AN12125

Using LPC802 as I²C-bus EEPROM

Rev.1.0 20 February 2018 **Application note**

Document information

Info	Content
Keywords	Flash, EEPROM, IAP, I²C-bus, LPC802
Abstract	This application note introduces how to use LPC802 as EEPROM via I²C-bus interface.

Revision history

Rev	Date	Description
1.0	20180220	Initial version

Contact information

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

1. Overview

LPC802 is a new member of the LPC800 series, which satisfies the demand for improved power efficiency. It is an ideal product for migration of market from 8-bit architecture with a very low cost. LPC802 has up to 15 MHz Cortex-M0 + core with 16 kB flash and 2 kB RAM. It is suitable for I/O expander and small programmable logic unit. This application note discusses about LPC802 as an EEPROM like device. As LPC802 internal flash can endure at least 200k R/W cycles, it is ideal to use internal flash of LPC802 as non-volatile memory for EEPROM. With a dedicated firmware, LPC802 can transform into smart EEPROM with selectable interface such as, UART or SPI.

This application note uses the I^2C-bus interface of LPC802 and on-chip flash memory to simulate traditional I^2C-bus EEPROM devices. It has the following information:

- General description, memory resources and layout, available peripherals
- Usage of flash IAP with example code
- Usage of I^2C-bus module, especially for I^2C-bus slave mode

Note: A basic knowledge of the I^2C-bus is required. For the I^2C-bus specification, see

http://www.i2c-bus.org/specification/

2. Hardware

2.1 MCU overview

The LPC802 are ARM® Cortex®-M0+ based, low-cost 32-bit MCU family operating at CPU frequencies up to 15 MHz. The LPC802 supports 16 kB flash memory and 2 kB SRAM, offering TSSOP16, TSSOP20, HVQFN33 and WLCSP16 package. In addition, the dual power supply parts provide level shifter function, which reduces the corresponding external components and the total system BOM cost.
The peripherals of the LPC802 include:

- One I^2C-bus interface
- Up to two USARTs
- One SPI interface
- One multi-rate timer, self-wake-up timer, one general purpose 32-bit counter/timer
- One 12-bit ADC, one analog comparator
- Function-configurable I/O ports through a switch matrix, and up to 17 general-purpose I/O pins. Three I/O pins have high driver capability providing up to 20 mA source current

2.1.1 Flash features

The on-chip flash (total 16 kB) of LPC802 contains 16 sectors. The size of each sector is 1 kB and contains 16 pages. The size of each page is 64 bytes. The IAP command supports:

- Page erase
- Page writes
- Sector erase

See Table 1.for flash partition and configuration

Table 1. Flash partition and configuration

Virtual Sector number	Virtual Sector Size (KB)	Page number	Address Range	16KB flash	32KB flash
0	1	0 - 15	0 × 0000 0000 - 0 × 0000 03FF	yes	yes
1	1	16 - 31	0 × 0000 0400 - 0 × 0000 07FF	yes	yes
2	1	32 - 47	0 × 0000 0800 - 0 × 0000 0BFF	yes	yes
3	1	48 - 63	0 × 0000 0C00 - 0 × 0000 0FFF	yes	yes
4	1	64 - 79	0 × 0000 1000 - 0 × 0000 13FF	yes	yes
5	1	80 - 95	0 × 0000 1400 - 0 × 0000 17FF	yes	yes
6	1	96 - 111	0 × 0000 1800 - 0 × 0000 1BFF	yes	yes
7	1	112 - 127	0 × 0000 1C00 - 0 × 0000 1FFF	yes	yes
8	1	128 - 143	0 × 0000 2000 - 0 × 0000 23FF	yes	yes
9	1	144 - 159	0 × 0000 2400 - 0 × 0000 27FF	yes	yes
10	1	160 - 175	0 × 0000 2800 - 0 × 0000 2BFF	yes	yes
11	1	176 -191	0 × 0000 2C00 - 0 × 0000 2FFF	yes	yes
12	1	192 - 207	0 × 0000 3000 - 0 × 0000 33FF	yes	yes
13	1	208 - 223	0 × 0000 3400 - 0 × 0000 37FF	yes	yes
14	1	224 - 239	0 × 0000 3800 - 0 × 0000 3BFF	yes	yes
15	1	240 - 255	0 × 0000 3C00 - 0 × 0000 3FFF	yes	yes

Following is the IAP flash command executing time (tested on : MDK,V5.06 O3):

- Sector erase: 19.921 ms
- Page erase: 1.466 ms
- Page write: 1.526 ms

2.1.2 I²C-bus features

In this application note, LPC802 has one I²C-bus interface which is used in slave mode.

