AN12122

LPC540xx Image Header Structure

Rev. 1.1 — 27 August 2018 Application note

Document information

Info	Content
Keywords	LPC540xx, image header structure, XIP images, non-XIP images, Rev 0A, Rev 1B.
Abstract	This application note describes the various fields of the image header of LPC540xx devices that should be configured to execute XIP and non-XIP images.

Revision history

Rev	Date	Description
1.1	20180827	Updated Table:1 Image header for LPC540xx devices
1.0	20180327	Initial version

Contact information

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

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

1. Introduction

The LPC540xx is a family of ARM® Cortex®-M4 based microcontrollers for embedded applications, featuring a rich peripheral set with a very low power consumption and enhanced debug features.

The LPC540xx family includes 360 kB of on-chip SRAM, and a quad SPI Flash Interface (SPIFI) for expanding program memory. It also includes one high-speed and one full-speed USB host and device controller, Ethernet AVB, LCD controller, smart card interface, SD/MMC, CAN FD, an External Memory Controller (EMC), a DMIC subsystem with PDM microphone interface and I^{2}S, five general purpose timers, SCTimer/PWM, RTC/alarm timer, Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), ten flexible serial communication peripherals (USART, SPI, I^{2}S, I^{2}C interface), Secure Hash Algorithm (SHA), AES-256 engine, Physical Unclonable Function (PUF), 12-bit 5.0 Msamples/sec ADC, and a temperature sensor.

The application images constructed for LPC540xx devices have a header structure that determines the boot sequence of the image. This application note describes the image header structure and the method to configure the header parameters.

2. LPC540xx boot flow summary

LPC540xx has no internal flash for code and data storage. In systems with LPC540xx devices, the application images should reside in external devices like quad SPI and parallel flash memory. The images are downloaded upon reset or executes from an external memory. Images can be booted into on-chip SRAM from external flash (SPI, QSPI, or parallel flash). It can also be downloaded via the serial ports (UART, I^{2}C, SPI, USB0, and USB1). The code is then validated and the boot ROM will vector to on-chip SRAM.

Depending on the values of the OTP bits, ISP pins, and the image header type definition, the bootloader decides to download the code into the on-chip SRAM or run from external memory. After the boot mode is determined, and if the image is stored externally, the boot ROM copies the first 512 bytes from the image (vector components and image header) into the internal SRAMX. The first 512 bytes from the image are copied at the location 0x0000 0000, to validate the vector table and the image header. If the image is downloaded from a serial interface (via USART, I^{2}C, SPI, and USB), the complete image including the header is already loaded into SRAMX. After validating the image header, the application is executed.

3. Image header format

An application image consists of a header structure. See Section 2. See Table 1 for the format of the image header for LPC540xx devices.
Note: The quad SPI descriptor in image header is different in LPC540xx Rev 0A devices and Rev 1B devices.

Table 1. Image header for LPC540xx devices

Address	Size (bytes)	Register	Bit	Symbol	Description
0x00	4	Header_marker	-	-	Always set to 0xFEEDA5A5
0x04	4	image_type	0	CRC	0: Compute CRC 1: No CRC computation
			1	XIP	0: Load image. 1: XIP image. Once CRC check passes program control will jump to reset_vector.
0x08	4	load_address	-	-	Load address within internal SRAM (SRAMX or SRAM0) or 0x10000000 for SPIFI XIP and 0x80000000 for parallel flash XIP image. This address is used to set VTOR register before passing control to application.
0x0C	4	Image_length	-	-	Length should be actual length – 4 (Image length excludes CRC32 field). For load-image types, Image_length specifies the size of data to be copied in to SRAMX. For XIP-images, Image_length specifies number of bytes included in CRC32 calculation.
0x10	4	crc_value	-	-	CRC32 of image excluding this field. Image length excludes this field.
0x14	4	image_version	-	-	Sequentially increasing version number of the user application image or store Unix EPOCH time stamp in this field.
0x18	4	emc_timings			EMC static memory configuration settings, required for EMC boot. Set to 0 to use boot ROM default.
			3:0	-	See STATICWAITWEN register description.
			7:4	-	See STATICWAITOEN register description
			12:8	-	See STATICWAITRD register description.
			17:13	-	See STATICWAITPAGE register description
			22:18	-	See STATICWAITWR register description.
			27:23	-	See STATICWAITTURN register description.
			28	-	See PM field description in STATICONFIG register
			29	-	See PB field description in STATICONFIG register
			30	-	Extended wait, see PB field description in STATICONFIG register.
			31	-	Buffer enable. See PB field description in STATICONFIG register.
0x1C	4	spi_clock_freq	-	-	SPI and SPIFI flash device clock speed (MHz). If this value is 0, then default clock speed for SPI is 12MHz and 24MHz for SPIFI.

