# R_<nnnnn>

## LPC18xx/LPC43xx SPIFI software library

Rev. 0.01 — 27 April 2012 Report

DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DR DRAF D DRA

### Document information

| Info | Content |
|---|---|
| **Keywords** | LPC18xx, LPC43xx |
| **Abstract** | This document describes the SPI Flash Interface (SPIFI) software library |

DRAFT DRAFT DRAFT DRAFT DRAFT

## Revision history

| Rev | Date | Description |
|---|---|---|
| 0.1 | <tbd> | Preliminary version |

# Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT

## 1. Introduction

This document describes the support functions for the SPI Flash Interface (SPIFI) provided for NXP microcontrollers that include SPIFI.

## 2. Supported devices

Serial flash devices with the following features are supported:

- Read JDEC ID
- Page programming
- at least one command with uniform erase size throughout the device

Table 1 shows a list of vendor QSPI devices which are verified to support the SPIFI API. Other devices can be used and will run in basic single SPI mode at lower speed.

**Remark:** All QSPI devices have been tested at an operating voltage of 3.3 V.

**Table 1. Supported QSPI devices**

| Manufacturer | Device name |
|---|---|
| AMIC | A25L512, A25L010, A25L020, A25L040, A25L080, A25L016, A25L032, A25LQ032 |
| Atmel | AT25F512B, AT25DF021, AT25DF041A, AT25DF081A, AT25DF161, AT25DQ161, AT25DF321A, AT25DF641 |
| Chingis | Pm25LD256, Pm25LD512, Pm25LD010, Pm25LD020, Pm25LD040, Pm25LQ032 |
| Elite (ESMT) | F25L08P, F25L16P, F25L32P, F25L32Q |
| Eon | EN25F10, EN25F20, EN25F40, EN25Q40, EN25F80, EN25Q80, EN25QH16, EN25Q32, EN25Q64, EN25Q128 |
| Gigadevice | GD25Q512, GD25Q10, GD25Q20, GD25Q40, GD25Q80, GD25Q16, GD25Q32, GD25Q64 |
| Macronix | MX25L8006, MX25L8035, MX25L8036, MX25U8035[1], MX25L1606, MX25L1633, MX25L1635, MX25L1636, MX25U1635[1], MX25L3206, MX25L3235, MX25L3236, MX25U3235[1], MX25L6436, MX25L6445, MX25L6465, MX25L12836, MX25L12845, MX25L12865, MX25L25635, MX25L25735 |
| Numonyx | M25P10, M25P20, M25P40, M25P80, M25PX80, M25P16, M25PX16, M25P32, M25PX32, M25P64, M25PX64, N25Q032, N25Q064, N25Q128 |
| Spansion | S25FL004K, S25FL008K, S25FL016K, S25FL032K, S25FL032P, S25FL064K, S25FL064P, S25FL129P |
| SST | SST26VF016, SST26VF032, SST25VF064 |
| Winbond | W25Q40, W25Q80, W25Q16, W25Q32, W25Q64 |

[1] Level translation circuitry, which might affect performance, is required for these parts.

The following devices lack one or more of these features and are not supported:

Elite: F25L004, F25L008, F25L016.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT

Eon: 25B64.

SST: 25VF512, 25WF512, 25VF010, 25WF010, 25LF020, 25VF020, 25WF020, 25VF040, 25WF040, 25VF080, 25WF080, 25VF016, 25VF032.

# 3. SPIFI hardware

The LPC18xx/LPC43xx microcontrollers define a base address for the SPIFI registers and a base address for the memory area in which the serial Flash connected to the SPIFI can be read.

The first operation with the serial Flash is Read JEDEC ID, which is implemented by most serial Flash devices. Depending on the device identity code returned by the serial Flash in this operation, device-specific commands are used for further operation. Programming and other operations on the serial Flash are performed by API calls as described in this document.

# 4. SPIFI software library

## 4.1 SPIFI function allocation

Table 2 shows an overview of the SPIFI API calls. For details see Section 4.2.