The I²C-bus interface has the following features:

- Independent master, slave, and monitor functions
- Supports both multi-master and multi-master with slave functions
- Multiple I²C-bus slave addresses supported in hardware
- One slave address can be selectively qualified with a bit mask or an address range in

In this application, I²C0 is configured in slave mode and used as a communication interface for EEPROM device.

2.2 Hardware connection

The hardware connection is simple, with only two pins used.

- I²C_SDA: P0_10
- I²C_SCL: P0_16

The I²C-bus requires two 4.7 kΩ pull-up resistors for SDA and SCL lines.

3. Software

3.1 IAP (In Application Programing)

This section gives an example of how to use the flash IAP function of LPC802.

1. Call the IAP routine with a word pointer in register r0, pointing to memory (RAM) containing command code and parameters.
2. Define a IAP_Call function pointer in the address 0x0F001FF1.
3. Use IAP_Call to setup IAP function and organize those function calls into higher level APIs.

For detailed information about IAP command, see Chapter 4: LPC802 ISP and IAP in the LPC802 user manual.

Note that when calling IAPs the flash might be disabled temporarily. The simplest work-around is to disable interrupts when executing flash IAP calls.

Declare IAP calls:

```
1          /* IAP Call */
2          struct sIAP
3          {                  // IAP Structure
4              unsigned long cmd;           // Command
5              unsigned long par[4];        // Parameters
6              unsigned long stat;          // Status
7              unsigned long res[2];        // Result
8          } IAP;
9          typedef void (*IAP_Entry) (unsigned long *cmd, unsigned long *stat);
10         #define IAP_Call ((IAP_Entry) 0x0F001FF1)
11         #define PAGE_SIZE   (64)
12         #define CCLK        (15000)
```

Erase page function:

```
13         uint8_t FLASH_ErasePage(uint32_t addr)
14         {
15             unsigned long n;
16             unsigned long page;
17
18             n = GetSecNum(addr);                          // Get Sector Number
19
20             IAP.cmd    = 50;                             // Prepare Sector for
     Erase
21             IAP.par[0] = n;                              // Start Sector
22             IAP.par[1] = n;                              // End Sector
23             __disable_irq();
24             IAP_Call (&IAP.cmd, &IAP.stat);              // Call IAP Command
25             __enable_irq();
26             if (IAP.stat) return (1);                    // Command Failed
27
28             page = addr/PAGE_SIZE;
```

```
29
30            IAP.cmd    = 59;
31            IAP.par[0] = page;
32            IAP.par[1] = page;
33            IAP.par[2] = CCLK;
34            __disable_irq();
35            IAP_Call (&IAP.cmd, &IAP.stat);
36            __enable_irq();
37            if (IAP.stat) return (1);
38
39            return (0);
40        }
41
```

Write Page function:

```
42        uint8_t FLASH_WritePage(uint32_t addr, const uint8_t *buf)
43        {
44            unsigned long n;
45
46            n = GetSecNum(addr);                          // Get Sector Number
47            IAP.cmd    = 50;                             // Prepare Sector for
      Write
48            IAP.par[0] = n;                              // Start Sector
49            IAP.par[1] = n;                              // End Sector
50            __disable_irq();
51            IAP_Call (&IAP.cmd, &IAP.stat);              // Call IAP Command
52            __enable_irq();
53            if (IAP.stat) return (1);                    // Command Failed
54
55            IAP.cmd    = 51;                             // Copy RAM to Flash
56            IAP.par[0] = addr;                           // Destination Flash
      Address
57            IAP.par[1] = (unsigned long)buf;             // Source RAM Address
58            IAP.par[2] = PAGE_SIZE;                      // Fixed Page Size
59            IAP.par[3] = CCLK;                           // CCLK in kHz
60            __disable_irq();
61            IAP_Call (&IAP.cmd, &IAP.stat);              // Call IAP Command
62            __enable_irq();
63            if (IAP.stat) return (1);                    // Command Failed
64
65            return 0;
66        }
```

