AN11782

LPC82x UART secondary bootloader

Rev. 1.0 — 07 December 2015 Application note

Document information

Info	Content
Keywords	LPC82x, secondary bootloader, IAP, LPC11U68, UART
Abstract	This application note illustrates how update the LPC82x firmware without interrupting the normal flow of the execution using UART secondary bootloader (SBL). The LPC11U68 is used as a transmitter module to send the firmware data to the LPC82x via UART.

Revision history

Rev	Date	Description
1	20151207	Initial version.

Contact information

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

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

1. Overview

NXP's LPC82x is an ARM Cortex-M0+ based, low-cost 32-bit MCU family and it can operate at up to 30 MHz CPU frequency. The LPC82x supports up to 32 KB of flash memory and 8 KB of SRAM. It also Features:

- Bootloader.
- On-chip ROM APIs for ADC, SPI, I2C, USART, power configuration (power profiles) and integer divide.
- Flash In-Application Programming (IAP) and In-System Programming (ISP).

The LPC82x provides users a convenient way to update the flash content in real-time for bug fixes or product updates, it can be achieved through the following two methods:

- ISP: In-System programming is programming or re-programming of the on-chip flash memory using the bootloader software and UART0 serial port. This can be done when the part resides on an end-user board.
- IAP: In-Application programming performs erase and write operations on on-chip flash memory, as directed by the end-user application code.

Smart connectivity plays a major role in everyday user experience. The convenience of remotely updating firmware on a connected device has become a necessity. The solution should address these basic requirements:

- Good experience - the user should be able to use the device normally while the firmware is updating.
- Availability - the device should be available during firmware updates or even after a firmware update fails.

An overview of the application is as follows:

The transmitter device sends the new firmware to the host device via UART and host device programs its flash memory with the new firmware. See Fig 1.

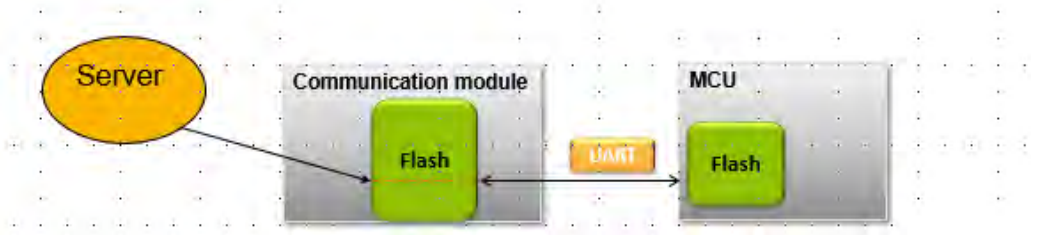

Fig 1. Application scenario

This application note explains how an MCU receives data and updates the firmware into the flash memory while still executing its original task. The system architecture is also detailed in this application note.

- LPC82x is a target device that receives the firmware via UART and programs it on the flash memory.
- Communication module is represented by the LPC11U68 development board which sends the firmware to the target via UART. See Fig 1.

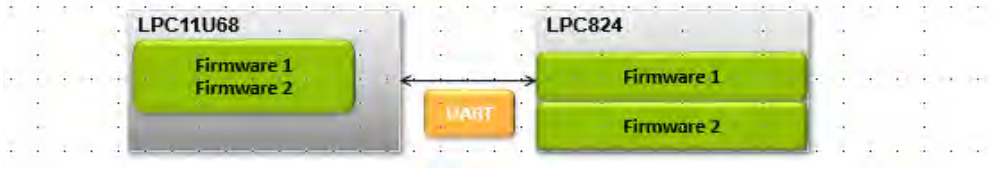

Fig 2. System architecture

2. Environment

2.1 Hardware

Board:

- LPCXpresso82x-MAX board (OM13071).
- Development board for LPC11U68 (OM13065).

Debugger:

- Keil U-Link2 for LPC11U68 projects in Keil and LPCXpresso IDEs.
- J-LINK for LPC11U68 projects in IAR IDE.
- Integrated JTAG debugger on LPCXpresso82x-MAX board for LPC82x projects in all IDEs.

Miscellaneous:

Cables to connect an LPCXpresso82x-MAX board to a development board for LPC11U68.

Board setup:

- **Jumper setting:** Default settings.
- **Board connection:** Both the boards are connected via UART interface.

