

## ARM-based Microcontroller

### WRITE/ERASE SECURITY

In the NXP ARM based microcontroller, each flash sector can be secured against write/erase operation. This feature allows the application to secure the sensitive code/data against tempering. If the Code Read Protection feature is enabled then the sensitive code/data is also protected against unauthorized reads. Please refer to the appropriate user manual for a description of the Code Read Protection feature.

Altering the contents of a secured sector is not possible. The write or erase operation (via ISP or IAP) on a secured sector will succeed but the internal erase/programming voltages to alter the flash sector contents are not generated. The write/erase security feature is irreversible. It should only be activated in the final design phase to avoid unusable parts/application boards. If the Code Read Protection is enabled along with the write/erase security on sector 0 then the Code Read Protection feature is also irreversible. However an application executing from the internal memory will still be able to communicate with the external world, if implemented by the system designer. The flash sectors which are not secured against write/erase can be erased/reprogrammed by such application. A write/erase operation spanning across a secured and an unsecured sectors will not alter the contents of either secured or unsecured sectors.

The write/erase security is enabled by calling the `write_erase_secure_user_sector()` function. This function is available in the library file. A 'C' header file (`write_erase_secure.h`) is also provided. While sectors are being secured the flash memory is not available for code execution hence it is very important to relocate functions in RAM during run time. Please refer to the linker documentation for details. The following section describes the functions available in the library. The function return codes are described in Table 4.

#### Current Release

`write_erase_secure_2_00_LPC_23xx_24xx_ADS_1_2.a`

`write_erase_secure_2_00_LPC_23xx_24xx_gcc_3_4_2.a`

#### write\_erase\_secure Library Functions

`write_erase_secure_user_sector(unsigned start, unsigned end, unsigned cclk)`

Function	<code>write_erase_secure_user_sector(unsigned start, unsigned end, unsigned cclk)</code>
Input Parameters	start: Start Sector Number end: End Sector Number: Should be greater than or equal to start sector number. cclk: System Clock Frequency (cclk) in kHz.
Return Code	SUCCESS   INVALID_SECTOR   SECTOR_ALREADY_SECURED   NOT_EXECUTING_IN_RAM   WRONG_PART
Description	This function is used to enable the write/erase security on one or more sectors of on-chip Flash memory. A write/erase operation spanning across a secured and an unsecured sectors will <b>not</b> alter the contents of either secured or unsecured sectors. Sector security is effective after a reset cycle.

Table 1: `write_erase_secure_user_sector` function description

## ARM-based Microcontroller

### write\_erase\_secure\_boot\_sector(unsigned cclk)

<b>Function</b>	write_erase_secure_boot_sector(unsigned cclk)
<b>Input Parameters</b>	cclk: System Clock Frequency (cclk) in kHz.
<b>Return Code</b>	SUCCESS   SECTOR_ALREADY_SECURED   NOT_EXECUTING_IN_RAM   WRONG_PART
<b>Description</b>	This function is used to enable the write/erase security on the boot sector of on-chip Flash memory. Sector security is effective after a reset cycle.

**Table 2: write\_erase\_secure\_boot\_sector function description**

### write\_erase\_secure\_get\_version(void)

<b>Function</b>	write_erase_secure_get_version(void)
<b>Input Parameters</b>	none
<b>Return Code</b>	Library version number(0x0000xxyy). Where xx is the major and yy is the minor version number.
<b>Description</b>	This function is used to get the write_erase_secure library version number.

**Table 3: write\_erase\_secure\_get\_version function description**

Return Code	Mnemonic	Description
300	SUCCESS	Function is executed successfully.
301	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
302	SECTOR_ALREADY_SECURED	One or more sectors are already secured.
303	NOT_EXECUTING_IN_RAM	write_erase_secure Library function is not executing in RAM.
304	WRONG_PART	write_erase_secure Library function is executing in wrong part.

**Table 4: write\_erase\_secure Library Functions Return Codes Summary**