3.2 I²C-bus interface programming

In this application, I²C-bus is configured as slave function. To use I²C-bus module, correct initialization steps must be performed before using it. It includes clock gate control, clock routing, pin MUX, etc. The following code snippet shows initializing the I²C-bus and enabling the I²C-bus interrupt:

```
67          void app_i2c_slave_init(uint8_t slv_addr)
68          {
69              /* pin mux */
70              ConfigSWM(I2C0_SDA, P0_10);
71              ConfigSWM(I2C0_SCL, P0_16);
72
73              /* using main clock */
74              LPC_SYSCON->I2C0CLKSEL = 1;
75
76              /* give I2C a reset */
77              LPC_SYSCON->PRESETCTRL[0] &= (I2C0_RST_N);
78              LPC_SYSCON->PRESETCTRL[0] |= ~(I2C0_RST_N);
79
80              LPC_I2C0->DIV = 2;
81              LPC_I2C0->CFG = CFG_MSTENA | CFG_SLVENA;
82
83              LPC_I2C0->SLVADR0 = (slv_addr << 1) | 0;
84
85              // Enable the I2C0 slave pending interrupt
86              LPC_I2C0->INTENSET = STAT_SLVPEND | STAT_SLVDESEL;
87              NVIC_EnableIRQ(I2C0_IRQn);
88          }
```

I²C-bus slave operation is done by software interrupt handling. Two major I²C-bus interrupt sources are used:

- SLVPENDING: Indicates that the slave function is waiting to continue communication on the I²C-bus and needs software service
- SLVDESEL: A stop condition occurred or the new address on bus does not match the current slave address

In I²C-bus interrupt, the software should check the interrupt state register. When a STOP condition occurs, SLVDESEL is generated. Software posts a message to main thread to process more operations such as, writing the received data into NVM. When SLVPENDING is generated, software should check the state code in SLVSTATE filed in I²C0->STAT register to determine the next operation.

- SLVSTATE = 0x00 (slave address received and matched)

Software should record address by reading SLVDATA and interpret whether it is a reading or a writing operation

- SLVSTATE = 0x01 (slave received a new byte)

Software should read SLVDATA to get transferred data and store in RAM. Then set CTL_SLVCONTINUE bit in SLVCTRL register, to let I²C-bus hardware continue processing bus transition

- SLVSTATE = 0x02 (slave need transmit a new byte to master)

Software should feed data into SLVDATA, then set CTL_SLVCONTINUE bit in SLVCTRL register, to let I²C-bus hardware continue processing bus transition

3.3 Communication protocol

R/W timing is similar to other EEPROM devices in the market, but only supports sequential R/W.

Following shows the detailed communication protocol:

- Sequential write

  For write operation, two bytes of address are required after the slave write address is sent out. These two bytes select one out of the 65535 bytes of locations in the memory, with lower byte transferred first.

  The format is:

  CHIP_ADDR (7 bits + W (1)) + DATA_ADDR_L + DATA_ADDR_H + DATA (0) +… + DATA(N)

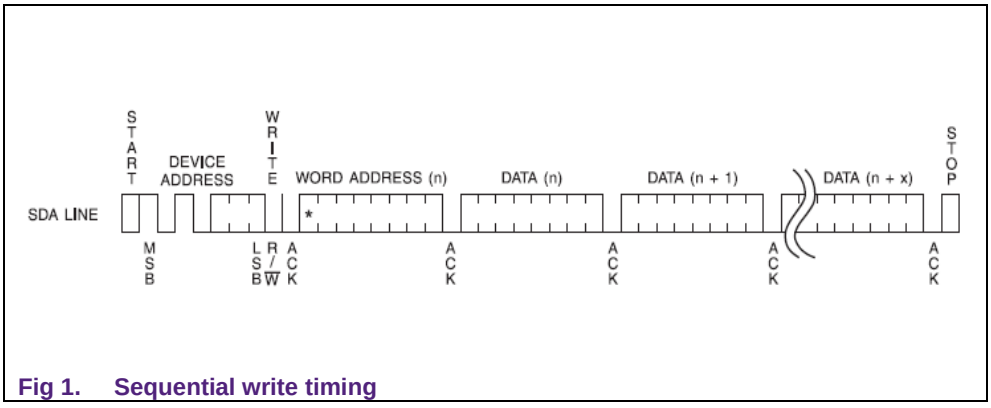

Fig 1. Sequential write timing

- Sequential read

  Sequential read operation is similar to the write operation, but, requires an I²C-bus restart signal with slave read address. The master now responds with an acknowledge, indicating that it requires additional data. The device continues to output data for each acknowledge received.

