STSECTOOLFUSE Library

API Specification

ST Document Reference Number: DVD-API-351

Document Version: 1.7.3

Library Version: 1.7.3

This library is intended for 7105

CONFIDENTIAL

1 Document Change History

Date	Document Version	Library Version	Reason and Change
12/01/2007	1.00.00	1.00.00	First release
17/01/2007	1.00.01	1.00.01	Minor updates. Added SECTOOLITEM_ALL_OTP_ITEM to Appendix A.
18/01/2007	1.00.02	1.00.02	Minor updates. Added Appendix C.
26/02/2007	1.00.03	1.00.03	Added mtp_locka bit fuses. Added restrictions to bulk data writing.
09/03/2007	1.00.04	1.00.04	New library version released.
08/05/2007	1.01.00	1.01.00	7200 build added.
08/05/2007	1.01.01	1.01.01	Added STSECTOOLFUSE_ReadSysRegItem and STSECTOOLFUSE_ReadMtpItem functions.
10/05/2007	1.01.02	1.01.02	Added information about utility builds (section 2.5). Added more 7200 fuses.
24/05/2007	1.01.03	1.01.03	Added SAF locations to fuse value list in the utility. Remove warning about utility build libraries being large. Added STSECTOOLFUSE_ERROR_NOT_OTP_ITEM_ERROR.
27/09/2007	1.01.04	1.01.04	Added dependencies section. More comments for Start/StopOTP. Clarified unused bits in ReadItem and WriteItem.
27/09/2007	1.01.05	1.01.05	New release for st40 4.0.2 compilation.
11/02/2008	1.2.0	1.2.0	New release with 5167 support
15/02/2008	1.2.1	1.2.1	Added STSECTOOLFUSE_ALL_OTP_ITEM fuse item for 5167. Built against STCOMMON 2.1.3
25/03/2008	1.2.2	1.2.2	Fixed issue: unable setting boot_rom_en fuse on 5167.
21/04/2008	1.2.3	1.2.3	Added HowToExecuteUtility.txt help file. Renamed ST40 utility executable to use .out suffix.
13/06/2008	1.2.4	1.2.4	This addresses the problem of the otp appnote not appearing in the release's zip files for 7109
07/08/2008	1.3.0	1.3.0	New Release for 7105 and 7111 on OS21
17/11/2008	1.3.2	1.3.2	New release supporting 7141 and 7200 cut 2 on Linux and OS21
03/12/2008	1.3.3	1.3.3	Minor update to fuse definitions
05/12/2008	1.3.4	1.3.4	Updated fuse definitions for: STSECTOOLFUSE_TRANS_CW_ENABLE STSECTOOLFUSE_CW_SECURE
05/01/2009	1.3.5	1.3.5	Fixed bug in Linux ioctl driver code regarding STSECTOOLFUSE_BulkReadItem and STSECTOOLFUSE_BulkWriteItem functions.

CONFIDENTIAL

Date	Document Version	Library Version	Reason and Change
17/03/2009	1.4.0	1.4.0	New release support 5197
30/03/2009	1.4.1	1.4.1	Added new fuses for 5197
14/03/2009	1.4.2	1.4.2	Updated 5197 write (auto retry upon failure)
29/06/2009	1.4.3	1.4.3	1.4.4
11/08/2009	1.4.4	1.4.4	Fixed bug BRISec1638 - Set correct R/W mode for STSECTOOLFUSE_TRANS_CW_SECURE_OTP_ITEM. Fixed bug BRISec1634 - 7141 cut >=2 support added for fuse STSECTOOLFUSE_TRANS_CW_SECURE_OTP_ITEM 7109 gcc version 3.x support dropped, due to dependencies (bug BRISec1637)
09/09/2009	1.4.5	1.4.5	Fixed bugs BRISec1712 - jtag_protect address incorrect.
30/09/2009	1.4.6	1.4.6	Fixed bug BRISec1732 - Bulk32 write fails.
14/10/2009	1.4.7	1.4.7	Fixed bug BRISec1794 - STSECTOOLFUSE 7141 c2 utility includes 7141 c1 fuse definitions file Fixed feature request BRISec1792 - Added additional fuses to 7141 c2 fuse definitions to meet customer requirements
21/10/2009	1.4.8	1.4.8	Implemented request BRISec1797 - Added STSECTOOLFUSE_DIRT_DISABLE fuse item for 5197 Fixed re-opened bug BRISec1792 since some 7141 c2 fuse addresses for new fuses in 1.4.7 were incorrect
23/10/2009	1.4.9	1.4.9	Fix for bug BRISec1808 - Add PCI disable fuse for 7105.
11/11/09	1.4.10	1.4.10	Added uclibc support for Linux library

