AN12127

LPC5411x Flexcomm Interface with Eight USARTS

Rev.1.0 — 7 March 2018 Application note

Document information

Info	Content
Keywords	LPC5411x, Flexcomm, UART, LPCXpresso5411x LQFP64 Rev A board, Keil MDK, IAR embedded workbench, MCUXpresso
Abstract	This application note introduces Flexcomm features. It also shows implementation of eight UARTs with LPC5411x Flexcomm Interface, with an example of supporting IAR, Keil and the MCUXpresso IDEs. It is easy to implement eight UARTs with Flexcomm and MCUXpresso tool. One UART is connected to the PC to show the demo on the console, while the other seven UARTs are demonstrated in a loopback mode.

Revision history

Rev	Date	Description
1.0	20180307	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. Overview

The LPC5411x series are ARM® Cortex®-M4 based microcontrollers for embedded applications. These devices include an optional ARM Cortex-M0+ co-processor, up to 192 kB of on-chip SRAM, and up to 256 kB on-chip flash, full speed USB device interface, a DMIC subsystem with dual-channel PDM microphone interface and I²S-bus. It also includes five general-purpose timers, one versatile timer with PWM and many other capabilities (SCTimer/PWM), one RTC/alarm timer, one 24-bit multi-rate timer (MRT), a Windowed WatchDog Timer (WWDT), eight flexible serial communication peripherals Flexcomm (each of which can be a USART, SPIs, or I²C-bus interface), and one 12-bit 5.0 Msamples/sec ADC, and a temperature sensor.

Flexcomm Interface is one of the characteristics of numerous peripherals in the LPC serials and is easy to use.

Key features of Flexcomm Interface are:

- 4-in-1 serial communication interface: USART, I²C-bus, SPI, two have I²S-bus
  - Select one at a given time.
- Eight instances of Flexcomm Interface available
  - Maximum eight USARTs, eight SPIs, eight I²C-buses, and two I²S-buses.
- Built-in FIFO
  - FIFO has some dedicated features for different configurations.
  - I²C-bus does not use FIFO.
- DMA support
  - Using DMA *peripheral request* input, each DMA channel maps to one FC instance.
- Hardware batching with DMA when CPU is OFF.

Each Flexcomm Interface provides a choice for the serial peripheral functions discussed. One of them should be chosen by the user, before the function can be configured and used. See Fig 1 for the peripheral functions with the Flexcomm channels.

Flexcomm	Peripheral Function
Flexcomm 0 to Flexcomm 7	USART with asynchronous operation or synchronous master or slave operation.
	SPI master or slave, up to 4 SSEL per channel
	I²C with separate master, slave and monitor functions.
Flexcomm 6 and Flexcomm 7	I²S function with one to four I²S channel pairs.

Fig 1. Flexcomm channels and peripheral functions

In some applications, multiple serial ports may be required. For example, the IOT node should be connected to devices like WIFI, LORA, ZIGBEE, Bluetooth, sensor, and LCD, with serial ports.
This application note introduces how to implement eight UARTs with Flexcomm Interface. For demonstration, one UART is connected to PC for display on console and other seven UARTs are demonstrated with a loopback mode.

2. Flexcomm Interface and USART

2.1 Block diagram

See Fig 2 for the overall structure of one Flexcomm Interface.

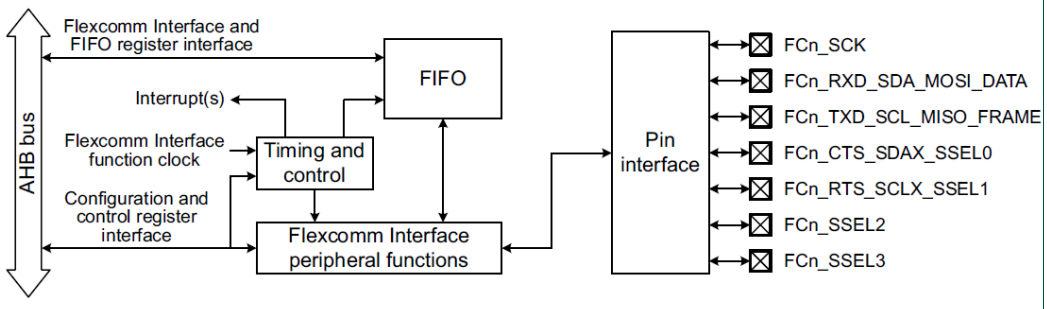

Fig 2. Flexcomm interface block diagram

See Fig 3 for the overall structure of the USART interface.