0x20	4	-	-	-	Reserved.
0x24	4	Image_marker	-	-	Always set to 0xEDDC94BD
0x28	4	-	-	-	Reserved.
0x2C	4	-	-	-	Reserved.
0x30	4	descriptor valid	-	-	0x00000000 - SPIFI descriptor present. 0xFFFFFFFF - no SPIFI descriptor. SPIFI flash accessed in SPI mode.
0x34	3	mfgId	-	-	JEDEC ID data
0x37	1	extCount	-	-	Number of extended bytes to check.
0x38	8	extId	-	-	Extended Data
0x40	4	caps	-	-	Capabilities supported.
			0	-	Supports dual read
			1	-	Supports dual write
			2	-	Supports quad read
			23:3	-	Unused. Must be set to 0
			27:24	-	Quad mode read dummy bytes (quad cycles/2). If in quad mode and the device has 6 dummy cycles for quad reads, set bits 27:24 to 3 bytes.
			31:28		Dual mode read dummy bytes (dual cycles/4) If in dual mode and the device has 4 dummy cycles for dual reads, set bits 31:28 to 1 byte.
0x44	2	Blks	-	-	Number of blocks. Must be 2^N. Can be 0 for LPC540xx Revision 1B.
0x46	2	-	-	-	Padding bytes. Must be 2^N. Can be 0 for LPC540xx Revision 1B.
0x48	4	blkSize	-	-	Size of block. Must be 2^N. Can be 0 for LPC540xx Revision 1B.
0x4C	4	-	-	-	Reserved. Set to 0.
0x50	2	pageSize	-	-	Page Size. Must be 2^N. Can be 0 for LPC540xx Revision 1B.
0x52	2	-	-	-	Padding bytes. Set to 0.
0x54	4	maxReadSize	-	-	Maximum read allowed in one operation. User must set this field to 0x3F00. Can be 0 for LPC540xx Revision 1B.
0x58	5	-	-	-	Padding bytes. Set to 0.
0x5D	1	initDeInitFxId	-	-	Specify initialization function. 0 - No functionality 1 - Clears bits 3-0 (“read latency control” field in some devices) register in status/configuration register 3. Also, requires getStatus/setStatus functions to be 24-bit operations.
0x5E	1	clearStatusFxId	-	-	Specify device status clearing function. 2 – No functionality – function stub

					3 – Send serial command 0x30 to clear status bits
0x5F	1	getStatusFxId	-	-	Specify device status read function. 4 – send commands 0x05 and 0x35 to obtain 16-bit status. Copy bits 5,6 to bits 25,24 of returned 32-bit status. 5 – send commands 0x05, 0x35 and 0x33 to obtain 24-bit status 6 – send command 0x05 to obtain 8-bit status 7 – send commands 0x05 and 0x35 to obtain 16-bit status 8 – send commands 0x05, 0x35 and 0x15 to obtain 24-bit status. 25 – send commands 0x05 and 0x15 to obtain 16-bit status (Revision 1B only)
0x60	1	setStatusFxId	-	-	Specify device status write function. 9 – send 16-bit data using command 0x01 to status (byte 0) and configuration (byte 1) registers. Use Write Enable command 0x06. 10 – send 24-bit data using command 0x01 to status register 0 (byte 0), configuration register 1 (byte 1), and configuration register 2 (byte 2) registers. Use Write Enable command 0x06. 11 – send 8-bit data using command 0x01 to status register (byte 0). Use Write Enable command 0x06. 12 – send 24-bit data in 3 commands. Command 0x01 (byte 0) to status register 0, command 0x31 (byte 1) to status register 1, command 0x11 (byte 2) to status register 2. 20 - send 24-bit data in 3 commands. Command 0x01 (byte 0) to status register 0, command 0x31 (byte 1) to status register 1, command 0x11 (byte 2) to status register 2. Use Write Enable command 0x50 (Revision 1B only). 21 - send 24-bit data in 3 commands. Command 0x01 (byte 0) to status register 0, command 0x31 (byte 1) to status register 1, command 0x11 (byte 2) to status register 2. Use Write Enable command 0x50. Send Enable QPI Mode command 0x38 after modifying status (Revision 1B only)
0x61	1	setOptionsFxId	-	-	Specify device set options function. 13 – Enable quad mode setting bit 9 in device status register using the getStatus/setStatus functions. Send command 0x81 (Write 0xAB to volatile configuration register, 10 quad read dummy cycles).