Pins connections for UART:

Borad	Rx	Tx
Development board for LPC11U68	pin94	pin95
LPCXpresso82x-MAX board	P0_0	P0_4

For UART communication between both the boards, pin94 on port CN4 on the LPC11U68 should be connected to pin P0_4 on the LPC82x and pin95 on port CN4 should be connected to pin P0_0. Also the GND pins of both the boards should be connected.

NOTE: Connect UART and grounds after programming the boards.

2.2 Software

Development IDE:

- Keil uVision5 (V5.13.0.0).
- IAR Embedded Workbench for ARM V7.40.2.
- LPCXpresso V7.7.2_379.

Tool:

Use Bin2C.exe from emWin BSP to convert the binary file to a C array file. See Fig 3.

NOTE:

- emWin BSP is located in Board Support Packages (BSPs) section at https://www.lpcware.com/content/project/emwin-graphics-library.
- Bin2C.exe is located at C:\nxp\emWin\NXP_LPC1788_emWin522_BSP\NXP_emWin522_BSP\Start\Tools

Fig 3. Bin2C tool along with emWin BSP

3. Flashing the LPC82x

3.1 Flash sectors

For the LPC82x, most IAP and ISP commands are executed on sectors and specific sector numbers. Additionally, a page erase command is also supported. The size of a sector is 1 KB and the size of a page is 64 byte, therefore one sector contains 16 pages. See Fig 4 for LPC82x flash configurations.

Sector number	Sector size [KB]	Page number	Address range	16 KB flash	32 KB flash
0	1	0 -15	0x0000 0000 - 0x0000 03FF	yes	yes
1	1	16 - 31	0x0000 0400 - 0x0000 07FF	yes	yes
2	1	32 - 47	0x0000 0800 - 0x0000 0BFF	yes	yes
3	1	48 - 63	0x0000 0C00 - 0x0000 0FFF	yes	yes
4	1	64 - 79	0x0000 1000 - 0x0000 13FF	yes	yes
5	1	80 - 95	0x0000 1400 - 0x0000 17FF	yes	yes
6	1	96 - 111	0x0000 1800 - 0x0000 1BFF	yes	yes
7	1	112 - 127	0x0000 1C00 - 0x0000 1FFF	yes	yes
8	1	128 - 143	0x0000 2000 - 0x0000 23FF	yes	yes
9	1	144 - 159	0x0000 2400 - 0x0000 27FF	yes	yes
10	1	160 - 175	0x0000 2800 - 0x0000 2BFF	yes	yes
11	1	176 - 191	0x0000 2C00 - 0x0000 2FFF	yes	yes
12	1	192 - 207	0x0000 3000 - 0x0000 33FF	yes	yes
13	1	208 - 223	0x0000 3400 - 0x0000 37FF	yes	yes
14	1	224 - 239	0x0000 3800 - 0x0000 3BFF	yes	yes
15	1	240 - 255	0x0000 3C00 - 0x0000 3FFF	yes	yes
16	1	256 - 271	0x0000 4000 - 0x0000 43FF	-	yes

Fig 4. Flash sectors

3.2 IAP

IAP allows the user applications to erase and write the on-chip flash memory.

Detailed description of the IAP commands can be found in the LPC82x user manual. IAP APIs can be found in the iap.c file in the project.

4. Software design

This section gives an overview of the design of an in-Application programming demo and the software implementation for the LPC82x and LPC11U68.

The LPC82x software package contains:

1. *Firmware* that contains two tasks:
   a. Task1 blinks an LED periodically. LED can be either green or blue depending on the firmware being executed. Each LED color indicates different firmware, Firmware1 and Firmware 2.
   b. Task2 receives the program (Firmware1 or 2) via UART, writes and executes the firmware code. Secondary bootloader (SBL) uses the IAP commands to write an updated firmware on the flash memory.

2. *Secondary bootloader* that has two primary functions:
   a. Checks if the firmware is updated and valid.
   b. Run the valid firmware.

The LPC11U68 software's function is to send the firmware code to the LPC82x via UART.

4.1 Flash assignment on LPC82x

The secondary bootloader is placed at the starting address 0x0. Starting at the address of 0x00001F40, there are two pages of flash spaces that are allocated to the parameter data which is used for firmware check and execution (Note: Alternatively, parameter data can be stored in the EEPROM). Firmware1 and Firmware2 are placed after the parameter data. See Fig 5.