CONFIDENTIAL

Date	Document Version	Library Version	Reason and Change
12/01/10	1.5.0	1.5.0	Added support for STi5206/ STi5289. Added stsectoolfuse_utility_readme.txt, describing how to examine and blow fuses using the utility. The following API functions have been deprecated - there functionality is performed internally. The functions remain to allow backwards-compatibility but will cause compilation warnings: STSECTOOLFUSE_StartOTP STSECTOOLFUSE_StopOTP STSECTOOLFUSE_PermanentWriteEnable STSECTOOLFUSE_PermanentWriteDisable STSECTOOLFUSE_Identity STSECTOOLFUSE_ReadMtpItem STSECTOOLFUSE_ReadSysRegItem Document examples and fuse list appendices updated.
11/03/10	1.6.0	1.6.0	Added 5197 cut 3 support (BRIsec002122)
15/04/10	1.6.1	1.6.1	Fixed : BRIsec002197 [5289/5206 SIgnature checking cannot be enabled] BRIsec002173[Support for 5197 cut 3.x omitted from release list] BRIsec002168 [STSECTOOLFUSE update to new build system] BRIsec002196 [7141 c.2.2 fuses not supported] BRIsec002168 [STSECTOOLFUSE update to new build system] BRIsec002155 [Minor doc error's]
20/05/10	1.6.2	1.6.2	Fixed: BRIsec2273 - Updated 5197/5167 swaf timing and reset sequence BRIsec2293 - Re-introduced 5167 support BRIsec2232 - 5197 cut 3 VCC size change BRIsec2349 - Reduce Safmem programming delay

CONFIDENTIAL

Date	Document Version	Library Version	Reason and Change
26/05/10	1.6.3	1.6.3	Added 5206 Linux support Fixed: BRIsec2242 - 5206 crypto_enable missing
23/08/10	1.7.0	1.7.0	BRIsec2502 - Added device multi-cut support Increased code maintainability index
29/09/10	1.7.1	1.7.1	BRIsec2573 - Added crypt_fw_version_ctrl fuse on 5206 and 7141 platforms
28/10/10	1.7.2	1.7.2	Fixed BRIsec2620 - Error in ioctl code forces BulkWrite operation instead of BulkRead
31/01/11	1.7.3	1.7.3	BRIsec2602: 5289: Add trans_cp_secure and trans_cpcw_sec_polarity BRIsec2710: [STSECTOOLFUSE] _BulkWriteItem() should return error for locked fuse

CONFIDENTIAL

2 Introduction

2.1 Library Version

This document references Library version `STSECTOOLFUSE_REL_1.7.3`.

2.2 Overview

STSecToolFuse is a STAPI-compliant driver that allows fuse values to be read and written.

Fuses are used to store read-only or write-once value. The technology used to store these values varies across devices.

As well as a description of the driver API, this document also contains:

Appendix A - descriptions of the fuses handled by this library.

Appendix B - examples of how to use this driver.

2.3 Dependencies

Below is a list of libraries/drivers on which STSECTOOLFUSE has a direct dependency. (The library/driver version indicates the version used during release testing. Later versions may be used, where they retain backwards-compatibility with the versions specified.)

Dependencies:

`STAPLER version STAPLER_REL-1.2.0`

2.4 Abbreviations

STSECTOOLFUSE	Security Toolkit Fuse driver
MTP	Multiple Time Programmable - Bit programmable - any unblown bit can be blown at any time. MTP fuses may be written multiple times, no bit in the fuse already set to 1 may be reset to 0.
OTP	OTP fuses may be written only once. Their value cannot be further modified

Table 1 : Abbreviations

2.5 Utility build

This library has a bundled utility (stsectoolfuse_utility.out) for reading and setting fuses on a reference board. The utility takes the form of an interactive command-line interface.

Refer to the file HowToExecuteUtility.txt for instructions on loading the utility on a reference board.

Refer to the file stsectooolfuse_utility_readme.txt for instructions on using the utility.

