AN11772

LPC11Cxx CAN secondary bootloader

Rev. 1.0— 3 December 2015 Application note

Document information

Info	Content
Keywords	LPC11C12FBD48/301, LPC11C14FBD48/301, LPC11C22FBD48/301, LPC11C24FBD48/301, Secondary bootloader, CAN, IAP
Abstract	This application note illustrates how to update user application code using the CAN bus interface. The secondary bootloader uses IAP ROM APIs to update the firmware in the flash.

Revision history

Rev	Date	Description
1.0	20151203	Initial version

Contact information

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

1. Introduction

The LPC11Cxx provides users a convenient way to update the flash content in real time. This can be achieved using following two methods:

- ISP: In-System programming mode can be used to program or re-program the on-chip flash memory using the bootloader software and UART0 serial board. This can be done when the part resides on the 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.

A secondary bootloader (SBL) is used to download the user application code to flash memory using channels other than the standard UART0, which is used by the internal bootloader. The primary bootloader is the firmware that resides in the microcontroller's boot ROM block and is executed at power-up and reset. After boot ROM's execution, the secondary bootloader is invoked, which executes the end-user application.

This application note explains the operation of the included CAN SBL software example which uses IAP write and erase commands to perform field updates.

2. LPC11Cxx Flash Programming

2.1 In-Application programming (IAP)

In-Application programming (IAP) performs erase and write operations on the on-chip flash memory, as directed by the secondary bootloader.

A secondary bootloader is programmed into the on-chip flash. The secondary bootloader controls the initial operation after reset and also provides the means to accomplish programming of the flash memory. This could be initial programming of a blank device, erasing and re-programming of a previously programmed device or programming of the flash memory by an application program running in the system.

Using IAP, users can update the application code by various communication interfaces such as UART, USB or Ethernet. Flash sectors, which are not being used for the secondary bootloader or the user application, can be used as non-volatile data storage.

While the application is running, the user can update some portion of the code using IAP commands, called field updates, without restarting the device.

2.2 IAP commands of LPC11Cxx

IAP parameter passing is illustrated in Fig 1. The IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. The result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for results by passing the same pointer in registers r0 and r1. The parameter table should be large enough to hold all the results in case the number of results is more than the number of parameters.

The number of parameters and results vary based on the IAP commands.

