfigIdx	UINT	Index of the selected Configuration.
	pNbOfInterfaces	PUINT	pointer to Interfaces Number.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.13 STDFU_GetStringDescriptor

STDFU_GetStringDescriptor (PHANDLE phDevice, DWORD Index, LPSTR szString, UINT nStringLength)

Gets the string descriptor.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	Index	DWORD	Requested string descriptor Index. If this index is too high, this function will return an error.
	szString	LPSTR	Buffer the string descriptor will be copied to.
	nStringLength	UINT	Buffer size.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.14 STDFU_Getstate

STDFU_Getstate (PHANDLE phDevice, UCHAR *pState)

Issues a DFU_GETSTATE request on the Control Pipe (Endpoint0).

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	pState	UCHAR *	Buffer DFU Status structure will be copied to.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.15 STDFU_Getstatus

STDFU_Getstatus (PHANDLE phDevice, DFUSTATUS *DfuStatus)

Issues a DFU_GETSTATUS request on the Control Pipe (Endpoint0).

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open. If current DFU state is STATE_DFU_MANIFEST_SYNC the pointed handle will contain NULL after GetStatus call.
	DfuStatus	DFUSTATUS *	Buffer DFU State will be copied to.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.16 STDFU_Open

STDFU_Open (LPSTR szDevicePath, PHANDLE phDevice)

Opens the DFU driver, giving access to its descriptor.

	Field	Type	Description
Input	szDevicePath	LPSTR	Device symbolic link.
	phDevice	PHANDLE	Handle returned by the function while the driver is successfully open.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.17 STDFU_SelectCurrentConfiguration

STDFU_SelectCurrentConfiguration (PHANDLE phDevice,UINT nConfigIdx, UINT nInterfaceldx, UINT nAltSetIdx)

Selects the currently active mode for a device, giving the configuration, the interface and the alternate setting. The pipes must be in closed state for this function to success.

	Field	Type	Description
Input	phDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	nConfigIdx	UINT	Index of the desired configuration.
	nInterfaceldx	UINT	Index of the desired interface.
	nAltSetIdx	UINT	Index of the desired alternate setting.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

3.4.18 STDFU_Upload

STDFU_Upload (PHANDLE phDevice, UCHAR *pBuffer, ULONG nBytes, USHORT nBlock)

Issues an DFU_UPLOAD request on the Control Pipe (Endpoint0).

	Field	Type	Description
Input	hDevice	PHANDLE	Pointer to handle returned by the function STDFU_Open.
	pBuffer	UCHAR *	Buffer of Data.
	nBytes	ULONG	Size of data to be uploaded in bytes.
	nBlock	USHORT	Block number.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

4 STDFUPRT library (STDFUPRT.DLL)

4.1 Library description

This DLL implements all the state machines, as described in the standard DFU for managing the state machine independently of any GUI. The user can launch an operation and get its state even when the operation is running, on other hand, it is not possible to launch an operation if another one is already running. Additionally, this DLL allows you to extract the memory mapping of the device, as specified in the Dfu STMicronics Extension Specification, and to get a decoded view of it.

4.2 Constants

Name	Value
MAX_PATH	260

4.3 Types

4.3.1 DFUIMAGEELEMENT

Table 10. DFUIMAGEELEMENT record

Offset	Field	Size	Value	Description
0	dwAddress	4	Address	Start data address
4	dwDataLength	4	Number	Data length
8	Data	4		Data buffer

4.3.2 MAPPINGSECTOR

Table 11. MAPPINGSECTOR record

Offset	Field	Size	Value	Description
0	dwStartAddress	4	Address	Sector start address.
4	dwAliasedAddresses	4	Address	Reserved.
8	dwSectorIndex	4	Index	Sector index.
12	dwSectorSize	4	Number	Sector Size in KB.

Table 11. MAPPINGSECTOR record

Offset	Field	Size	Value	Description
16	bSectorType	1	00h..04h	Type of sector, to know if the sector allows read, write or erase operations. Bit0 : BIT_READABLE Bit1 : BIT_ERASABLE Bit2 : BIT_WRITEABLE
17	UseForOperation	2	Boolean	To include or exclude the sector for an operation.

4.3.3 MAPPING

Table 12. MAPPING record

Offset	Field	Size	Value	Description
0	nAlternate	1	Number	Alternate number, the mapping is created for.
1	Name	260	string	Sector name.
261	NbSectors	4	Number	Number of available sectors .
265	pSectors	18	Pointer	Pointer to sectors list.