3 Constants

3.1 Error constants

ST_NO_ERROR	No error occurred during the specified operation.
ST_ERROR_BAD_PARAMETER	A parameter passed to an API function has an invalid value.
ST_ERROR_ALREADY_INITIALIZED	An attempt was made to initialise the driver when it was already initialised.
ST_ERROR_FEATURE_NOT_SUPPORTED	This function is not supported on the current device
STSECTOOLFUSE_ERROR_WRITE_NOT_ENABLED	An attempt has been made to write a fuse that is not writable.
STSECTOOLFUSE_ERROR_OTP_DATA_ERROR	A hardware error occurred while reading OTP data. Refer to device's fuse map to check if the fuse is write-only.
STSECTOOLFUSE_ERROR_OTP_WRITE_ERROR	A hardware error occurred while writing OTP data. Refer to your device's fuse map to check if the fuse is read-only
STSECTOOLFUSE_ERROR_NOT_OTP_ITEM_ERROR	Trying to read a fuse item, using STSECTOOLFUSE_ReadItem(), which cannot be read by this function. E.g. a bulk fuse
STSECTOOLFUSE_ERROR_NOT_INITIALISED	An attempt was made to call an API function when the driver has not been initialised.

Table 2 : Error constants

CONFIDENTIAL

3.2 Other API constants

STSECTOOLFUSE_NOINT	Constant that may be passed to STSECTOOLFUSE_Init to indicate that no interrupts are to be used by the driver.

4 Types

The following types are defined by the driver API:

STSECTOOLFUSE_InitParams_t	Type passed during Library initialisation.
STSECTOOLFUSE_TermParams_t	Type passed during Library termination.
STSECTOOLFUSE_ITEM_t	Type used enumerate all available fuses for a particular device

Table 3 :Types defined by the driver API

4.1 STSECTOOLFUSE_InitParams_t

Description

A pointer to this structure is passed during a call to STSECTOOLFUSE_Init. This structure is not currently used, but may be in future extensions of the driver.

Definition

```
typedef void STSECTOOLFUSE_InitParams_t;
```

Elements

None

Comments

See Also

STSECTOOLFUSE_TermParams_t

4.2 STSECTOOLFUSE_TermParams_t

Description

A pointer to this structure is passed during a call to STSECTOOLFUSE_Term. This structure is not currently used, but may be in future extensions of the driver.

Definition

```
typedef void STSECTOOLFUSE_TermParams_t;
```

CONFIDENTIAL

Elements

None

Comments

See Also

`STSECTOOLFUSE_InitParams_t`

CONFIDENTIAL

5 Functions

STSECTOOLFUSE_Init	Initialises the driver.
STSECTOOLFUSE_GetRevision	Gets the driver revision.
STSECTOOLFUSE_Term	Terminates the driver.

Table 4 : Common functions

STSECTOOLFUSE_ReadItem	Reads a non-bulk fuse value.
STSECTOOLFUSE_WriteItem	Writes a non-bulk fuse value.
STSECTOOLFUSE_ReadBulkItem	Reads a multi-byte bulk fuse value.
STSECTOOLFUSE_WriteBulkItem	Writes a multi-byte bulk fuse value.

Table 5 : STSECTOOLFUSE driver-specific functions

The following functions have been deprecated from the STSECTOOLFUSE API in the version indicated. The deprecated functions will not be updated in future releases and it is recommended that new code should not use these functions.

These functions will continue to be available in the stsectoolfuse.h header file and as functions in the released library in order to maintain backwards-compatibility with existing code.

STSECTOOLFUSE_StartOTP	Powers-up the OTP. Operation now handled internally.	deprecated 1.5.0
STSECTOOLFUSE_StopOTP	Powers-down the OTP. Operation now handled internally.	deprecated 1.5.0
STSECTOOLFUSE_PermanentWriteEnable	Enables operations on the specified fuse which cannot be undone. Operation now handled internally.	deprecated 1.5.0
STSECTOOLFUSEPermanentWriteDisable	Reverts to safe operation Operation now handled internally..	deprecated 1.5.0
STSECTOOLFUSE_Identity	Returns identity data. Function returned data no longer relevant for current devices.	deprecated 1.5.0
STSECTOOLFUSE_ReadSysRegItem	Reads a fuse value from system registers STSECTOOLFUSE_ReadItem performs identical functionality.	deprecated 1.5.0
STSECTOOLFUSE_ReadMtpItem	Reads a fuse value from system registers. STSECTOOLFUSE_ReadItem performs identical functionality.	deprecated 1.5.0

Table 6 : STSECTOOLFUSE deprecated functions

CONFIDENTIAL

STSECTOOLFUSE_GetRevision()

Description

Returns the current Library version

Definition

```
ST_Revision_t STSECTOOLFUSE_GetRevision(void);
```

Arguments

None.