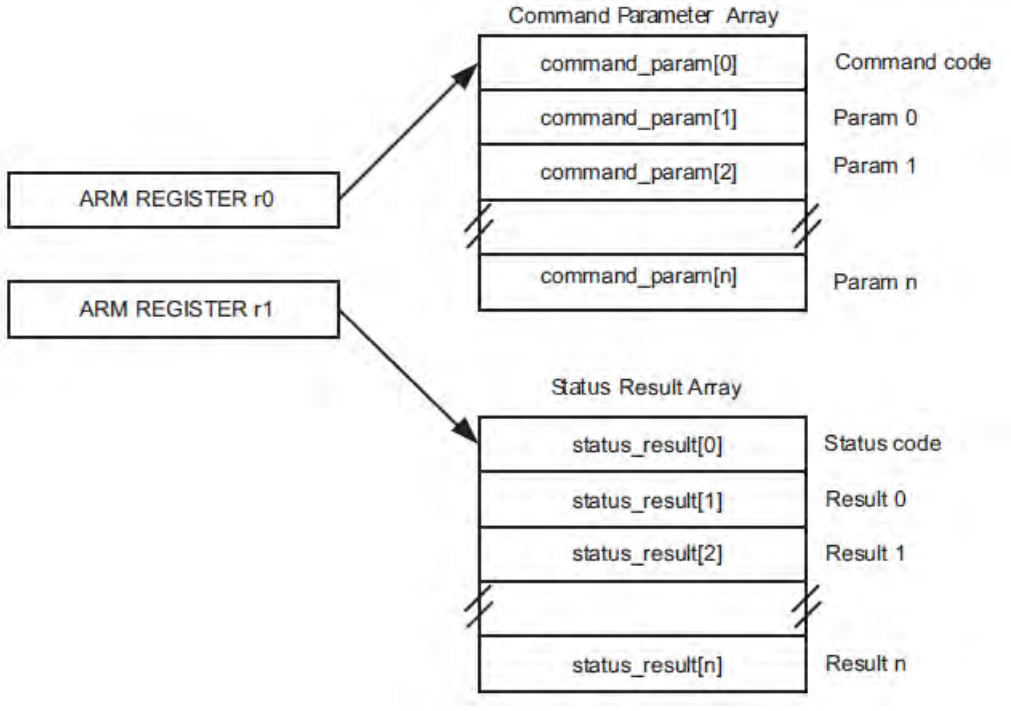

Fig 1. IAP parameter passing

The IAP commands and codes are listed in Fig 2.

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50 (decimal)	Table 396
Copy RAM to flash	51 (decimal)	Table 397
Erase sector(s)	52 (decimal)	Table 398
Blank check sector(s)	53 (decimal)	Table 399
Read Part ID	54 (decimal)	Table 400
Read Boot code version	55 (decimal)	Table 401
Compare	56 (decimal)	Table 402
Reinvoke ISP	57 (decimal)	Table 403
Read UID	58 (decimal)	Table 404
Erase page	59 (decimal)	Table 405

Fig 2. IAP command summary

2.3 Interrupts during IAP

The on-chip flash memory is not accessible during erase/write operations. When the user application code starts executing, interrupt vectors from the user flash area remain active. The user should either disable interrupts or ensure that the user's interrupt vectors residing in the RAM are active, before making the flash erase/write IAP call. The IAP code does not use or disable interrupts.

2.4 RAM used by IAP command handler

For this demo, flash erase/write are the major operations handled by IAP commands. IAP commands requires 32 bytes of space from address 0x10001FCD to 0x10001FFF of the on-chip RAM for execution. The user program should not use this space if IAP flash programming is permitted in the application.

2.5 Flash programming with a Secondary Bootloader (SBL)

For the applications that utilize the SBL for field updates, the boot ROM executes after the restart and it actuates the SBL that jumps to the application code. This also allows the SBL to update the application code, if requested by the user, or start executing the code in the application code space. Since the write and erase IAP commands are sector specific, the SBL and application code should be programmed into the flash sectors that are mutually exclusive, but both must reside in the on-chip flash. Therefore, the application should be programmed/erased in the flash sectors where application code is residing, separate from the bootloader.

The following table shows the correspondence between sector numbers and memory addresses for the LPC111x/LPC11Cxx devices. It also suggests that, in order to make any modifications (as small as one byte) to a particular sector, the entire sector must be erased.

Sector number	Sector size [kB]	Address range	LPC1110 (4 kB flash)	LPC1111 (8 kB flash)	LPC1112/ LPC11C12/L PC11C22 (16 kB flash)	LPC1113 (24 kB flash)	LPC1114/ LPC11C14/L PC11C24 (32 kB flash)
0	4	0x0000 0000 - 0x0000 0FFF	yes	yes	yes	yes	yes
1	4	0x0000 1000 - 0x0000 1FFF	-	yes	yes	yes	yes
2	4	0x0000 2000 - 0x0000 2FFF	-	-	yes	yes	yes
3	4	0x0000 3000 - 0x0000 3FFF	-	-	yes	yes	yes
4	4	0x0000 4000 - 0x0000 4FFF	-	-	-	yes	yes
5	4	0x0000 5000 - 0x0000 5FFF	-	-	-	yes	yes
6	4	0x0000 6000 - 0x0000 6FFF	-	-	-	-	yes
7	4	0x0000 7000 - 0x0000 7FFF			-	-	yes

Fig 3. On-chip flash sectors in LPC11Cxx devices

3. In-Application programming (IAP) demo

3.1 Objective

This section provides a software example which includes three projects:

1) CAN IAP secondary bootloader that receives field updates via CAN bus.
2) An application that sends the field updates via CAN bus.
3) A simple blinky software example which can be used to easily verify whether or not the application code space was updated.