4.3.4 DFUThreadContext

Table 13. DFUThreadContext class

Offset	Field	Size	Value	Description
0	DfuGUID	16	GUID	Device GUID in DFU mode.
16	AppGUID	16	GUID	Device GUID in application mode.
32	Operation	2	0..4	Operation type defined as: OPERATION_DETACH = 0 OPERATION_RETURN = 1 OPERATION_UPLOAD = 2 OPERATION_ERASE = 3 OPERATION_UPGRADE = 4.
34	bDontSendFFTransfersForUpgrade	2	Boolean	= TRUE, the 0xFF bytes will not be sent on download operation.
36	hImage	4		Pointer to an Image handle containing binary data to be downloaded or uploaded.
40	szDevLink	260		Symbolic Device link, used as path to access device.
300	dwTag	4	Number	Data increasing each time there any changes in the context.
304	Percent	1	Number	Current operation progress.
305	wTransferSize	2	Number	Data size to be transferred.
307	LastDFUStatus	6	DFU Status code	Last DFU status received.

Table 13. DFUThreadContext class

Offset	Field	Size	Value	Description
313	CurrentRequest	2	DFU Request code	Current request.
315	CurrentNBlock	2	Number	Current block number.
317	CurrentLength	2	Number	Current used length.
319	CurrentAddress	4	Address	Current used address.
323	CurrentImageElement	2		Current used image element.
325	ErrorCode	4	DFUError code	Error code value.
329	hDevice	4		Pointer to an opened device handle.

4.4 APIs

4.4.1 STDFUPRT_CreateMappingFromDevice

STDFUPRT_CreateMappingFromDevice(PCHAR szDevLink, PMAPPING *ppMapping, PDWORD pNbAlternates)

Gets memory device mapping for a given alternate number, the result is copied to the ppMapping argument as an array of mapping sector lists.

	Field	Type	Description
Input	szDevLink	PCHAR	Symbolic Device link.
	ppMapping	PMAPPING*	Pointer to PMAPPING reference, the mapping will be copied to.
	pNbAlternates	PDWORD	Number of alternates to be used
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

4.4.2 STDFUPRT_DestroyMapping

STDFUPRT_DestroyMapping (PMAPPING *ppMapping)

Destroys the given mapping and free allocated memory.

	Field	Type	Description
Input	ppMapping	PMAPPING*	Pointer to PMAPPING reference.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

4.4.3 STDFUPRT_GetOperationStatus

STDFUPRT_GetOperationStatus (DWORD OperationCode, PDFUThreadContext pContext)

Gets current context for the given operation. If given operation does not exists, STDFUPRT_BADPARAMETER error is returned.

	Field	Type	Description
Input	OperationCode	DWORD	Operation reference got when calling STDFU_LaunchOperation.
	pContext	PDFUThreadContext	Pointer to active DFUThreadContext.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

4.4.4 STDFUPRT_LaunchOperation

STDFUPRT_LaunchOperation (PDFUThreadContext pContext, PDWORD pOperationCode)

Launches the programmed operation described by the given context and operation code.

	Field	Type	Description
Input	pContext	PDFUThreadContext	Pointer to PDFUThreadContext which contains all data needed to launch an operation.
	pOperationCode	PDWORD	Reference to the launched operation.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

4.4.5 STDFUPRT_StopOperation

STDFUPRT_StopOperation (DWORD OperationCode, PDFUThreadContext pLastContext)

Terminates the operation thread.

	Field	Type	Description
Input	OperationCode	DWORD	Operation reference got when calling STDFU_LaunchOperation.
	pLastContext	PDFUThreadContext	Pointer to the current context.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

5 STDFUFiles library (STDFUFiles.DLL)

5.1 Library description

The STDFUFiles library implements all the encoding and decoding of DFU file format. This format is described in the DFU STMicroelectronics Extension Specification document. Additionally, this DLL also implements a way to import or export binary files in Motorola-S19, Intel-Hex and Binary formats. This layer implements a way to access the files or memory images a DFU application needs to work with.

5.2 Constants

Name	Value
PREFIX_SIGNATURE_SIZE	5
PREFIX_SIGNATURE	"DfuSe"
PREFIX_VERSION	1
TARGET_PREFIX_SIGNATURE_SIZE	6
TARGET_PREFIX_SIGNATURE	"Target"

Table 14. File access errors

Name	Value	Description
STDFUFILES_ERROR_OFFSET	0x12346000	File access error offset.
STDFUFILES_NOERROR	0x12340000	No error.
STDFUFILES_BADSUFFIX	0x12346002	Bad DFU Suffix format.
STDFUFILES_UNABLETOOPENFILE	0x12346003	Unable to open file.
STDFUFILES_UNABLETOOPENTEMPFILE	0x12346004	Unable to open temporary file.
STDFUFILES_BADFORMAT	0x12346005	Bad DFU file format.
STDFUFILES_BADADDRESSRANGE	0x12346006	Bad address range.
STDFUFILES_BADPARAMETER	0x12346008	Bad parameter.
STDFUFILES_UNEXPECTEDERROR	0x1234600A	Unexpected error.
STDFUFILES_FILEGENERALERROR	0x1234600D	File general error.