The code execution process follows this sequence:

1. The secondary bootloader is executed first after a reset. It selects the firmware to execute next based on the parameters stored in the flash.
2. Before the bootloader jumps to execute the firmware (Firmware 1 or 2), an interrupt vector table of the firmware is copied to the address 0x10000000 of SRAM.
3. While the firmware is being executed, if a new firmware data is received via UART, the executing firmware will finish programing on the other flash space, updating the parameters and copying the interrupt vector of the new firmware while the LED light blinks periodically. Finally, execution process will jump to execute the new firmware.
4. After new firmware starts executing, step 3 is repeated if further firmware update requests arrive, SBL overwrites the old firmware with the new one received.

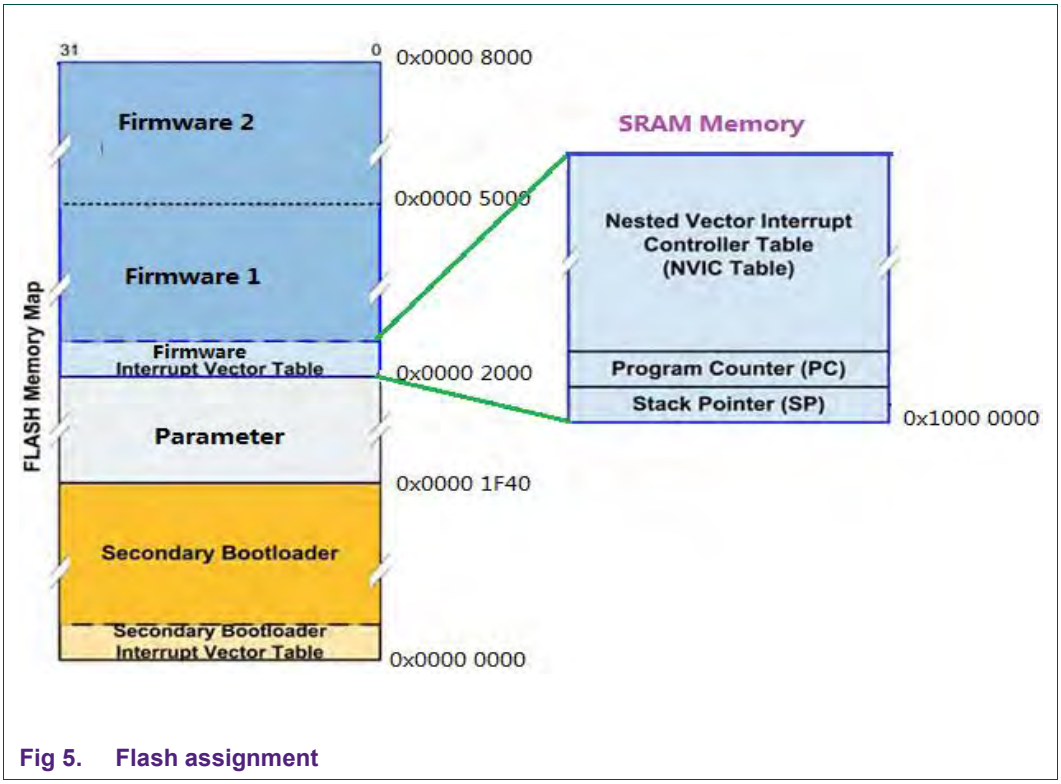

Fig 5. Flash assignment

4.2 Parameter data

Data address is assigned as shown in Fig 6.

```
#define FW_BOOT_INFO_ADDR    (0x1F40) //start address of the structure
                                      //defined by FW_BOOT_INFO_T
#define FW_CHECK_INFO_ADDR   (0x1F80) //start address of the structure
                                      //defined by FW_CHECK_INFO_T
```

Fig 6. Data address assignment

Parameter is defined by two structures shown in Fig 7.

The firmware selection process by bootloader is as follows:

1. During the SBL execution, if the parameter “new_addr” is blank (i.e 0xFFFFFFFF) in the structure “FW_BOOT_INFO_T, then the SBL fetches the address from “curr_addr” variable and executes the current firmware. Blank value of the “new_addr” indicates the new firmware is not available.