The software is based on NXP's LPCOpen software platform and supports Keil MDK, IAR EWARM and LPCXpresso.

LPCXpresso can be found at https://www.lpcware.com/lpcxpresso/download

3.2 Requirements

1) Keil MDK, IAR EWARM or LPCXpresso.
2) Two Keil MCB1000 demo boards with the LPC11Cxx MCUs, can be found at http://www.keil.com/MCB1000/

3.3 Hardware setup

Fig 4. MCB1000 demo board with LPC11Cxx MCUs

This demo requires two Keil MCB1000 demo boards: one board serves as a CAN BUS transmit board and the other as a receiver board. Connect the CANH signal and the CANL signal on the P4 ports of the boards for CAN communication.

Use the Link2 (CMSIS-DAP) debugger, Link2 introduction can be found at http://www.lpcware.com/lpclink2

Connect UART port on each board to PC, Tera Term serves as UART terminal output.
Tera Term UART1/2 Setting: 115200 baud, 8N1.
And in setup -> Terminal, set Transmit New line with CR+LF, See Fig 5.

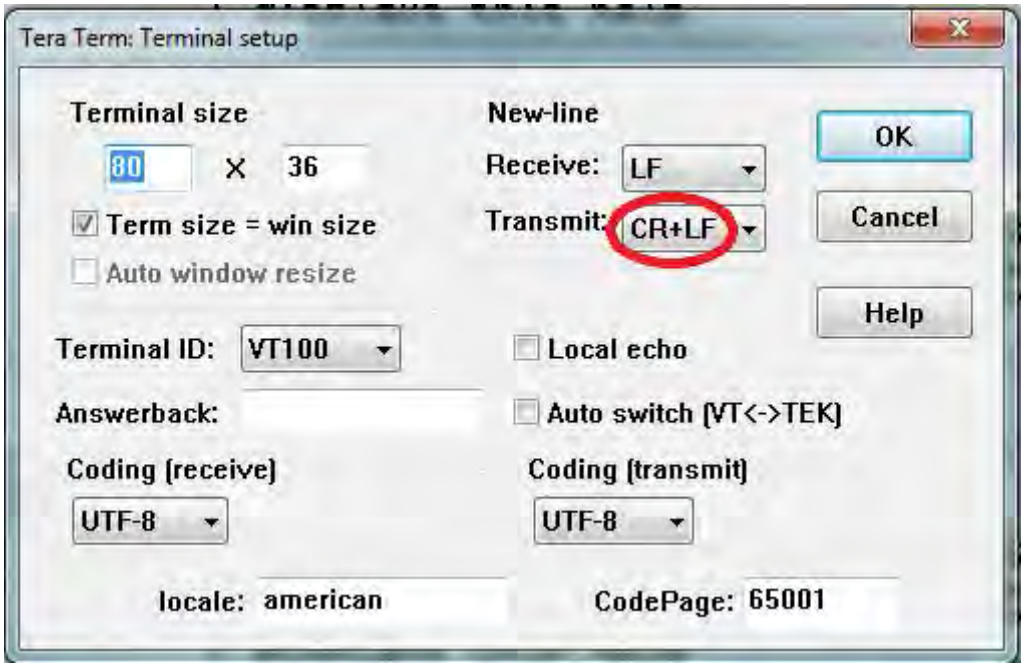

Fig 5. Tera Term setup

3.4 Demo under MDK

MDK version is 5.14.

1) Program the blinky project to the address 0x4000 (sector 4) on the TX board.

   Blinky project is in the “LPCOpen_CAN_BootLoader” workspace located at ..\AN_LPC11C00_CAN_IAP\keil_iar_can_iap\applications\lpc11xx\keil_uvision_projects\nxp_lpcxpresso_11c24

   Programing method:

   - By debug download.
   - By other tools to program the HEX file, e.g. Flashmagic.

   Fig 6 shows the memory configuration for the user application.