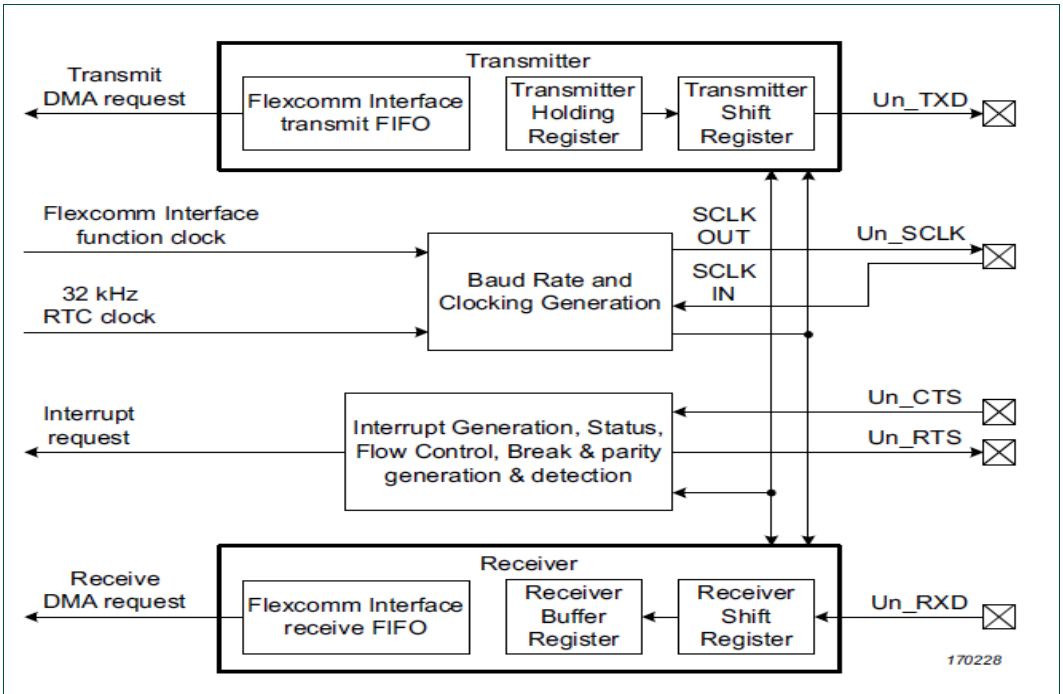

Fig 3. USART block diagram

2.2 Pin description

Each Flexcomm Interface allows up to seven pin connections. If the Flexcomm Interface does not use all of these pins, the unused pins can be configured as other function pins via IOCON. See Fig 4 for the Flexcomm Interface pin description.

Pin	Type	Description
SCK	I/O	Clock input or output for the USART function in synchronous modes.
	I/O	Clock input or output for the SPI function.
	I/O	Clock input or output for the I^2S function (if present).
RXD_SDA_MOSI or RXD_SDA_MOSI_DATA	Input	Receive data input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Master data output/slave data input for the SPI function.
	I/O	Data input or output for the I^2S function (if present).
TXD_SCL_MISO or TXD_SCL_MISO_WS	Output	Transmit data output for the USART function.
	I/O	SCL input/output for the I^2C function.
	I/O	Master data input/slave data output for the SPI function.
	I/O	WS (also known as LRCLK) input or output for the I^2S function (if present).
CTS_SDA_SSEL0	Input	Clear To Send input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Slave Select 0 input or output for the SPI function.
RTS_SCL_SSEL1	Output	Request To Send output for the USART function.
	I/O	SCL (clock) input/output for the I^2C function.
	I/O	Slave Select 1 input or output for the SPI function.
SSEL2	I/O	Slave Select 2 input or output for the SPI function.
SSEL3	I/O	Slave Select 3 input or output for the SPI function.