2. If the parameter “new_addr” is not blank, the SBL calculates the signature value of the parameters “new_addr”, “start_sect” and “end_sect” of the structure “FW_CHECK_INFO_T” and compares the result with the value of “signatrue_val”. In case of equal values of the structure parameters and the signature, SBL gets the address from the “new_addr” variable and executes the new firmware.

```
typedef struct {
    uint32_t  cur_addr;          //Start address of the current firmware
    uint32_t  new_addr;          //Start address of the new firmware
} FW_BOOT_INFO_T;

typedef struct {
    uint32_t  new_addr;          //Start address of the new firmware
    uint16_t    start_sect;      //Start sector of the new firmware
    uint16_t  end_sect;          //End sector of the new firmware
    uint32_t  signatrue_val;     //Signature value of the new firmware
} FW_CHECK_INFO_T;
```

Fig 7. Parameter structure

5. Generate C array from image file

The firmware image (binary format) on the LPC82x needs to be converted to C array which can be included and built along with the C file on the LPC11U68. Fig 8 shows how to include the C array in a .c file on the LPC11U68 for IAR and Keil IDE. The LPC11U68 will send the data in the form of a C array to the LPC82x via UART.

```
#ifdef __ICCARM__
const
#include "user_app_example1_IAR.c"
const
#include "user_app_example2_IAR.c"
#else
const
#include "user_app_example1_KEIL.c"
const
#include "user_app_example2_KEIL.c"
#endif
```

Fig 8. C arrays included in .c file on the LPC11U68

The steps to generate the C array from the image file in different IDEs are listed here.

5.1 Keil uVision

Keil supports generation of the C array from the .axf file after building the project, steps are as follows:

1. Open Keil project “all_example_lib” located at: ..\fw_IAP__keil_iar_v1\lpc824_fw_IAP\applications\keil_uvision.
2. Compile the Chip and the Board library projects.
3. Compile the ‘Bootloader_example’ project.
4. Set the ‘user_app_example’ as an active project.
5. Select the user_app_example1 in the dropdown box to create the C array for the firmware 1, see Fig 9.
6. Open “options for target”.
7. Click on the “User” tab.
8. Configure as shown in Fig 10 to generate the C array file “user_app_example1_KEIL.c”.
9. Also generate the C array file “user_app_example2_KEIL.c” for firmware 2 following steps 1 to 8. The only change is to select the ‘user_app_example2’ in the dropdown box and use “example2” in place of “example1”.
10. Copy the two C array files, ‘user_app_example1_KEIL.c’ and ‘user_app_example2_KEIL.c’, to the directory of the LPC11U68 software package: lpc11u68_fw_IAP_Host\applications\examples.

Note: Change the name of an array in “*user_app_example**2**_KEIL.c*” to “**LR1**”.

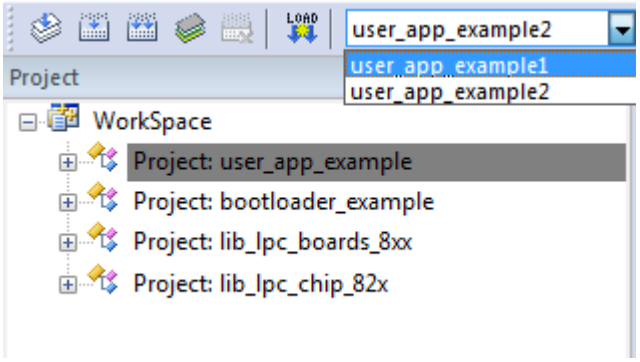

Fig 9. Select user_app_example1 in Keil

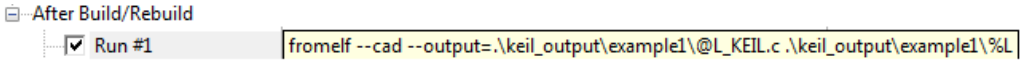

Fig 10. Generate C array file in Keil IDE

5.2 IAR Embedded Workbench for ARM

IAR does not provides the functionality to generate the C array but includes functionality to generate the binary file. The process to convert the binary file to the C array file using a tool follows these steps:

1. Open “all_example_lib” workspace in IAR located at: ..\fw_IAP__keil_iar_v1\lpc824_fw_IAP\applications\iar_ewarm_projects.
2. Rebuild the Chip and the Board library projects.
3. Rebuild the ‘Bootloader_example’ project
4. Set ‘user_app_example – Green Blinky’ as an active project, see the Fig 11.
5. Open “options…”.
6. Click on “Output Converter” under “Category”
7. Configure as shown in the Fig 12 to generate the binary file
8. Rebuild the ‘user_app_example – Green_Blinky’ project
9. Convert the binary file to C array file “user_app_example1_IAR.c” using the tool. Use the Bin2C tool along with emWin BSP as shown in Fig 3.
10. Also generate the C array file “user_app_example2_IAR.c” for the firmware 2 following steps 1 to 9, only changing selection to ‘user_app_example – Blue_blinky’ in the dropdown box, see Fig 11.
11. Copy the two C array files to the directory of the LPC11U68 software package: lpc11u68_fw_IAP_Host\applications\examples.

Note: Change the name of an array in “*user_app_example1_IAR.c*” to “**LR0**” and an array in “*user_app_example2_IAR.c*” to “**LR1**”.

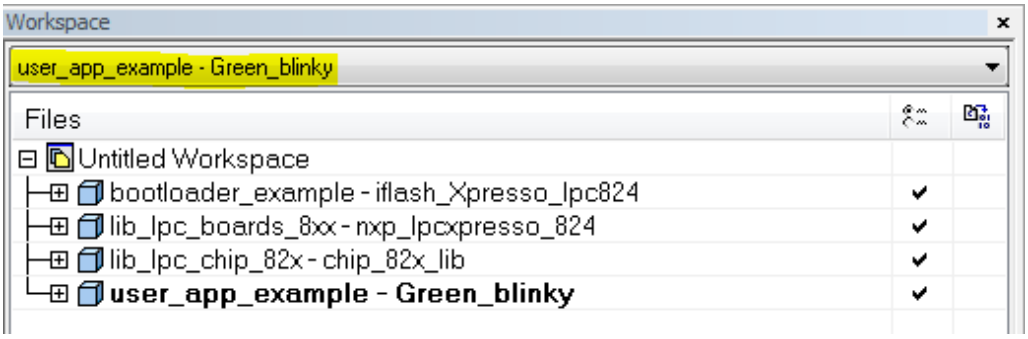

Fig 11. Select user_app_example – Green_blinky in IAR

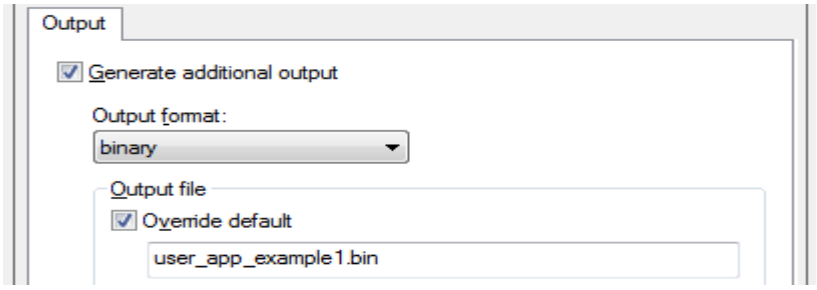

Fig 12. Generate binary file in IAR IDE

5.3 LPCXpresso

LPCXpresso does not provide functionality to generate the C array but it provides the functionality to generate the binary file. Follow the steps below to convert the binary file to the C array file using the tool.

1. Open the LPCXpresso IDE.
2. Import the “lpc824_fw_iap_xpresso_v1” project, located in software package, to LPCXpresso workspace using ‘Import Project’ option in the Quickstart Panel window.
3. Compile the Chip and the Board libraries.
4. Compile the Bootloader project.
5. Open the properties for the project ‘user_app_example’.
6. Click on the “Settings” in “C/C++ Build”
7. Select the tab “Build steps” and click the button “Edit…” for “Post-build steps”
8. Configure as shown in Fig 13 to generate the binary file.
9. Compile the ‘user_app_example’ project. (This will compile and create the .bin file for Example1)

10. To compile and create the .bin file for an Example2, go to project properties by right clicking on the project in the Project Explorer window.
11. In C\C++ Build, go to settings and click on the “Manage Configurations…” button.
12. Select the ‘user_app_example2’ from the table and click on the “Set Active” button to activate example2 in the project ‘user_app_example’, save the settings by clicking on OK button.
13. Compile the ‘user_app_example’ to create .bin file for example2.
14. Convert the binary files to C array files “*user_app_example1.c*” and ‘user_app_example2.c’ using the Bin2C along with emWin BSP tools as shown in Fig 3.
15. Copy the two C array files in ‘example/src/’ folder in project workspace for “lpc11u68_fw_IAP_Host_xpresso_v1”.