					14 – Enable quad mode setting bit 9 in device status register using the getStatus/setStatus functions. 15 – Enable quad mode setting bit 6 in device status register using the getStatus/setStatus functions. 16 – No functionality. 23 – Enable quad or dual mode depending on SPIFI header structure capabilities. If quad mode, send command 0x81 write configuration with data = 0x6B (6 quad read dummy cycles), command 0x61 enhanced write configuration with data = 0x59. If dual mode, command 0x81 write configuration with data= 0x4B (4 dual read dummy cycles), command 0x61 enhanced write configuration with data = 0x99.
0x62	1	getReadCmdFxId	-	-	Device read function. Specifies either quad mode or dual mode read depending on SPIFI header structure capabilities (offset 0x40). Read dummy cycles must be programmed in SPIFI header structure capabilities, bits 27-24 for quad read dummy cycles, or bits 31-28 for dual read dummy cycles. 17 – Use 3-byte addressing commands with quad Read (0xEB) and dual read (0xBB). Both commands serial opcode, quad/dual address, data (1-4-4). 18 – Use 3-byte addressing commands with quad Read (0xEB) and dual read (0xBB). Both commands quad/dual opcode, address, data (4-4-4). 19 – Use 4-byte addressing commands with quad Read (0xEC) and dual read (0xBC). Both commands serial opcode serial, quad/dual address, data (1-4-4). 24 – Use 4-byte addressing commands with quad Read (0xEC) and dual read (0xBC). Both commands quad/dual opcode, address, data (4-4-4).
0x63	1	-	-	-	Reserved. Set to 0.

The image header is present in the startup files in LPC540xx SDK - *startup_LPC54018.s* in Keil MDK, IAR EWARM and *startup_lpc54018.c* in MCUXpresso IDE.

Note: The quad SPI descriptor in image header is different in LPC540xx Rev 0A devices and Rev 1B devices.

Following is the code snippet for Keil and IAR IDE.

```
__vector_table_0x1c
                DCD     0                   ; (0x1c)Checksum of the first 7 words
                DCD     0xFFFFFFFF          ; (0x20)ECRP
                DCD     0xEDDC94BD          ; (0x24)Enhanced image marker,set to
                                          0x0 for legacy boot
                DCD     0x160               ; (0x28) Pointer to image header
                                         marker
                DCD     SVC_Handler
                DCD     DebugMon_Handler
                DCD     0
                DCD     PendSV_Handler
                ...
                ...
```

Following is the code snippet for MCUXpresso IDE.

```
__valid_user_code_checksum,        // LPC MCU checksum
0,                                 // ECRP
(void *)0xEDDC94BD,                // (0x24)Enhanced image marker,
(void *)0x160,                     // (0x28) Pointer to image header marker
SVC_Handler,                       // SVCall handler
DebugMon_Handler,                  // Debug monitor handler
0,                                 // Reserved
PendSV_Handler,                    // The PendSV handler
SysTick_Handler,                   // The SysTick handler
```

The bootloader scans for the user images by examining the enhanced image marker located at 0x0000 0024 and it should match with the value 0xEDDC 94BD to begin header validation. Image header offset at 0x000 00028 points to the image header structure. In the above code, the image header structure is located at address 0x0000 0160.