Return Value

`Revision`	The Library version.

Comments

The Revision is a character string which identifies the revision (or version) of the software which controls this Library.

This function may be called at any time (including before initialization of the Library).

The revision string returned by the function has the following form:

STSECTOOLFUSE-REL_**A.B.C DcE**

Where each of the capital letters above is replaced with the following data:

A	Decimal integer	Driver major revision number
B	Decimal integer	Driver minor revision number
C	Decimal integer	Driver patch revision number
D	String	Name of device for which the driver has been built
E	Decimal integer	Revision (cut) of the device for which the driver has been built

CONFIDENTIAL

STSECTOOLFUSE_Init()

Description

Initializes the STSECTOOLFUSE driver.

Definition

```
ST_ErrorCode_t STSECTOOLFUSE_Init(const ST_DeviceName_t DeviceName,
                                  const STSECTOOLFUSE_InitParams_t *InitParams,
                                  int InterruptLevel)
```

Arguments

`DeviceName`	STAPI device name string. The same devicename must be used for terminating the driver as was used when initialising it.
`InterruptLevel`	Indicates the interrupt level to use for interrupts installed by the driver. This parameter is currently not used and must be set the constant `STSECTOOLFUSE_NOINT`..
`InitParams`	Pointer to a driver intialisation structure. This parameter is currently unused and must be set to `NULL`.

Return Value

`ST_NO_ERROR`	No error occurred during the specified operation.
`ST_ERROR_BAD_PARAMETER`	A parameter passed to an API function has an invalid value.
`ST_ERROR_ALREADY_INITIALIZED`	An attempt was made to initialise the driver when it was already initialised.

Comments

No other API functions (except `STSECTOOLFUSE_GetRevision`) may be called before this function has been called.

CONFIDENTIAL

STSECTOOLFUSE_Term()

Description

Terminates the STSECTOOLFUSE driver and frees all driver resources.

Definition

```
ST_ErrorCode_t STSECTOOLFUSE_Term(const ST_DeviceName_t DeviceName,
                                  const STSECTOOLFUSE_TermParams_t *TermParam)
```

Arguments

DeviceName	STAPI devicename identifier. This should match the identifier used in STSECTOOLFUSE_Init.
TermParam	Pointer to a driver termination parameters structure. This parameter is currently unused and must have the value NULL.

Return Value

ST_NO_ERROR	No error occurred during the specified operation.
ST_ERROR_BAD_PARAMETER	A parameter passed to an API function has an invalid value.
STSECTOOLFUSE_ERROR_NOT_INITIALISED	The driver has not been initialised.

Comments

This function releases all resources allocated by the driver since STSECTOOLFUSE_Init was called.

CONFIDENTIAL

STSECTOOLFUSE_ReadItem()

Description

Reads a non-bulk fuse value.

Definition

```
ST_ErrorCode_t STSECTOOLFUSE_ReadItem(STSECTOOLFUSE_ITEM_t item, U32*
value)
```

Arguments

`item`	Fuse identifier. Must be a valid fuse identifier for this device and driver version.
`value`	Pointer to unsigned integer to hold the fuse value. Must be non-NULL

Return Value

`ST_NO_ERROR`	No error occurred during the specified operation.
`ST_ERROR_BAD_PARAMETER`	A parameter passed has an invalid value
`STSECTOOLFUSE_ERROR_OTP_DATA_ERROR`	A hardware error occurred while reading OTP data. Refer to device's fuse map to check if the fuse is write-only.
`STSECTOOLFUSE_ERROR_NOT_OTP_ITEM_ERROR`	Trying to read a fuse item, using `STSECTOOLFUSE_ReadItem()`, which cannot be read by this function. E.g. a bulk fuse
`STSECTOOLFUSE_ERROR_NOT_INITIALISED`	An attempt was made to call an API function when the driver has not been initialised.