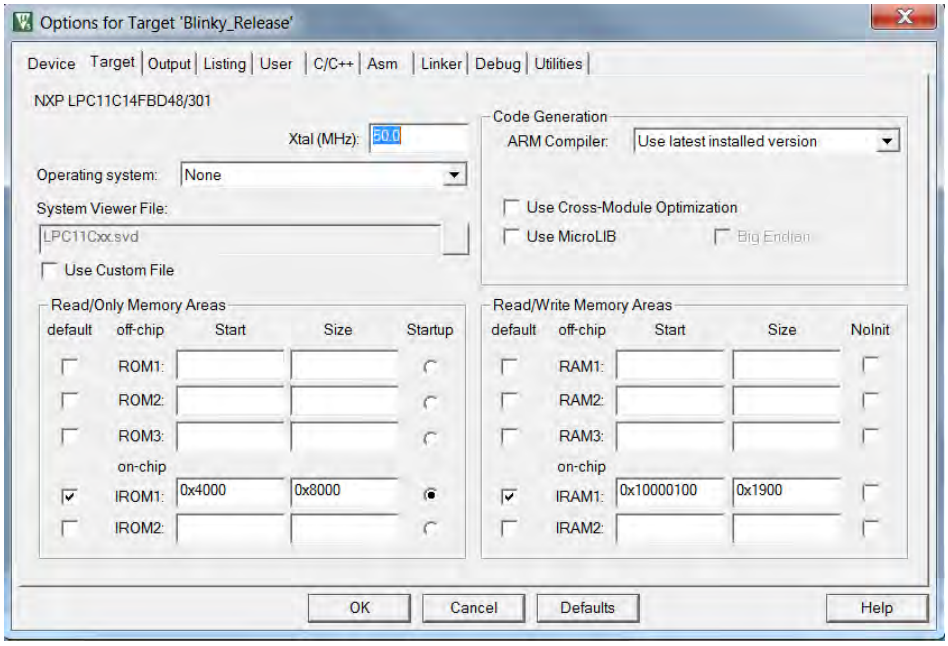

Fig 6. Memory configuration for the user application under MDK

RAM size settings needs to be configured as it is shown in Fig 6 because on-chip RAM from the address 0x1000 0050 to 0x1000 00B8 is used by the CAN API.

NOTE: When the demo project is open in a different version of MDK, it may perform migration operation which may overwrite memory configuration settings. Therefore, it is important to check the memory configuration to ensure the settings are correct after opening the demo project.

NOTE: The onboard LED D1 will not start blinking until after user issues command using serial port from command terminal to start running the code from the address 0x4000 after downloading the blinky code.

2) Download the firmware for TX and RX board.

- Need to extract two lib projects (board_lib & chip_lib) to compile the demo project.
- Two lib projects are located on:
  - ..\keil_iar_can_iap\software\lpc_core\lpc_board\boards_11xx\nxp_lpcxpresso_11c24
  - ..\keil_iar_can_iap\software\lpc_core\lpc_chip\chip_11xx

Fig 7 shows the memory configuration for the project Tx and Rx.

NOTE: If linker cannot find the library, try relocating the application (keil_iar_can_iap) folder to C:\ to shorten the path for linker. Linker cannot find the library file if the file path is longer than a certain number of characters.