Fig 4. Flexcomm Interface pin description

The USART receive, transmit, and control signals are movable functions and are assigned to external pins via IOCON. See Fig 5 for the USART pin description.

Pin	Type	Name used in Pin Configuration chapter	Description
TXD	O	FCn_TXD_SCL_MISO_WS	Transmitter output for USART on Flexcomm Interface n. Serial transmit data.
RXD	I	FCn_RXD_SDA_MOSI_DATA	Receiver input for USART on Flexcomm Interface n. Serial receive data.
$\overline{RTS}$	O	FCn_RTS_SCL_SSEL1	Request To Send output for USART on Flexcomm Interface n. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.
$\overline{CTS}$	I	FCn_CTS_SDA_SSEL0	Clear To Send input for USART on Flexcomm Interface n. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low).
SCLK	I/O	FCn_SCK	Serial clock input/output for USART on Flexcomm Interface n in synchronous mode. Clock input or output in synchronous mode. **Remark:** When the USART is configured as a master, such that SCK is an output, it must actually be connected to a pin in order for the USART to work properly.

Fig 5. USART pin description

2.3 Function summary

LPC5411x devices include Flexcomm Interfaces and peripherals. See Fig 6. Specific part numbers and package variations may include a subset of this list.

Flexcomm number	Base address	USART	SPI	I^2C	I^2S
0	0x4008 6000	Yes	Yes	Yes	-
1	0x4008 7000	Yes	Yes	Yes, special I^2C pins available	-
2	0x4008 8000	Yes	Yes	Yes	-
3	0x4008 9000	Yes	Yes	Yes	-
4	0x4008 A000	Yes	Yes	Yes, special I^2C pins available	-
5	0x4009 6000	Yes	Yes	Yes	-
6	0x4009 7000	Yes	Yes	Yes	Yes, 1 channel pair
7	0x4009 8000	Yes	Yes	Yes	Yes, 1 channel pair

Fig 6. Flexcomm Interfaces and peripherals

A peripheral function can be selected by writing specific values in the field PERSEL in register PSELID. For example, to configure FC0 to USART, the value 0x1 should be written into PERSEL via software. Fig 7 shows the values for different functions.

To use the USART, configure the related registers of the USART peripheral.

Bit	Symbol	Value	Description	Reset Value
2:0	PERSEL		Peripheral Select. This field is writable by software.	0x0
		0x0	No peripheral selected.	
		0x1	USART function selected.	
		0x2	SPI function selected.	
		0x3	I^2C function selected.	
		0x4	I^2S transmit function selected.	
		0x5	I^2S receive function selected.	
		0x6	Reserved	
		0x7	Reserved	

Fig 7. Peripheral select by using the register

3. Implementation

This section introduces how to implement eight UARTs with LPC5411x Flexcomm Interface based on MCUXpresso SDK on LPCXpresso5411x board. To show the capacity of supporting up to eight UARTs, one UART is connected to PC to show the demo on the console (The H/W VCOM UART bridge is designed on LPCXpresso5411x board) and other seven UARTs are connected in a loopback mode. See Fig 8 for the system block diagram.