5.3 Types

5.3.1 DFUPREFIX

Table 15. DFUPREFIX record

Offset	Field	Size	Value	Description
0	szSignature	5	"DfuSe"	File signature identifier.
5	bVersion	1	0x01	Dfu format revision.
6	dwImageSize	4	Number	DFU file length in byte.
10	bTargets	1	Number	Number of DFU images in the file.

5.3.2 DFUSUFFIX

Table 16. DFUSUFFIX record

Offset	Field	Size	Value	Description
0	bcdDeviceLo	1		LSB of the release number of the device associated with this file. Either FFh or a BCD firmware release or version number.
1	bcdDeviceHi	1		MSB of the release number of the device associated with this file. Either FFh or a BCD firmware release or version number.
2	idProductLo	1		LSB of the Product ID associated with this file. Either FFh or must match device's Product ID.
3	idProductHi	1		MSB of the Product ID associated with this file. Either FFh or must match device's Product ID.
4	idVendorLo	1		LSB of the Vendor ID associated with this file. Either FFh or must match device's Vendor ID.
5	idVendorHi	1		MSB of the Vendor ID associated with this file. Either FFh or must match device's Vendor ID.
6	bcdDFULo	1	0x1A	LSB of the DFU specification number.
7	bcdDFUHi	1	0x01	MSB of the DFU specification number.
8	ucDfuSignature	3	"UFD"	The unique DFU signature field.
11	bLength	2	16	The length of this DFU Suffix including dwCRC.
13	dwCRC	4		The CRC of the entire file, excluding dwCRC.

5.3.3 TARGETPREFIX

Table 17. TARGETPREFIX record

Offset	Field	Size	Value	Description
0	szSignature	6	"Target"	Target signature.
6	bAlternateSetting	1		Alternate setting index for witch the associated image can be used.
7	bTargetNamed	1	Boolean	Target is named or not
8	szTargetName	255		Target name.
303	dwTargetSize	4	Number	Length of the associated image excluding Target prefix.
307	dwNbElements	4	Number	Number of element in the associated image.

5.3.4 ELEMENT

Table 18. ELEMENT record

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

5.4 APIs

5.4.1 STDFUFILES_AppendImageToDFUFile

STDFUFILES_AppendImageToDFUFile (HANDLE hFile, HANDLE Image)

Stores the given image in the given DFU file.

	Field	Type	Description
Input	hFile	HANDLE	Pointer to a file handle
	Image	HANDLE	Pointer to an image handle
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

5.4.2 STDFUFILES_CloseDFUFile

STDFUFILES_CloseDFUFile (HANDLE hFile)

Closes the given DFU file and frees its descriptor.

	Field	Type	Description
Input	hFile	HANDLE	Pointer to a file handle.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

5.4.3 STDFUFILES_CreatelImage

STDFUFILES_CreatelImage(PHANDLE pHandle, BYTE nAlternate)

Creates an empty Image for a given alternate number in the given Handle pointer

	Field	Type	Description
Input	pHandle	PHANDLE	Pointer to a handle.
	nAlternate	BYTE	Alternate number.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.4 STDFUFILES_CreatelImageFromMapping

STDFUFILES_CreatelImageFromMapping (PHANDLE pHandle, PMAPPING pMapping)

Creates an empty image according to the given mapping.

	Field	Type	Description
Input	pHandle	PHANDLE	Pointer to a handle
	pMapping	PMAPPING	Pointer to a mapping
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

5.4.5 STDFUFILES_CreateNewDFUFile

STDFUFILES_CreateNewDFUFile (PSTR pPathFile, PHANDLE phFile, WORD Vid, WORD Pid, WORD Bcd)

Creates an empty DFU file and register given identifiers.

	Field	Type	Description
Input	pPathFile	PSTR	File path
	phFile	PHANDLE	Pointer to the file handle to be opened
	Vid	WORD	Device Vender ID
	Pid	WORD	Device Product ID
	Bcd	WORD	USB specification release number in binary coded decimal
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

5.4.6 STDFUFILES_DestroyImage

STDFUFILES_DestroyImage (PHANDLE pHandle)

Destroys the given image and frees the allocated memory.