Following is the code snippet for Keil MDK IDE.

```
__image_header
                DCD   0xFEEDA5A5 ; (0x00, 0x160) Header Marker
                IF      :DEF: XIP_IMAGE
                    IF      :DEF: ADD_CRC
                        DCD   2         ; (0x04) Image Type
                    ELSE
                        DCD   3         ; (0x04) Image Type
                    ENDIF

                ELSE

                    IF      :DEF: ADD_CRC
                        DCD   0             ; (0x04) Image Type
                    ELSE
                        DCD   1             ; (0x04) Image Type
                ENDIF

                ENDIF
                IF      :DEF: XIP_IMAGE
                        DCD   0x10000000          ; (0x08) Load_address
                ELSE
                        DCD   0x00000000          ; (0x08) Load_address
                ENDIF
```

Following is code snippet for IAR Embedded Workbench IDE.

```
__image_header
        DCD   0xFEEDA5A5 ; (0x00, 0x160) Header Marker
        #ifdef XIP_IMAGE
          #ifdef ADD_CRC
                DCD   2           ; (0x04) Image Type
          #else
                DCD   3           ; (0x04) Image Type
          #endif
        #else
          #ifdef ADD_CRC
                DCD   0           ; (0x04) Image Type
          #else
                DCD   1           ; (0x04) Image Type
          #endif
        #endif

        #ifdef XIP_IMAGE
                DCD   0x10000000  ; (0x08) Load_address
        #else
                DCD   0x00000000  ; (0x08) Load_address
        #endif
```

Following is the code snippets for MCUXpresso IDE.

```
(void *)0xFEEDA5A5,         // Header Marker
#if defined (ADD_CRC)
    (__imghdr_imagetype - 1),   // (0x04) Image Type
     __imghdr_loadaddress,      // (0x08) Load_address
#else
    __imghdr_imagetype,         // (0x04) Image Type
    __imghdr_loadaddress,       // (0x08) Load_address
#endif
```

The first entry in the image header structure at address 0x160 should be 0xFEED A5A5. It is followed by the *image_type* field. The *image_type* field indicates if an image is XIP image or not and whether CRC computation is required for the image. See Table 2 for the image type fields.

Table 2. Image type fields

Image type	Description
0x0	Load Image into internal SRAM (SRAMX or SRAM0-3) based on *load_address* field in image header. CRC computation is performed on the image.
0x1	Load Image into internal SRAM (SRAMX or SRAM0-3) based on *load_address* field in image header. CRC computation is not performed.
0x2	XIP image. CRC computation is performed.
0x3	XIP image. CRC computation is not performed.

If the image is stored externally, the boot ROM copies the first 512 bytes from the image (vector components and image header) into internal SRAMX at location 0x0000 0000 to validate the vector table and image header. If the image is downloaded from a serial interface (via USART, I2C, SPI, USB), the complete image is loaded into SRAMX before the image is validated.

For non-XIP images, the *load_address* field determines in the location in the SRAM region (SRAMX or SRAM0-3) where the image should be loaded. Set *load_address* value to 0x0000 0000 to load and execute image in SRAMX. Set *load_address* value to 0x2000 0000 to load and execute image in SRAM0-3. In this case, the application image is copied from SRAMX to SRAM0-3 after validation is completed.

For XIP images, *load_address* specifies the execution address of the image. Set *load_address* to 0x1000 0000 to execute from SPIFI and set *load_address* to 0x8000 0000 to execute from parallel flash (EMC).

Following is the code snippet from Keil MDK IDE.

```
DCD   |Load$$LR$$LR_m_text$$Length| + |Image$$VECTOR_ROM$$Length| - 4 ; (0x0C)
load_length, exclude 4 bytes CRC field.
DCD   0   ; (0x10) CRC value (only applicable to NON Non-secure images).
DCD   0   ; (0x14) Version (only applicable to DUAL_ENH image type.
DCD   0   ;(0x18) EMC static memory configuration settings, required for EMC boot
```

Following is code snippet for IAR Embedded Workbench IDE.

```
DCD   sfe(RO) - __vector_table - 4   ; (0x0C) load_length, image size is RO end -
__vector_table - CRC field
DCD   0    ; (0x10) CRC value (only applicable to NON Non-secure images).
DCD   0    ; (0x14) Version (only applicable to DUAL_ENH image type.
DCD   0    ; (0x18) EMC static memory configuration settings, required for EMC boot
```

Following is the code snippet from MCUXpresso IDE.

```
      _image_size - 4, // (0x0C) load_length, exclude 4 bytes CRC field.
      0,              // (0x10) CRC value (only applicable to NON Non-secure images).
      0,             // (0x14) Version (only applicable to DUAL_ENH image type.
      0,    // (0x18) EMC static memory configuration settings, required for EMC boot
```

The *image_length* field at address 0x16C contains the total length of the image - 4. This length does not include the 4 bytes that make up the CRC value field. The *CRC* field at address 0x170 contains the CRC32 value of the image. This field should be excluded during the CRC computation of the image. *Image_version* field contains the version number of the image. The *emc_timings* field contains the EMC static memory configuration settings required for EMC boot. If parallel NOR flash is not used, set the field to 0x0. LPC540xx SDK examples use external quad SPI device as boot device. Therefore, this field is set to 0x0.