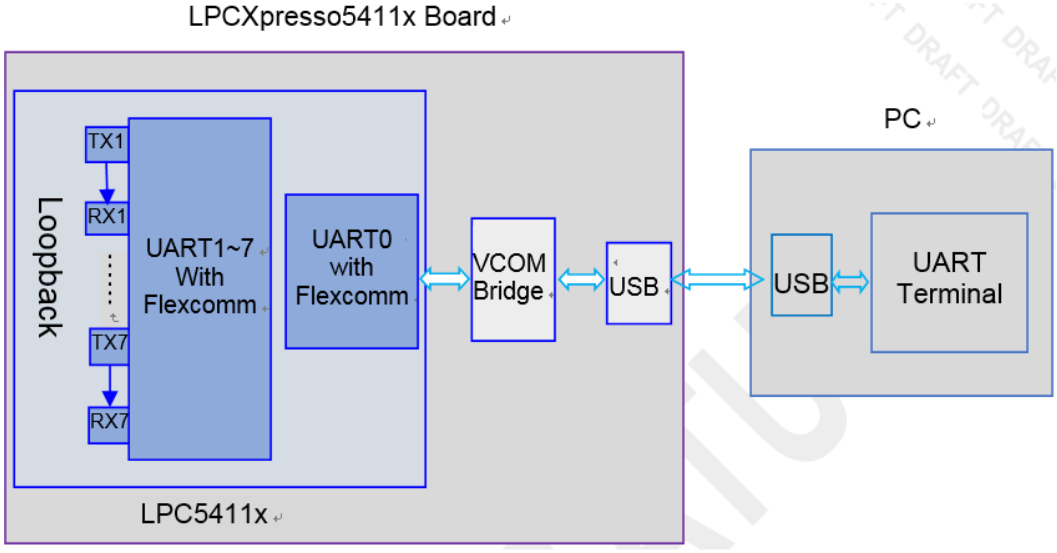

Fig 8. System block diagram

3.1 Hardware Design

Since the VCOM UART bridge is designed on the board (for more details, see the schematic of the board) and the loopback mode is integrated in the part, no extra circuit

is required to set up. The pins that should be selected for the eight UARTs is explained in the following section.

3.1.1 Pin configurations

See Fig 9 for recommended USART pin settings.

IOCON bit(s)	Type D pin	Type A pin	Type I pin
10	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1.
9	SLEW: Generally set to 0.	Not used, set to 0.	I2CDRIVE: Set to 0.
8	FILTEROFF: Generally set to 1.	Same as type D.	Same as type D.
7	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
6	INVERT: Set to 0.	Same as type D.	Same as type D.
5	Not used, set to 0.	Not used, set to 0.	I2CSLEW: Set to 1.
4:3	MODE: Set 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).	Same as type D.	Not used, set to 0.
2:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for USART input or output.	A reasonable choice for USART input or output.	Not recommended for USART functions that can be outputs in the chosen mode.

Fig 9. Recommended USART pin settings

It is recommended to use Type D and Type A pin.
Based on the target board LPCXpresso5411x, the specific RX and TX pins for eight UARTs are assigned. See Table 1.

Table 1. Pin assignment list

Pin number	Peripheral/Function	Pin signal
31	FLEXCOMM0/UART0	PIO0_0/FC0_RXD_SDA_MOSI
32	FLEXCOMM0/UART0	PIO0_1/FC0_TXD_SCL_MISO
11	FLEXCOMM1/UART1	PIO0_29/FC1_RXD_SDA_MOSI
12	FLEXCOMM1/UART1	PIO0_30/FC1_TXD_SCL_MISO
44	FLEXCOMM2/UART2	PIO0_9/FC2_TXD_SCL_MISO
57	FLEXCOMM2/UART2	PIO1_14/FC2_RXD_SDA_MOSI
47	FLEXCOMM3/UART3	PIO0_12/FC3_RXD_SDA_MOSI
48	FLEXCOMM3/UART3	PIO0_13/FC3_TXD_SCL_MISO
16	FLEXCOMM4/UART4	PIO1_2/FC4_RXD_SDA_MOSI
15	FLEXCOMM4/UART4	PIO1_1/FC4_TXD_SCL_MISO
51	FLEXCOMM5/UART5	PIO1_12/FC5_RXD_SDA_MOSI
54	FLEXCOMM5/UART5	PIO1_13/FC5_TXD_SCL_MISO
46	FLEXCOMM6/UART6	PIO0_11/FC6_RXD_SDA_MOSI_DATA
30	FLEXCOMM6/UART6	PIO1_10/FC6_TXD_SCL_MISO_WS
27	FLEXCOMM7/UART7	PIO1_7/FC7_RXD_SDA_MOSI_DATA
28	FLEXCOMM7/UART7	PIO1_8/FC7_TXD_SCL_MISO_WS

Note: The above pin assignment is one of the pin options which are dependent on target applications.