	Field	Type	Description
Input	pHandle	PHANDLE	Pointer to the Image handle
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

5.4.7 STDFUFILES_DestroyImageElement

STDFUFILES_DestroyImageElement (HANDLE Handle, DWORD dwRank)

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

	Field	Type	Description
	pHandle	PHANDLE	Image handle
Input	DWORD	dwRank	Element index
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

5.4.8 STDFUFILES_DuplicateImage

STDFUFILES_DuplicateImage (HANDLE hSource, PHANDLE pDest)

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

	Field	Type	Description
Input	hSource	HANDLE	Original image handle
	pDest	PHANDLE	Pointer to the duplicated image handle
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code)

5.4.9 STDFUFILES_FilterImageForOperation

STDFUFILES_FilterImageForOperation (HANDLE Handle, PMAPPING pMapping, DWORD Operation, BOOL bTruncateLeadFFFForUpgrade)

Filter an image for the given operation. The bTruncateLeadFFFForUpgrade is used for UPGRADE and ERASE operations to remove unnecessary traffic if needed.

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	pMapping	PMAPPING	Pointer to the mapping.
	Operation	DWORD	Operation code.
	bTruncateLeadFFFForUpgrade	BOOL	Truncate if TRUE.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.10 STDFUFILES_GetImageAlternate

STDFUFILES_GetImageAlternate (HANDLE Handle, PBYTE pAlternate)

Gets the alternates setting stored in the given Image.

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	pAlternat	PBYTE	Buffer, the alternates will be copied to.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.11 STDFUFILES_GetImageElement

STDFUFILES_GetImageElement (HANDLE Handle, DWORD dwRank, PDFUIIMAGEELEMENT pElement)

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

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	dwRank	DWORD	Element index.
	pElement	PDFUIIMAGEELEMENT	Pointer to the retrieved IMAGEELEMENT.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.12 STDFUFILES_GetImageName

STDFUFILES_GetImageName (HANDLE Handle, PSTR Name)

Gets the image name.

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	Name	PSTR	image name.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.13 STDFUFILES_GetImageNbElement

STDFUFILES_GetImageNbElement (HANDLE Handle, PDWORD pNbElements)

Gets the number of elements in the given image.

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

5.4.14 STDFUFILES_ImageFromFile

STDFUFILES_ImageFromFile (PSTR pPathFile, PHANDLE plmage, BYTE nAlternate)

Create a DFU image from the given S19 or Hex file specified by pPathFile. The file type is recognised by its extension (*.s19 or *.Hex).

	Field	Type	Description
Input	pPathFile	PSTR	Path of the file to be loaded, can be S19 or Hex file.
	pImage	PHANDLE	Pointer to a DFU image handle to be created.
	nAlternate	BYTE	Alternate number to be accisited to the DFU image.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.15 STDFUFILES_ImageToFile

STDFUFILES_ImageToFile (PSTR pPathFile, HANDLE Image)

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

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

5.4.16 STDFUFILES_OpenExistingDFUFile

STDFUFILES_OpenExistingDFUFile (PSTR pPathFile, PHANDLE phFile, PWORD pVid, PWORD pPid, PWORD pBcd, PBYTE pNbImages)

Opens an existing DFU file and retrieves a pointer to the file handle, the registered Vid, Pid, Bcd and the image count.

	Field	Type	Description
Input	pPathFile	PSTR	File path.
	phFile	PHANDLE	Pointer to file handle.
	pVid	PWORD	Device Product ID.
	pPid	PWORD	Device Vendor ID.
	pBcd	PWORD	USB specification release number in binary coded decimal.
	pNbImages	PBYTE	Image count.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.17 STDFUFILES_ReadImageFromDFUFile

STDFUFILES_ReadImageFromDFUFile (HANDLE hFile, int Rank, PHANDLE pImage)

Reads one image from a given DFU file handle at a given index, the read image is copied to the given image handle.

	Field	Type	Description
Input	hFile	HANDLE	Pointer to file handle.
	Rank	int	Image index to be read.
	plmage	PHANDLE	Handle, the image will be copied to.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.18 STDFUFILES_SetImageElement

STDFUFILES_SetImageElement (HANDLE Handle, DWORD dwRank, BOOL bInsert, DFUIMAGEELEMENT Element)

Inserts or replace a DFUIMAGEELEMENT in the given image at the given index.

	Field	Type	Description
Input	Handle	HANDLE	Pointer to the image handle.
	dwRank	DWORD	Index, the DFUIMAGEELEMENT will be inserted at.
	bInsert	BOOL	Insert if TRUE. Replace if FALSE.
	Element	DFUIMAGEELEMENT	Pointer to DFUIMAGEELEMENT.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

5.4.19 STDFUFILES_SetImageName

STDFUFILES_SetImageName (HANDLE Handle, PSTR Name)

Sets image name.

	Field	Type	Description
Input	Handle	HANDLE	Pointer to image handle.
	Name	PSTR	Image name.
Output		DWORD	STDFU_NOERROR = SUCCESS, Error otherwise (Error Code).

6 Document references

Table 19. Document references

ID	Document
UM0391	DfuSe File Format Specification

7 Revision history

Table 20. Document revision history

Date	Revision	Changes
06-Jun-2007	1	Initial release.

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.

© 2007 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

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

www.st.com