**Table 2. SPIFI function allocation**

| Function | Description |
|---|---|
| spifi_init | This call sends the standardized JEDEC ID command to the attached serial flash device. If a serial flash responds with an ID known to the SPIFI API, it is set up for operation as standard memory.<br>Parameter0 - Pointer to `SPIFIobj`<br>Parameter1 - (minimum clock cycles with CS pin HIGH) - 1<br>Parameter2 - SPIFI options<br>Parameter3 - Serial clock rate<br>Return - SPIFI error code |
| spifi_program | This call programs `length` bytes in the serial flash. `obj` must point to the object returned by the preceding `spifi_init` call.<br>Parameter0 - Pointer to the object returned by the preceding `spifi_init` call.<br>Parameter1 - Source address (in RAM or other memory) of the data to be programmed.<br>Parameter2 - Number of bytes to be programmed.<br>Return - SPIFI error code. |
| spifi_erase | This command can be used to erase sections of the serial flash. It is not needed for re-programing because `spifi_program` automatically erases as necessary. `obj` should point to the object returned by the preceding `spifi_init` call.<br>Parameter0 - Pointer to the object returned by the preceding spifi_init call.<br>Parameter1 - SPIFI memory area to be erased.<br>Return - SPIFI error code |

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DR DRAF DRAFT D DRA

## 4.2 SPIFI function calls

### 4.2.1 Calling the SPIFI driver

**Remark:** Compile any module that calls the SPIFI API with the compiler set for ARM ABI compatibility. This is the default in most compilers.

### 4.2.2 SPIFI initialization call spifi_init

The SPIFI initialization API call sends the standardized Read JEDEC ID command to the attached serial flash device. If a serial flash responds, it is set up for reading in ARM memory space.

```
int spifi_init (SPIFIobj *obj, unsigned csHigh, unsigned options, uclnsigned MHz)
```

After a `spifi_init` call that returns one of the unknown error codes (0x20009 to 0x20006, see Table 4), the caller can read and check the SPIFI memory area but should not issue any `spifi_program` or `spifi_erase` calls because not enough is known about the device to accomplish these tasks.

`spifi_init` can be called repeatedly in order to change some of its operands. The subsequent call need not use the same `SPIFIobj`, and need not use the same version of the driver as the preceding call. The only case in which problems should arise with reusing `spifi_init` is if the SPIFI and microcontroller hardware has been reset but the serial flash hardware has not (since most serial flashes don't have a Reset pin).

#### Parameter0 obj

`obj` points to an area of memory large enough to receive the object created by spifi_init. The space required for the SPIFI object is 192 bytes.

#### Parameter1 csHigh

`csHigh` is one less than the minimum number of clock cycles with the CS pin HIGH, that the SPIFI should maintain between commands. Compute this parameter from the SPIFI clock period and the minimum HIGH time of CS from the serial flash data sheet:

`csHigh` = ceiling(min CS HIGH / SPIFI_CLK ) - 1

where ceiling means round up to the next higher integer if the argument isn't an integer.

#### Parameter2 options

`options` contains 10 bits controlling the binary choices shown in Table 3. `options` can be 0 or any AND or OR combination of the bits represented in Table 3. An optional use of names for the enumeration of bit values is also shown.

Table 3. Bit values for spifi_init options parameter

| Bit | Value | Description | Name |
|---|---|---|---|
| 0 | | SCL output mode | |
| | 0 | SCL is low when a frame/command is not in progress. | S_MODE0 |
| | 1 | The SCL output is high when a frame/ command is not in progress.<br>Note that S_MODE3+ S_FULLCLK+S_RCVCLK is not allowed.<br>Use S_MODE0 or S_INTCLK. | S_MODE3 |