3.2 Software design

3.2.1 Overview

There are several examples of UART with different driver modes on SDK V2.2.1, for example polling and interrupt_transfer. The software implementation is based on the example of interrupt_transfer with interrupt and transfer modes.

Only the board and the application layers need to be developed for the chip driver. For board layer, the pins of eight UARTs should be configured using the MCUXpresso config tool, which is available in desktop and web versions. In this application, the desktop version V3.0 is used.

3.2.2 Introduction to config tools

The config tools (download link: https://mcuxpresso.nxp.com/en/welcome) which contains pins, clocks and project generator components that can be used to configure the pins. The *pins tool* is a GUI tool that can assign internal signals to external pins, set electrical properties, and generate ANSI-C source code. For more details, see the help contents of the tool.

Configure pins as mentioned in the *pins tool* and generate code: pin_mux.c and pin_mux.h. Replace the files already in the project folder with the new files.

For application layer, initialization and data transfer with loopback mode for eight UARTs with Flexcomm Interface should be implemented.

3.2.3 Example implementation

This section introduces how to initialize the eight UARTs with Flexcomm Interface and transfer data via the UARTs at the application level.

1. Initializations
    - Attach clock to eight Flexcomm Interfaces. See Fig 10.

```
/* attach 12 MHz clock to FLEXCOMM0~7 */
for (i = CHN_0; i < FLX_USART_NUM; i++)
{
  CLOCK_AttachClk(s_clk_id[i]);
}
```

Fig 10. Attach clock to eight Flexcomm Interface

    - Initialize and create transfer handles for eight UARTs with Flexcomm Interface. One UART is connected to PC for display on console and others for loopback test. See Fig 11.

```
s_demoUSARTClkFreq[CHN_0] = CLOCK_GetFreq(s_flexcommClks[CHN_0]);
USART_Init(s_demoUSART[CHN_0], &config, s_demoUSARTClkFreq[CHN_0]);
g_channel_no[0] = CHN_0;
USART_TransferCreateHandle(s_demoUSART[CHN_0], &g_uartHandle[CHN_0],
                           USART_UserCallback, &g_channel_no[0]);

config.loopback = true;
for (i = CHN_1; i < FLX_USART_NUM; i++)
{
  s_demoUSARTClkFreq[i] = CLOCK_GetFreq(s_flexcommClks[i]);
  USART_Init(s_demoUSART[i], &config, s_demoUSARTClkFreq[i]);
  g_channel_no[i] = i;
  USART_TransferCreateHandle(s_demoUSART[i], &g_uartHandle[i],
                             USART_UserCallback, &g_channel_no[i]);
}
```

Fig 11. Initialize eight UARTs

2. Transfer data with non-blocking mode

   The non-blocking mode is used for the USART TX and RX at application layer. See Fig 12 for the process of data transfer for the seven USARTS with loopback mode.

```
if(ch_no < 8)//non-blocking data transfer for one channel with loopback test
{
  USART_TransferSendNonBlocking(s_demoUSART[ch_no], &g_uartHandle[ch_no],
                                &sendXfer[ch_no]);
  USART_TransferReceiveNonBlocking(s_demoUSART[ch_no], &g_uartHandle[ch_no],
                                   &receiveXfer[ch_no], NULL);
}
else//non-blocking data transfer for seven channels with loopback test
{
  for (i = CHN_1; i < FLX_USART_NUM; i++)//send
  {
    USART_TransferSendNonBlocking(s_demoUSART[i], &g_uartHandle[i], &sendXfer[i]);
  }
  for (i = CHN_1; i < FLX_USART_NUM; i++)//receive
  {
    USART_TransferReceiveNonBlocking(s_demoUSART[i], &g_uartHandle[i],
                                     &receiveXfer[i], NULL);
  }
}
```

Fig 12. Transfer data with non-blocking mode

When the transfer of one channel is completed, an interrupt will be generated and a callback function *USART_UserCallback* will be called in this completion interrupt handler. In the callback function, the flag for the channel will be set to indicate the completion of transfer. In the loop of the background main function, the flag is checked repeatedly if the transfer of one channel is completed.