Following is the code snippet from Keil MDK IDE.

```
IF      :DEF: IMG_BAUDRATE
DCD   IMG_BAUDRATE;(0x1C) image baudrate
ELSE
DCD   0          ; (0x1C) reserved
ENDIF
DCD   0          ; (0x20) reserved
DCD   0xEDDC94BD ; (0x24) Image_marker
DCD   0          ; (0x28) reserved
DCD   0          ; (0x2C) reserved
```

Following is code snippet for IAR Embedded Workbench.

```
DCD   IMG_BAUDRATE; (0x1C) image_baudrate
DCD   0          ; (0x20) reserved
DCD   0xEDDC94BD ; (0x24) Image_marker
DCD   0          ; (0x28) reserved
DCD   0          ; (0x2C) reserved
```

Following is the code snippet from MCUXpresso IDE.

```
(void *)IMG_BAUDRATE,        // (0x1C) image baudrate
 0,                           // (0x20) reserved
 (void *)0xEDDC94BD,          // (0x24) Image_marker
 0,                           // (0x28) reserved
 0,                           // (0x2C) reserved
```

The address 0x17C is the *image baudrate* field that indicates the SPI clock if a SPI or SPIFI flash device is used. If this value is set 0x0, the serial clock defaults to 24 MHz for SPIFI and 12 MHz for SPI. See below table for the SPI/SPIFI clock frequencies.

Table 3. SPI/SPIFI clock speed

Header SPI/SPIFI clock frequency value (MHz)	SPI/SPIFI clock speed (MHz)
< 24 (SPI only)	12 (SPI only)
< 32	24
< 48	32
< 96	48
>=96	96

The SPI/SPIFI device descriptor is located at an offset of 0x30 from the header marker. The descriptor parameters should be defined based on the external SPI flash device. See Fig 1 for SPIFI device descriptor for by W25Q128JV QSPI device from Windbond. The manufacturer ID for Winbond Serial Flash is 0xEF. The device ID for 0x9F command is 7018h. Extended count and extended data are 0.

3.1 Device ID and instruction set tables

3.1.1 Manufacturer and device identification

Manufacturer ID	**MF7 to MF0**	
Windbond serial flash	EFh	
Device ID	ID7 to ID0	ID15 to ID0
Instruction	ABh,90h,92h,94h)	9Fh
W25Q128JV-IQ	17h	4018h
W25Q128JV-IM*	17h	7018h

Fig 1. Manufacturer ID and device ID from Winbond W25Q128 datasheet

The capabilities supported on this device are dual read and quad read.

See Fig 2 for the read and write status register commands.

Data	**Byte 1**	**Byte 2**
Write enable	06h	-
Volatile SR write enable	50h	-
Read status register 1	05h	(S7- S0)(2)
Write status register 1(4)	01h	(S7-S0)(4)
Read status register 2	35h	(S15-S8)(2)
Write status register 2	31h	(S15-S8)
Read status register 3	15h	(S23-S16)(2)
Write status register 3	11h	(S23-S16)

Fig 2. Read/Write status register commands from Winbond W25Q128 datasheet

The value of *clearStatusFxId* is 0x03.
To read the status registers, the commands 0x05 and 0x35 are issued. Hence, *getStatusFxId* value is 0x04.
To write the status registers, the command 0x01 is issued with write enable command 0x06. Hence, *setStatusFxId* value is 0x09.
The quad mode enable bit (bit 9) in status register can be set using *setStatusFxId / getStatusFxId* functions and send command 0x81. Hence, *setOptionsFxId* value is 0x0D.
3-byte addressing commands is used with quad Read (0xEB) and dual read (0xBB). Both commands serial opcode, quad/dual address, data (1-4-4). Hence, *getReadCmdFxId* value is 0x11.