Comments

Fuses may be up to 16 bits in size. If the fuse size is smaller than 32-bits in size, the most-significant unused bits will be zero.

See Also

`STSECTOOLFUSE_WriteItem()`

CONFIDENTIAL

STSECTOOLFUSE_WriteItem()

Description

Writes a non-bulk fuse value.

Definition

```
ST_ErrorCode_t STSECTOOLFUSE_WriteItem(STSECTOOLFUSE_ITEM_t item, U32
value)
```

Arguments

`item`	Fuse identifier.
`value`	Unsigned value to set the fuse to.

Return Value

`ST_NO_ERROR`	No error occurred during the specified operation.
`ST_ERROR_BAD_PARAMETER`	A parameter passed has an invalid value
`STSECTOOLFUSE_ERROR_OTP_DATA_ERROR`	There was a problem reading back the data to check if the write succeeded. The fuse may be write only or there may be another issue.
`STSECTOOLFUSE_ERROR_NOT_INITIALISED`	An attempt was made to call an API function when the driver has not been initialised.
`STSECTOOLFUSE_ERROR_WRITE_NOT_ENABLED`	The fuse is not writable.
`STSECTOOLFUSE_ERROR_OTP_WRITE_ERROR`	There was a problem writing the data. This could be that verify after writing gave the wrong value or that the fuse is set to read-only.
`STSECTOOLFUSE_ERROR_NOT_INITIALISED`	The driver has not been initialised.

Comments

Non-bulk fuse items may be written more than once, although no bits that are already set to 1 may be reset to 0. Hence it is recommended, if a single bit needs to be set in a multi-bit fuse, that the fuse value is first read using `STSECTOOLFUSE_ReadItem`, bitwise OR-ed with the bit to be set, then the combined value written using this function.

Fuses may be up to 16 bits in size. If the fuse size is smaller than 32-bits in size, the most-significant unused bits will be ignored when writing the value.

After writing the fuse value, this function reads the fuse back to ensure that the value has been correctly written. If the read value does not match the written value it will return `STSECTOOLFUSE_ERROR_OTP_WRITE_ERROR`. Note: If the fuse is a non-readable fuse, the driver will not attempt to read it.

See Also

`STSECTOOLFUSE_ReadItem()`

CONFIDENTIAL

STSECTOOLFUSE_BulkReadItem()

Description

This function reads all or part of a multi-byte bulk fuse value. These fuses are 32-bits or greater in size and may only be written once.

Definition

```
ST_ErrorCode_t STSECTOOLFUSE_BulkReadItem(STSECTOOLFUSE_ITEM_t item, U32
location, U32 size, U8* data)
```

Arguments

`item`	An enumeration of type, STSECTOOLFUSE_ITEM_t (see stsectoolfuse.h).
`location`	Only applicable to item, STSECTOOLFUSE_BULK_OTP_OTP_ITEM. For otherbulk fuses, set to zero. Provides subscript into bulk_otp. Unit, bits. Must be a multiple of 128.
`size`	Number of bits to read. For all bulk fuses other than, STSECTOOLFUSE_BULK_OTP_OTP_ITEM, it must be set to fuse size (see Appendix A). For bulk_otp, size is set in multiples of 128.
`data`	Pointer to a buffer to read the fuse data into. The buffer must be at least `size` bits. Note byte order: Data byte zero, data[0], becomes content of fuse address zero, fuseAddr[0].

Return Value

`ST_NO_ERROR`	No error occurred during the specified operation.
`ST_ERROR_BAD_PARAMETER`	A parameter passed has an invalid value
`ST_ERROR_FEATURE_NOT_SUPPORTED`	Bulk fuses are not supported on the current device
`STSECTOOLFUSE_ERROR_OTP_DATA_ERROR`	There was a problem reading back the data to check if the write succeeded. The fuse may be write only or there may be another issue.
`STSECTOOLFUSE_ERROR_NOT_INITIALISED`	An attempt was made to call an API function when the driver has not been initialised.
`STSECTOOLFUSE_ERROR_NOT_INITIALISED`	The driver has not been initialised.

Comments

Location and size are bit values.

See Also

`STSECTOOLFUSE_BulkWriteItem()`

CONFIDENTIAL

STSECTOOLFUSE_BulkWriteItem()

Description

This function writes all or part of a multi-byte bulk fuse value. These fuses are 32-bits or greater in size and may only be written once.