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT

Table 3. Bit values for spifi_init options parameter

| Bit | Value | Description | Name |
|---|---|---|---|
| 1 | | SPIFI read mode | |
| | 0 | The fastest read operation provided by the device will be used. | S_MAXIMAL |
| | 1 | SPI mode and the slowest, most basic/ compatible read operation will be used. | S_MINIMAL |
| 5:2 | 0 | Reserved | - |
| 6 | | Sampling edge | - |
| | 0 | Data from the serial flash is sampled on rising edges of the SCL output, as in classic SPI applications. Suitable for slower clock rates. | S_HALFCLK |
| | 1 | Data from the serial flash is sampled on falling edges on the SCL output, allowing a full clock period for the serial flash to present each bit or group of bits. | S_FULLCLK |
| 7 | | Sampling clock | |
| | 0 | Data is sampled using the internal clock from which the SCL pin is driven. | S_INTCLK |
| | 1 | Data is sampled using the SCL clock fed back from the pin. This allows more time for the serial flash to present each bit or group of bits, but when used with S_FULLCLK can endanger hold time for data from the flash. | S_RCVCLK |
| 8 | | SPIFI mode | |
| | 0 | If the device can operate in quad mode, quad mode will be used, else SPI mode. | - |
| | 1 | If the connected device can operate in dual mode (2 bits per clock), dual mode will be used, else SPI mode. | S_DUAL |
| 9 | 0 | Reserved | - |

### Parameter3 MHz

`MHz` is the serial clock rate divided by 1000000, rounded to an integer. It is used for devices that allow a variable number of dummy bytes between the address and the read data in a memory read command. This operand is only required for some Numonyx and Winbond quad devices, but it is good practice to include it in all spifi_init calls.

### Return

A return value of zero indicates success. Non-zero error codes are listed in Table 4

Table 4. Error codes for spifi_init

| Error code | Description |
|---|---|
| 0x2000A | No operative serial flash (JEDEC ID all zeroes or all ones) |
| 0x20009 | Unknown manufacturer code |
| 0x20008 | Unknown device type code |
| 0x20007 | Unknown device ID code |
| 0x20006 | Unknown extended device ID value |
| 0x20005 | Device status error |
| 0x20004 | Operand error: S_MODE3 + S_FULLCLK + S_RCVCLK selected in `options` |

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT

### 4.2.3 SPIFI program call spifi_program

The SPIFI program API call programs `opers.length` bytes in the serial flash.

```
int spifi_program (SPIFIobj *obj,  char *source,  SPIFIopers *opers)
```

A `spifi_program` call with source equal to `opers.dest` and `opers.options` not including S_FORCE_ERASE will not do any erasing nor programming, since the data at `opers.dest` is equal to the data at source. Such a call can be used to protect or unprotect sector(s) depending on the value of `opers.protect`.

#### Parameter0 obj

`obj` points to the object returned by the preceding `spifi_init` call.

#### Parameter1 source

`source` is the address in RAM or other memory of the data to be programmed.

#### Parameter2 opers

Parameter2 is defined through the `SPIFIopers` C struct (see Section 4.2.5). `opers.length` is the length of bytes to be programmed in the serial flash. `opers.dest` is the destination address of the data in the SPIFI memory, and `opers.options` defines the options for programming the SIFI.

#### Return

`spifi_program` does not return until programming and erasure have been completed or an error is encountered. A return value of zero indicates success. Non-zero error codes are listed in Table 5.

**Table 5. Error codes for spifi_program and spifi_erase**

| Error code | Description |
|---|---|
| 0x20007 | Programming and erasure cannot be done because the serial flash was not identified in the `spifi_init` operation. |
| 0x20005 | Device status error |
| 0x20004 | Operand error: the `dest` and/or `length` operands were out of range. See <tbd>Address operands and checking below. |
| 0x20003 | Time-out waiting for program or erase to begin: protection could not be removed. |
| 0x20002 | Internal error in API code. |
| 0x2000B | S_CALLER_ERASE is included in options, and erasure is required. |
| other | Other non-zero values can occur if `options` selects verification. They will be the address in the SPIFI memory area at which the first discrepancy was found. |

### 4.2.4 SPIFI erase call spifi_erase

The `spifi_erase` call can be used instead of the `spifi_program` call to speed up erasing large memory areas. Since erasing is also done by `spifi_program`, the `spifi_erase` call is not strictly necessary.

```
int spifi_erase (SPIFIobj *obj, SPIFIopers *opers)
```

#### Parameter0 obj

`obj` points to the object returned by the preceding `spifi_init` call.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DR DRAF D DRA

### Parameter1 opers

Parameter1 is defined through the `SPIFIopers` C struct (see Section 4.2.5).