  The format is:

  CHIP_ADDR (7 bits + W (0)) +DATA_ADDR_L + DATA_ADDR_H + CHIP_ADDR (7 bits + R (1)) + DATA(0) +… +DATA(N)

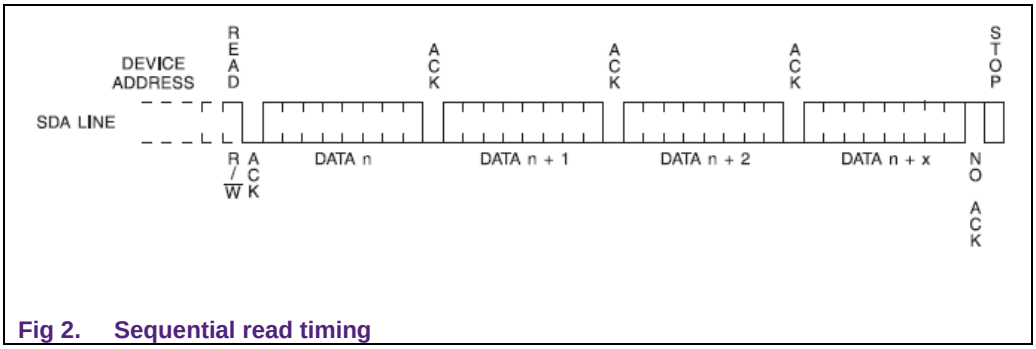

Fig 2. Sequential read timing

For sequential write, on receiving each data byte, the two-byte address is internally incremented. When the address reaches the page boundary, the following byte will be discarded. So, the maximum data length of write operation is page length (64 byte).

3.4 Summary

The demo uses flash address from 0X1000 to 0X4000 on LPC802 as EEPROM memory. The first 4 kB of flash is reserved to firmware itself. So, a total of 12 kB flash can be used as EEPROM memory.

For slave address, 0X50 is chosen for compatibility with current EEPROMs on markets.

For software workflow, a standard foreground and background system is used. Only I²C-bus interrupt is enabled. Software waits for I²C-bus transition and once I²C-bus interrupt is generated, software handles the top-half process and pushes message to main thread for bottom-half task. The main thread handles the task such as writing data into flash. Figure 3 shows software workflow:

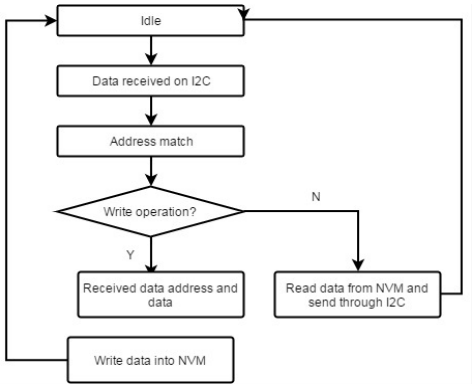

Fig 3. Diagram for software workflow

4. Test and result

4.1 Environment setup

In this section a test environment is built using LPC845 as master to read/write LPC802 through predefined I²C-bus interface. See Section 3.2.

Figure 4 shows the test environment block diagram.

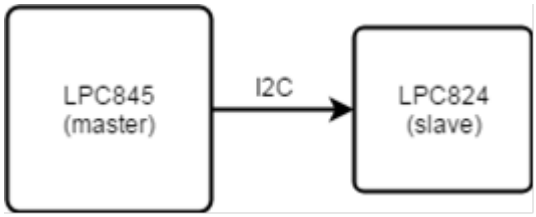

Fig 4. Test environment block diagram

4.2 Hardware and connection:

- Master: LPC845 XpressoMAX board
  - I^2C _SDA: P0_11
  - I^2C _SCL: P0_10
- Slave: LPC802 demo board
  - I^2C _SDA: P0_10
  - I^2C _SCL: P0_16

Figure 5 shows the hardware connection.