Definition

```
ST_ErrorCode_t STSECTOOLFUSE_BulkWriteItem(STSECTOOLFUSE_ITEM_t item, U32
location, U32 size, U8* data)
```

CONFIDENTIAL

Arguments

`item`	An enumeration of type, STSECTOOLFUSE_ITEM_t (see stsectoolfuse.h).
`location`	Only applicable to item, STSECTOOLFUSE_BULK_OTP_OTP_ITEM. For otherbulk fuses, set to zero. Provides subscript into bulk_otp. Unit, bits. Must be a multiple of 128.
`size`	Number of bits to write. For all bulk fuses other than, STSECTOOLFUSE_BULK_OTP_OTP_ITEM, it must be set to fuse size (see Appendix A). For bulk_otp, size is set in multiples of 128.
`data`	Pointer to a buffer containing the data to write into the fuse. The buffer must be at least `size` bits. Note byte order: Data byte zero, data[0], is written to fuse address zero, fuseAddr[0].

Return Value

`ST_NO_ERROR`	No error occurred during the specified operation.
`ST_ERROR_BAD_PARAMETER`	A parameter passed has an invalid value
`ST_ERROR_FEATURE_NOT_SUPPORTED`	Bulk fuses are not supported on the current device
`STSECTOOLFUSE_ERROR_OTP_DATA_ERROR`	There was a problem writing back the data to check if the write succeeded. The fuse may be write only or there may be another issue.
`STSECTOOLFUSE_ERROR_NOT_INITIALISED`	An attempt was made to call an API function when the driver has not been initialised.
`STSECTOOLFUSE_ERROR_WRITE_NOT_ENABLED`	The fuse is not writable.
`STSECTOOLFUSE_ERROR_OTP_WRITE_ERROR`	There was a problem writing the data. This could be that verify after writing gave the wrong value or that the fuse is set to read-only. This error will occur for a bulk fuse which has been locked.
`STSECTOOLFUSE_ERROR_NOT_INITIALISED`	The driver has not been initialised.

Comments

Location and size are bit values. Bulk fuses can only be written once. Their value is then locked and cannot be modified.

See Also

`STSECTOOLFUSE_BulkReadItem()`

CONFIDENTIAL

6 Appendix A - fuse definitions

This appendix lists all fuses available in this driver version and their properties. The fuses available vary by device. Later revisions of the STSECTOOLFUSE driver may also make additional fuses available.

The table(s) below show which fuses are linked with which `STSECTOOLFUSE_Item_t` values in the STSECTOOLFUSE API.

NOTE: For many fuse items the corresponding name in the device fuse map is also given to allow the fuses to be matched against a device fuse map. However, since names in fuse maps are sometimes subject to change, there may be some differences between the names given here and the fuse map. In such cases the name in the fuse map name should be considered the correct name.