4. Demonstration

4.1 Hardware environment

- **Board**

  ✓ LPCXpresso5411x LQFP64 Rev A (OM13089)

  (link: https://www.nxp.com/products/microcontrollers-and-processors/arm-based-processors-and-mcus/lpc-cortex-m-mcus/developer-resources-/lpcxpresso-boards/lpcxpresso54114-board:OM13089)

- **Debugger**

  ✓ Integrated CMSIS-DAP debugger on the board

- **Miscellaneous**

  ✓ 1 Micro USB cable

  ✓ PC

- **Board Setup**

✓ Connect the micro USB cable between PC and J7 *link* on the board for loading and running a demo. This is also used for UART communication to UART terminal on PC.

4.2 Software environment

- **Tool chain**
  - ✓ Keil MDK 5.18.0
  - ✓ IAR embedded workbench 8.11.2
  - ✓ MCUXpresso10.0.0
- **Software package**
  - ✓ lpc54110_8_uart_int_transfer_mcuxpresso.zip
  - ✓ lpc54110_8_uart_int_transfer_keil_iar.zip
- **UART terminal program**
  - ✓ PuTTy, or similar one

4.3 Steps and result

One UART with Flexcomm Interface is used to communicate with the terminal console on PC. Thus, it shows the demo at the same time and other seven UARTs are used to transfer data in a loopback mode. The received data will be compared with the transmitted data and the comparative result will be displayed on the terminal.

The demo provides two ways for loopback test. One is to input '8' for testing seven UARTs at one time and the other is to input '1' to '7' for testing one of the seven UARTs individually.

The basic steps are:

1. **Build and download**

- Open and build the *8_uart_int_transfer* project. Before this step, make sure that the hardware environment is setup as mentioned in the Section 4.1. Hardware environment
- Download the executable file using debugger

2. **Setup for UART terminal program**

- Check the COM number which is simulated as LPC-LinkII in *Device Manager* on your computer. See Fig 13.

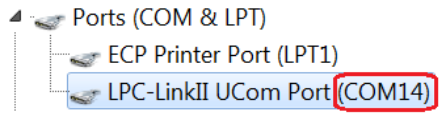

Fig 13. COM number

- Open the UART terminal program on your computer. Select the COM number as shown in the step 2 and configure the communication protocol as 115200+8+N+1.

3. **Run**

- Reset the board to run, by pressing the SW4 (reset) button on the board. See Fig 14 for the tips printed on the terminal.

```
<Flexcomm-8 UARTs Demo>
- UART0 used for this terminal communication
- UART1~7 used for loopback test
For loopback test, select all 7 UARTs by input '8' or select UART1~7 by input '1' ~ '7':
Data Length[bytes] = 32
```

Fig 14. Tips printed on the terminal

The tips indicate that the test data length is 32 bytes in the demo.
The result is shown on the terminal when any number from ‘1’ to ‘7’ is entered. For example, enter ‘3’, and the UART3 completes the loopback test. When the number ‘8’ is entered, all seven UARTs complete the loopback test. See Fig 15 for the test results.

```
UART3 loopback test passed!
UART1 loopback test passed!
UART2 loopback test passed!
UART3 loopback test passed!
UART4 loopback test passed!
UART5 loopback test passed!
UART6 loopback test passed!
UART7 loopback test passed!
```

Fig 15. Test results printed on UART terminal

Note: There is no response if the input is not between ‘1’ to ‘8’.

5. Conclusion

There are eight flexible serial communication peripherals called Flexcomm Interface on the LPC5411x series. Each one can be configured as a USART, SPI or an I^2C-bus. LPC5411x supports up to eight UARTs.

This application note gives an example of implementing eight UARTs with Flexcomm Interface based on LPC5411x SDK V2.2.1. It utilizes the interrupt and transfer mode of the chip drivers in SDK and uses non-blocking data transfer at application level. One UART is connected to PC through VCOM for demo on console and the other seven UARTs are demonstrated in a loopback mode. The VCOM UART bridge is designed on the LPCXpresso5411x board.

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: 7 March 2018
Document identifier: AN12127