The code will use the largest unit(s) of erasure it can to accomplish the indicated operation and will use the opers.scratch area only when required by a starting or ending address that is not a multiple of the smallest available erase size. The driver will attempt to remove any protection on the sector(s) indicated by opers.dest and opers.length. If this removal succeeds, the opers.protect value determines the protection of the sector(s) on return, as described in Section 4.2.7.

DRAFT DRAFT DRAFT DRAFT DRAFT

### Return

Return values are the same as for spifi_program. A return value of zero indicates success. Non-zero error codes are listed in Table 5

## 4.2.5 SPIFI operands for program and erase

`SPIFIopers` is a C struct that contains operands for the `spifi_program` and `spifi_erase` calls.

```
typedef struct {
    char *dest;/* starting address for programming or erasing */
    unsigned length;/* number of bytes to be programmed or erased */
    char *scratch;/* address of work area or NULL */
    int protect;/* protection to apply after programming/erasing is done */
    unsigned options;/* see the table below */
} SPIFIopers;
```

`dest` specifies the first address to be programmed or erased, either in the SPIFI memory area or as a zero-based device address. If `dest` is not a multiple of the smallest sector size that's uniformly available throughout the serial flash, the first part of the first sector is one of the following:

- Preserved if a scratch address is provided and/or an erase isn't needed for the first sector.
- Erased to all ones if scratch is NULL and an erase is needed for the first sector.

Similarly, if `dest` plus `length` is not a multiple of the sector size, the last part of the last sector is one of the following:

- Preserved if scratch is non-zero and/or an erase isn't needed for the last sector.
- Erased to all ones if scratch is zero and an erase is needed for the last sector.

For either `spifi_program` or `spifi_erase`, scratch should be NULL or the address of an area of RAM that the SPIFI driver can use to save data during erase operations. If provided, the scratch area should be as large as the smallest erase size that is available throughout the serial flash device. If scratch is NULL (zero) and an erase is necessary, any bytes in the first erase block before `dest` are left in erased state (all ones), as are any bytes in the last erase block after `dest + length`.

The driver uses the least number of bytes possible in the scratch area. If `dest` and `dest + length - 1` are in separate erase blocks, the driver will use the larger of (the number of bytes before `dest` in the first erase block) and (the number of bytes after (`dest + length`) in the last erase block). If only one erase block is involved, the driver will use the sum of these two numbers.

`options` contains 10 bits controlling the binary choices shown in Table 6. `options` can be 0 or any AND or OR combination of the bits represented in Table 6. An optional use of names for the enumeration of bit values is also shown.

Unless `options` includes S_CALLER_PROT, the driver attempts to remove write-protection on the sector(s) implied by `dest` and `length`.

The protect operand indicates whether the driver should protect the sector(s) after programming is completed. See Section 4.2.7 for details of the protect value.

All information provided in this document is subject to legal disclaimers.
© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT

**Table 6. Bit values for SPIFIopers options parameter**

| Bit | Value | Description | Name |
|---|---|---|---|
| 1:0 | 0 | Reserved | - |
| 2 | | Erase mode | - |
| | 0 | Erasing is done when necessary. | S_ERASE_AS_REQD |
| | 1 | All sectors in `dest` to `dest` + `length` will be erased. | S_FORCE_ERASE |
| 3 | | Erase mode | |
| | 0 | Erasing is done when necessary. | S_ERASE_AS_REQD |
| | 1 | Erasing is handled by the caller not by the driver. | S_CALLER_ERASE |
| 4 | | Verify program | - |
| | 0 | No reading or checking will be done. | S_NO_VERIFY |
| | 1 | Data will be read back and checked after programming. | S_VERIFY_PROG |
| 5 | | Verify erase | - |
| | 0 | No reading or checking will be done. | S_NO_VERIFY |
| | 1 | Sectors will be read back and checked for 0xFF after erasing | S_VERIFY_ERASE |
| 8:6 | 0 | Reserved | - |
| 9 | | Write protection | - |
| | 0 | The driver removes protection before the operation and sets it as specified thereafter. | S_DRIVER_PROT |
| | 1 | Write protection is handled by the caller not by the driver. | S_CALLER_PROT |

### 4.2.6 Address operands and checking

For both `spifi_program` and `spifi_erase`, the `opers.dest` value can be either the (zero-based) address within the serial flash or an address in the SPIFI memory area. `opers.dest` and `opers.length` operands are always checked against the device size; when verification is requested, they are also checked against the allocated size of the SPIFI memory area.