7105 fuses

STSECTOOLFUSE_Item_t	Fuse name	Size (bits)	Access	Type
STSECTOOLFUSE_JTAG_PROT_OTP_ITEM	jtag_protect	4	RW	Bit
STSECTOOLFUSE_ENGINEERING_TEST_000_OTP_ITEM	engineering_test_000	1	RO	Bit
STSECTOOLFUSE_TRANS_CW_SECURE_OTP_ITEM	trans_cw_secure	1	RW	Bit
STSECTOOLFUSE_TRANS_CW_ENABLE_OTP_ITEM	trans_cw_enable	1	RO	Bit
STSECTOOLFUSE_CRYPT_CPU0_IFETCH_SRC_RST_OTP_ITEM	crypt_cpu0_ifetch_src_rst	1	RW	Bit
STSECTOOLFUSE_CRYPT_CPU1_IFETCH_SRC_RST_OTP_ITEM	crypt_cpu1_ifetch_src_rst	1	RW	Bit
STSECTOOLFUSE_CRYPT_CPU2_IFETCH_SRC_RST_OTP_ITEM	crypt_cpu2_ifetch_src_rst	1	RW	Bit
STSECTOOLFUSE_CRYPT_SIGDMA_SRC_RST_OTP_ITEM	crypt_sigdma_src_rst	1	RW	Bit
STSECTOOLFUSE_CRYPT_SIGCHK_SRC_RST_OTP_ITEM	crypt_sigchk_src_rst	1	RW	Bit
STSECTOOLFUSE_CRYPT_WATCHDOG_SRC_RST_OTP_ITEM	crypt_watchdog_src_rst	1	RW	Bit
STSECTOOLFUSE_CRYPT_HASH_INCLUDE_ADDRESS_OTP_ITEM	crypt_hash_include_addr	1	RW	Bit
STSECTOOLFUSE_CRYPT_SIGCHK_ENABLE_OTP_ITEM	crypt_sigchk_enable	1	RW	Bit
STSECTOOLFUSE_MES0_ENABLE_OTP_ITEM	mes0_enable	1	RW	Bit
STSECTOOLFUSE_MES0_SRC_ID_MON_ENABLE_OTP_ITEM	mes0_src_id_mon_enable	1	RW	Bit
STSECTOOLFUSE_MES0_ENCRYPT_ALL_ENABLE_OTP_ITEM	mes0_encrypt_all_enable	1	RW	Bit
STSECTOOLFUSE_T1_FILTER_ENABLE_OTP_ITEM	t1_filter_enable	1	RW	Bit
STSECTOOLFUSE_DIRT_DISABLE_OTP_ITEM	dirt_disable	1	RW	Bit
STSECTOOLFUSE_ENGINEERING0_OTP_ITEM	engineering_0	16	RO	Bit
STSECTOOLFUSE_ENGINEERING1_OTP_ITEM	engineering_1	16	RO	Bit

CONFIDENTIAL

STSECTOOLFUSE_Item_t	Fuse name	Size (bits)	Access	Type
STSECTOOLFUSE_ENGINEERING2_OTP_ITEM	engineering_2	16	RO	Bit
STSECTOOLFUSE_ENGINEERING3_OTP_ITEM	engineering_3	16	RO	Bit
STSECTOOLFUSE_METAL_FIX_NB_OTP_ITEM	metal_fix_nb	4	RO	Bit
STSECTOOLFUSE_PROC_TYPE_OTP_ITEM	proc_type	3	RO	Bit
STSECTOOLFUSE_FAB_LOC_OTP_ITEM	fab_loc	5	RO	Bit
STSECTOOLFUSE_CUSTOMER_OTP0_OTP_ITEM	customer_otp0	16	RW	Bit
STSECTOOLFUSE_CUSTOMER_OTP1_OTP_ITEM	customer_otp1	16	RW	Bit
STSECTOOLFUSE_CUSTOMER_OTP2_OTP_ITEM	customer_otp2	16	RW	Bit
STSECTOOLFUSE_CUSTOMER_OTP3_OTP_ITEM	customer_otp3	16	RW	Bit
STSECTOOLFUSE_SOC_PUBLIC_ID_OTP_ITEM	soc_public_id	64	RO	Bulk
STSECTOOLFUSE_CRYPT_STC_OTP_ITEM	crypt_stc	32	RW	Bulk
STSECTOOLFUSE_FULL_SALES_TYPE_OTP_ITEM	full_sales_type	32	RO	Bulk
STSECTOOLFUSE_GP_BULK128_0_OTP_ITEM	gp_bulk128_0	128	RW	Bulk
STSECTOOLFUSE_GP_BULK128_1_OTP_ITEM	gp_bulk128_1	128	RW	Bulk
STSECTOOLFUSE_BULK_OTP_OTP_ITEM	bulk_otp	2048	RW	Bulk
STSECTOOLFUSE_PCI_DISABLE_OTP_ITEMl	pci_disable	1	RW	Bit

CONFIDENTIAL

7 Appendix B - Code examples

7.1 Reading a non-bulk fuse

The following code snippet shows the function calls required to read the 4-bit non-bulk fuse STSECTOOLFUSE_JTAG_PROT_OTP_ITEM. This fuse is not available on all devices, but the method remains the same.

```
U32 value;
ST_ErrorCode_t err;

/* Initialise the driver */
err = STSECTOOLFUSE_Init(“SECTOOLFUSE”, NULL, STSECTOOLFUSE_NOINT);
if (ST_NO_ERROR != err)
{
    printf(“Error 0x%x at line %d\n”, err, __LINE__);
}

/* Read the fuse */
err = STSECTOOLFUSE_ReadItem(STSECTOOLFUSE_JTAG_PROT_OTP_ITEM, &value);
if (ST_NO_ERROR != err)
{
    printf(“Error 0x%x at line %d\n”, err, __LINE__);
}
else
{
    /* Since this is a 4-bit fuse, the 28-MS bits of value will be zero and
the fuse value will be in the 4-LS bits */
    printf(“Read fuse value 0x%x\n”, value);
}

/* Terminate the driver */
err = STSECTOOLFUSE_Term(“SECTOOLFUSE”, NULL);
if (ST_NO_ERROR != err)
{
    printf(“Error 0x%x at line %d\n”, err, __LINE__);
}
```