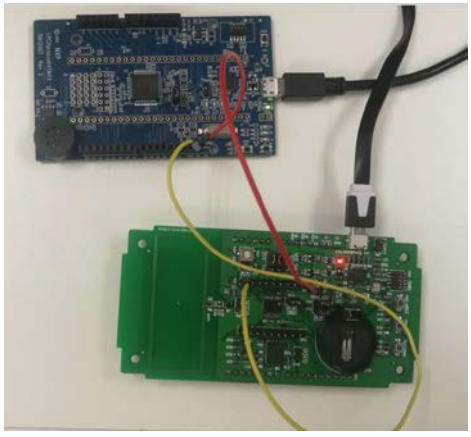

Fig 5. Hardware connection

4.3 Test steps

- Prepare and connect hardware as described in the Section 4.2
- Download firmware to each board
  - For master demo project (LPC845): compile project under "lpc845_EEPROM_master.zip" and download image into LPC845 board, or download the pre-compiled image under "/binary/ lpc845_EEPROM_master_demo.bin" into LPC845

- For slave firmware project (LPC802), compile project under “lpc845_EEPROM_master” and download image into LPC845 board or download the pre-compiled image under “/binary/ lpc802_EEPROM_frimware.bin” into LPC802

- Open serial terminal on LPC845 Xpresso board, set baud rate to 115200-N-8-N-1. See Figure 6 for the output.

```
I2C master test for lpc802 eeporm demo CoreClock:30000000Hz
chip test begin....
chip test finish....
write page(64byte) time:6056 us
read page(64byte) time:2499 us
```

Fig 6. Test log

4.4 Test results

Test using external MCU to send test sequential R/W command. Note that ONLY sequential R/W command is supported on current firmware. See Section 3.3 “Communication protocol”

Figure 7 and Figure 8 shows a typical read and write timing:

Four bytes sequential read operation timing:

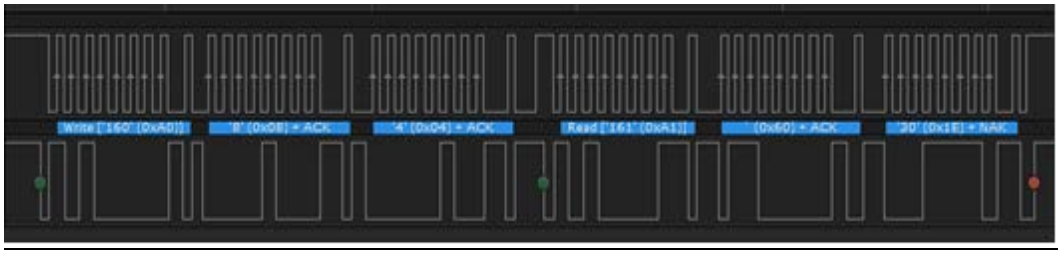

Fig 7. Sequential read timing

Two bytes sequential write operation timing:

Fig 8. Sequential write timing

4.5 Performance

The test conditions are:

- Test platform: LPC845 Xpresso board
- LPC845 core clock: 30 MHz
- I²C-bus clock: 373 kHz
- LPC802 core clock: 15 MHz

The firmware project and optimization setting of host test project does not take much effect on the performance. It is because, most of the time of R/W is spent on I²C-bus transfer and flash erase/write operation.

- Write one page (64 byte)
  - Time 6.074 ms
    - This time is the sum of I²C-bus transfer time plus internal flash page erase time of LPC802 and page write time. For flash erase and write time, refer Section 2.1.1.
    - The I²C-bus transfer time depends on I²C-bus clock speed and transfer size. In this case, transfer of one-page requires 64 (data) + 1 (chip address) + 2 (data address) = 67 bytes.
- Read one page (64 byte)
  - Time: 2.504 ms
    - This value is almost equal to I²C-bus transfer time because LPC802 takes few µs to fetch data from its internal flash and write it to I²C-bus.

5. Conclusion

This application note mainly discusses the following topics:

- LPC802 flash features: Includes flash partition, IAP usage and example code
- LPC802 I²C-bus features, slave mode, knowledge about how to write software to co-work with I²C-bus module and how to handle I²C-bus transition
- A demo software using LPC802 as an I²C-bus EEPROM to demonstrate the above two features
- A test demo based on LPC845 Xpresso board that acts as I²C-bus master to R/W NVM of LPC802, also providing sources and project file.

It is valuable to upgrade this demo to make LPC802 a smarter EEPROM- like device, such as adding UART/SPI interface, handling page boundary issue internally, and adding I/O expander function.

6. Legal information

6.1 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.

6.2 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

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 and its suppliers accept 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.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

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.

6.3 Licenses

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

6.4 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>

6.5 Trademarks

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

<Name> — is a trademark of NXP Semiconductors N.V.

7. List of figures

8. List of tables

9. Contents

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

Date of release: 20 February 2018
Document identifier: AN12125