Note: The SPIFI descriptor below (and in SDK) can be used as a common descriptor for both Rev 0A and Rev 1B devices.

Following is the code snippet for Keil MDK IDE.

```
IF      :DEF: W25Q128JVFM
; SPIFI Descriptor - W25Q128JVFM
DCD   0x00000000 ;0xFFFFFFFF to default 1-bit SPI mode ;DevStrAdr
DCD   0x001870EF ;mfgId + extCount
DCD   0x00000000 ;extid 0-3
DCD   0x00000000 ;reserved
DCD   0x1301001D ;caps
DCD   0x00000100 ;Blks
DCD   0x00010000 ;blkSize
DCD   0x00000000 ;reserved
DCD   0x00000100 ;pageSize
DCD   0x00003F00 ;maxReadSize
DCD   0x68506850 ;maxClkRate,maxReadRate,maxHSReadRate,maxProgramRate
DCD   0x04030050 ;maxHSProgramRate,initDeInitFxId,clearStatusFxId,getStatusFxId
DCD   0x14110D09 ;setStatusFxId,setOptionsFxId,getReadCmdFxId,getWriteCmdFxId
ENDIF
```

Following is code snippet for IAR Embedded Workbench.

```
#ifdef W25Q128JVFM
; SPI Descriptor - W25Q128JVFM
DCD   0x00000000 ;0xFFFFFFFF to default 1-bit SPI mode  ;DevStrAdr
DCD   0x001870EF ;mfgId + extCount
DCD   0x00000000 ;extid 0-3
DCD   0x00000000 ;reserved
DCD   0x1301001D ;caps
DCD   0x00000100 ;Blks
DCD   0x00010000 ;blkSize
DCD   0x00000000 ;reserved
DCD   0x00000100 ;pageSize
DCD   0x00003F00 ;maxReadSize
DCD   0x68506850 ;maxClkRate,maxReadRate,maxHSReadRate,maxProgramRate
DCD   0x04030050 ;maxHSProgramRate,initDeInitFxId,clearStatusFxId,getStatusFxId
DCD   0x14110D09 ;setStatusFxId,setOptionsFxId,getReadCmdFxId,getWriteCmdFxId
#endif
```

Following is the code snippet from MCUXpresso IDE.

```
#ifdef W25Q128JVFM
/* SPIFI Descriptor - W25Q128JVFM */
(void *)0x00000000,         // 0xFFFFFFFF to default 1-bit SPI mode  ;DevStrAdr
(void *)0x001870EF,         // mfgId + extCount
(void *)0x00000000,         // extid 0-3
(void *)0x00000000,         // reserved
(void *)0x1301001D,         // caps
(void *)0x00000100,         // Blks
(void *)0x00010000,         // blkSize
(void *)0x00000000,         // reserved
(void *)0x00000100,         // pageSize
(void *)0x00003F00,         // maxReadSize
(void *)0x68506850,   // maxClkRate,maxReadRate,maxHSReadRate,maxProgramRate
(void *)0x04030050,// maxHSProgramRate,initDeInitFxId,clearStatusFxId,getStatusFxId,
(void *)0x14110D09, // setStatusFxId,setOptionsFxId,getReadCmdFxId,getWriteCmdFxId
#endif
```

4. Conclusion

LPC540xx devices require an image header in the application image. The boot ROM validates the image header before executing user application. This application note gives an overview of the various parameters in the image header structure.

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.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

5.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

5.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

5.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP B.V.

6. List of figures

Fig 1. Manufacturer ID and device ID from Winbond W25Q128 datasheet 12

Fig 2. Read/Write status register commands from Winbond W25Q128 datasheet 12

7. List of tables

Table 1. Image header for LPC540xx devices 4
Table 2. Image type fields..10
Table 3. SPI/SPIFI clock speed 11

8. Contents

1. Introduction 3
2. LPC540xx boot flow summary 3
3. Image header format 3
3.1 Device ID and instruction set tables 12
3.1.1 Manufacturer and device identification 12
5. Legal information 15
5.1 Definitions 15
5.2 Disclaimers...15
5.3 Licenses...15
5.4 Patents...15
5.5 Trademarks..15
6. List of figures..16
7. List of tables 17
8. Contents..18

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

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

Date of release: 27 August 2018
Document identifier: AN12122