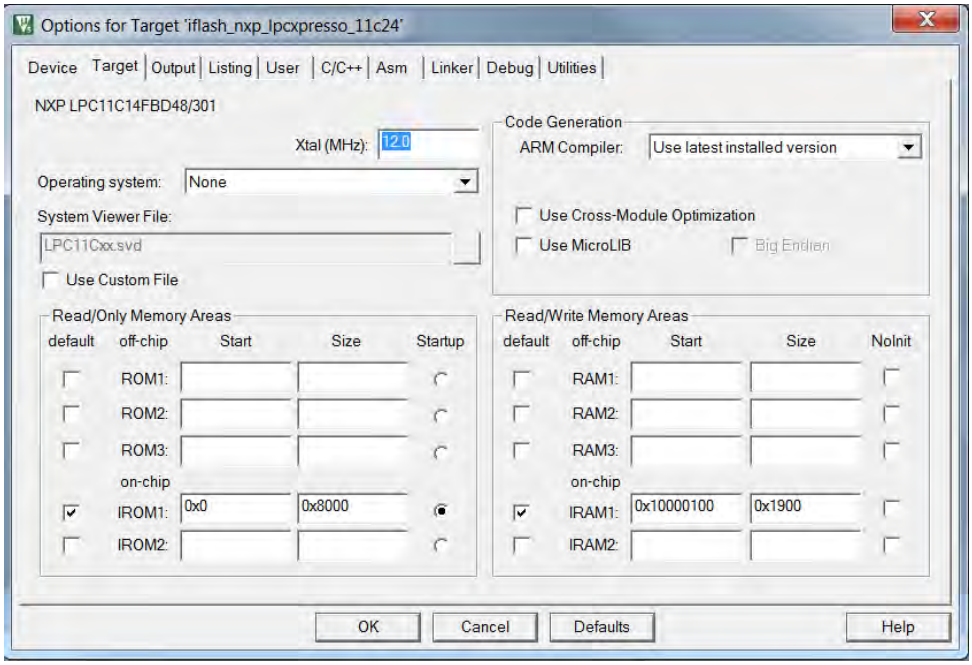

Fig 7. Memory configuration for the user application under MDK

3) Power on or reset TX board. TX board will send a menu through serial port which will be displayed on the serial terminal, see Fig 8.

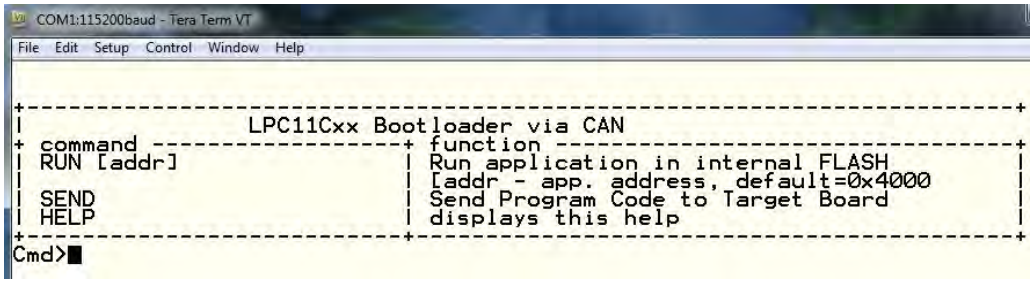

Fig 8. TX board start menu

4) Enter “run 0x4000” command in the TX board terminal, and LED0 starts blinking, see Fig 9.

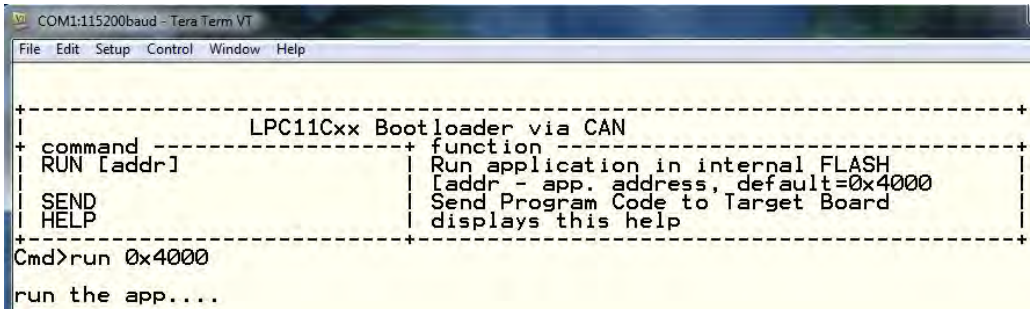

Fig 9. TX board Run APP

5) To start the IAP through CAN, reset the TX board, go to start menu again and enter ”send” command. Before entering “send” command, make sure RX board is in the “demo” status after setting up the RX board, step 8) see Fig 10.