7.2 Writing a non-bulk fuse

The following code snippet shows the function calls required to write the 4-bit non-bulk fuse STSECTOOLFUSE_JTAG_PROT_OTP_ITEM, setting bit 2. This fuse is not available on all devices, but the method remains the same.

```
U32 value;
ST_ErrorCode_t err;

/* Initialise the driver */
err = STSECTOOLFUSE_Init(“SECTOOLFUSE”, NULL, STSECTOOLFUSE_NOINT);
if (ST_NO_ERROR != err)
{
    printf(“Error 0x%x at line %d\n”, err, __LINE__);
}
```

CONFIDENTIAL

```
/* Read the fuse */
err = STSECTOOLFUSE_ReadItem(STSECTOOLFUSE_JTAG_PROT_OTP_ITEM, &value);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}

/* OR the read value with 0x4 to set bit 2*/
value |= 0x4;

/* Write the combined value to set bit 2 of the fuse */
err = STSECTOOLFUSE_WriteItem(STSECTOOLFUSE_JTAG_PROT_OTP_ITEM, value);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}

/* Terminate the driver */
err = STSECTOOLFUSE_Term("SECTOOLFUSE", NULL);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}
```

7.3 Reading a bulk fuse

The following code snippet shows the function calls required to read the 32-bit bulk fuse STSECTOOLFUSE_CRYPT_STC_OTP_ITEM. This fuse is not available on all devices, but the method remains the same.

```
U8 value[4];
ST_ErrorCode_t err;

/* Initialise the driver */
err = STSECTOOLFUSE_Init("SECTOOLFUSE", NULL, STSECTOOLFUSE_NOINT);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}

/* Read the fuse */
err = STSECTOOLFUSE_BulkReadItem(STSECTOOLFUSE_CRYPT_STC_OTP_ITEM, 0, 32,
value);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}
else
{
    printf("Read fuse value: 0x%.2x 0x%.2x 0x%.2x 0x%.2x\n", value[0],
value[1], value[2], value[3]);
}
```

CONFIDENTIAL

```
/* Terminate the driver */
err = STSECTOOLFUSE_Term("SECTOOLFUSE", NULL);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}
```

7.4 Writing a bulk fuse

The following code snippet shows the function calls required to write the 32-bit bulk fuse STSECTOOLFUSE_CRYPT_STC_OTP_ITEM with value 0x01234567. This fuse is not available on all devices, but the method remains the same.

```
/* Since this is a little-endian device, the integer value 0x01234567 is
stored as a byte array as shown below */
U8 value[4] = {0x67,0x45,0x23,0x01};
ST_ErrorCode_t err;

/* Initialise the driver */
err = STSECTOOLFUSE_Init("SECTOOLFUSE", NULL, STSECTOOLFUSE_NOINT);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}

/* Write the fuse. Note that no read-modify-write cycle is used since bulk
fuses cannot be written more than once */
err = STSECTOOLFUSE_BulkWriteItem(STSECTOOLFUSE_CRYPT_STC_OTP_ITEM, 0, 32,
value);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}

/* Terminate the driver */
err = STSECTOOLFUSE_Term("SECTOOLFUSE", NULL);
if (ST_NO_ERROR != err)
{
    printf("Error 0x%x at line %d\n", err, __LINE__);
}
```

CONFIDENTIAL

CONFIDENTIAL

Information furnished is believed to be accurate and reliable. However, STMicroelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No licence is granted by implication or otherwise under any patent or patent rights of STMicroelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. STMicroelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of STMicroelectronics.

The ST logo is a trademark of STMicroelectronics

Copyright 2010 STMicroelectronics - All Rights Reserved

STMicroelectronics GROUP OF COMPANIES

CONFIDENTIAL