    For example: ..\LPCXpresso_7.9.2_493\workspace1\fw_TX_example\example\src\user_app_example1.

Note: Need to change the name of an array in “*user_app_example1.c*” to “**LR0**” and an array in “*user_app_example2.c*” to “**LR1**”.

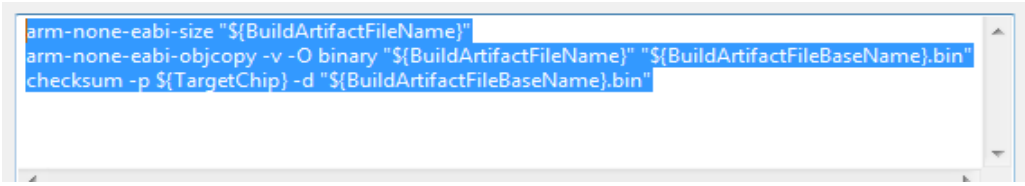

Fig 13. Generate binary file in LPCXpresso IDE

6. Demonstration

This section briefly describes how to demonstrate the remote IAP solution.

Preconditions:

1. Setup hardware and software environments. See section "2. Environment."
2. Build all the projects successfully.

NOTE: Make sure the C array is placed in the directory of the LPC11U68 software package before building the LPC11U68 project. For more details on how to do that, see section "5.Generate C array from image file".

Steps:

1. Download the 'bootloader_example' project at the 0x0 address and the 'user_app_example' project starting at 0x00002000 address (referred to as Example1) on the LPC82x board from the "all_example_lib" workspace.

   In the KEIL and LPCXpresso, example project's name is "user_app_example1".

   In the IAR, example project's name is "user_app_example – Green blinky".
2. Download the 'fw_TX_example' project from the "lib_examples" workspace on the LPC11U68 board, located at:

   ..\fw_IAP__keil_iar_v1\lpc11u68_fw_IAP_Host\applications\keil_uvision_projects
3. The onboard LED D1 will blink with green color when LCP82x board is powered on.
4. Power on or reset the LPC11U68 board. The binary data of the other example starting at the address of 0x00005000 (referred to as Example2) will be sent to the LPC82x via UART.
5. Within a second or two, the LED D1 will change the color to blue from green on the LPC82x board. Change in LED color indicates Example2 is updated and is running on the LPC82x board.
6. After the firmware2 is written to the LPC82x board, every time the board restarts firmware2 is executed. In other words, the secondary boot loader starts executing the latest firmware.
7. Reset the LPC11U68 board for a reboot, now Example1 is sent to the LPC82x board for firmware update.
8. LED D1 will change the color to green from blue on the LPC82x board within the time period of a second.

7. Conclusion

This application note describes a convenient way to update the flash content of LPC82x using the UART SBL. The flexibility allows users to perform a firmware update in real time without a device restart. When device receives the new firmware, the SBL replaces the older firmware with the newer firmware in the flash memory. Device receives the new firmware via UART and jumps to execute it after the flash memory is programmed.

8. Legal information

8.1 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.

8.2 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.

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.

8.3 Trademarks

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

9. Contents

1. Overview 3
2. Environment 4
2.1 Hardware 4
2.2 Software 5
3. Flashing the LPC82x 6
3.1 Flash sectors 6
3.2 IAP 6
4. Software design 7
4.1 Flash assignment on LPC82x 7
4.2 Parameter Data 8
5. Generate C array from image file 10
5.1 Keil uVision 10
5.2 IAR Embedded Workbench for ARM 11
5.3 LPCXpresso 12
6. Demonstration 14
7. Conclusion 15
8. Legal information 16
8.1 Definitions 16
8.2 Disclaimers 16
8.3 Trademarks 16
9. Contents 17

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

Date of release: 7 December 2015
Document identifier: AN11782