### 4.2.7 Protection

Serial flash devices provide write-protection in several ways. Most devices simply have 2 to 5 bits in their status registers that specify what fraction of the device is write protected, possibly in conjunction with a bit that specifies whether the fraction is at top or bottom and/or a bit that specifies whether the fraction is protected or unprotected. For such devices, at the start of `spifi_program` or `spifi_erase` the driver simply saves the status byte, then clears all of the 2 to 5 bits, so that the whole device is write-enabled.

The `opers.protect` value of a `spifi_program` or `spifi_erase` on such a device can be 0 to leave the device fully write-enabled, -1 to restore the protection status saved at the start of the call, or any other non-zero value to set the protection status to that value. (Consult the device data sheet for the content of the latter value.)

Some serial flash devices use individual protection bits for each sector. These include SST quad devices, Atmel devices, and Macronix devices that provide a WPSEL command and on which such a command has been executed (Setting WPSEL is an irrevocable operation). Similarly to devices which include status register protection, -1 in the `opers.protect` value makes the driver restore protection to the state in effect before the

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT

call. 0 leaves the programmed/erased sector(s) write-enabled, and 1 write-protects them. For small (high and low) sectors on SST quad devices only, `opers.protect` can be 3 to read- and write-protect the sectors, or 2 to read-protect but write-enable them (Write Only Memory!). 2 and 3 work like 0 and 1 respectively for other sectors and other devices.

DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT

# 5. Legal information

## 5.1 Definitions

**Draft —** The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

## 5.2 Disclaimers

**Limited warranty and liability —** Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

**Right to make changes —** NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use —** NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications —** Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

**Export control —** This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

© NXP B.V. 2012. All rights reserved.

DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT DRAFT

# 6. Contents

**1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . 3**
**2 Supported devices. . . . . . . . . . . . . . . . . . . . . . . 3**
**3 SPIFI hardware. . . . . . . . . . . . . . . . . . . . . . . . . 4**
**4 SPIFI software library . . . . . . . . . . . . . . . . . . . 4**
4.1 SPIFI function allocation. . . . . . . . . . . . . . . . . . 4
4.2 SPIFI function calls. . . . . . . . . . . . . . . . . . . . . . 5
4.2.1 Calling the SPIFI driver. . . . . . . . . . . . . . . . . . . 5
4.2.2 SPIFI initialization call spifi_init. . . . . . . . . . . . . 5
Parameter0 obj . . . . . . . . . . . . . . . . . . . . . . . . .5
Parameter1 csHigh. . . . . . . . . . . . . . . . . . . . . . .5
Parameter2 options . . . . . . . . . . . . . . . . . . . . . .6
Parameter3 MHz . . . . . . . . . . . . . . . . . . . . . . . .6
Return. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7
4.2.3 SPIFI program call spifi_program . . . . . . . . . . . 7
Parameter0 obj . . . . . . . . . . . . . . . . . . . . . . . . .7
Parameter1 source . . . . . . . . . . . . . . . . . . . . . .7
Parameter2 opers. . . . . . . . . . . . . . . . . . . . . . . .7
Return. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7
4.2.4 SPIFI erase call spifi_erase . . . . . . . . . . . . . . . 8
Parameter0 obj . . . . . . . . . . . . . . . . . . . . . . . . .8
Parameter1 opers. . . . . . . . . . . . . . . . . . . . . . . .8
Return. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .9
4.2.5 SPIFI operands for program and erase . . . . . . 9
4.2.6 Address operands and checking . . . . . . . . . . 10
4.2.7 Protection . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
**5 Legal information. . . . . . . . . . . . . . . . . . . . . . . 12**
5.1 Definitions. . . . . . . . . . . . . . . . . . . . . . . . . . . 12
5.2 Disclaimers . . . . . . . . . . . . . . . . . . . . . . . . . . 12

**6 Contents 13**

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section ‘Legal information’.

**© NXP B.V. 2012. All rights reserved.**

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

**Date of release: 27 April 2012**
**Document identifier: <Document ID>**