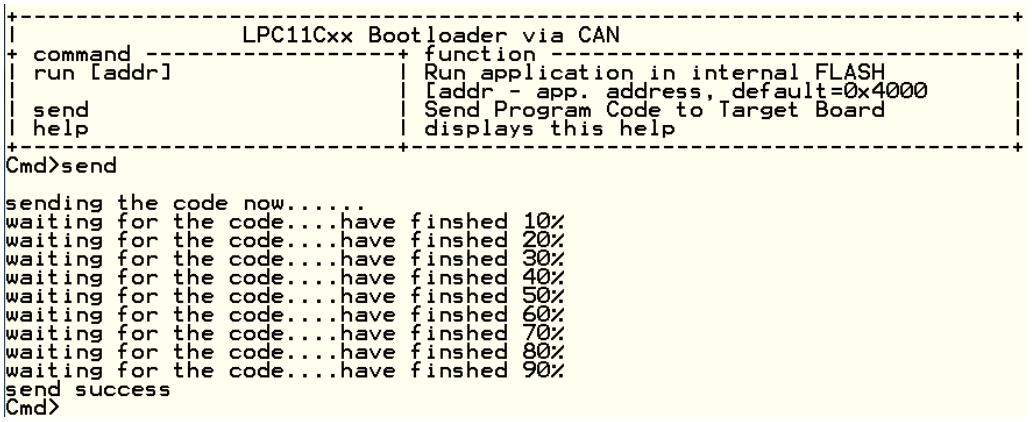

Fig 10. TX board send code at 0x4000

6) RX board, after power on or reset, send a menu through serial port which will be displayed on the serial terminal. Type “help” or “info” for more information, see Fig 11.

7) Before the RX board receives the code from TX board, enter command “blkchk 4” in terminal to see if the sector 4 in the memory is blank.

```
+----------------------------+-------------------------------------------+
Cmd>help


+-------------------------------------------------------------------------+
|                LPC11Cxx Bootloader via CAN
+ command ------------------+ function ----------------------------------+
| info                      | display the bootloader info               |
| readID                    | read part ID                              |
| blkchk [sectors]          | check if flash sectors are blank          |
| erase [sectors]           | erase flash sectors                       |
| run [addr]                | Run application in internal FLASH         |
|                           | [addr - app. address, default=0x4000      |
| demo                      | Run Demo Program, Wait Code...            |
| help                      | displays this help                        |
+---------------------------+-------------------------------------------+
Cmd>info

  Bootloader:
              Entry: 0x0
              Version: Aug 19 2014
Cmd>blkchk 4

sector 4 is blank
Cmd>run 0x4000

run the app.....
```

Fig 11. RX board start menu

8) Enter command “demo” in the terminal for RX board. After the command is given, RX board will wait for TX board to send the code. As soon as the TX board starts sending the code, Step 5), RX board will display receiving information synchronously. See Fig 12.

9) After transmit/receive is completed, enter command “blkchk 4” to see that the sector 4 is written successfully and is not blank.

10) Enter command “run 0x4000”. Consequently, the CPU will jump to 0x4000 to execute the application and the LED0 on RX board will blink.

```
+----------------------------+-------------------------------------------+
Cmd>demo

waiting for the code....have finshed 0%,please input send on the B1(TX board)
waiting for the code....have finshed 10%
waiting for the code....have finshed 20%
waiting for the code....have finshed 30%
waiting for the code....have finshed 40%
waiting for the code....have finshed 50%
waiting for the code....have finshed 60%
waiting for the code....have finshed 70%
waiting for the code....have finshed 80%
waiting for the code....have finshed 90%
waiting for the code....have finshed 100%
code has copied in the secoter 4
Cmd>blkchk 4

sector 4 is not blank
Cmd>run 0x4000

run the app.....
```

Fig 12. RX board demo

3.5 Demo under IAR

IAR EWARM version is 7.303.
Demo under IAR is same as under MDK.

NOTE: It is required to power-cycle (reconnect the power cable) the board after downloading the firmware using Link2. However, power-cycle is not required for other debuggers (ULINK2, J-LINK).

1) Erase TX board's memory before downloading the blinky project, see Fig 13.

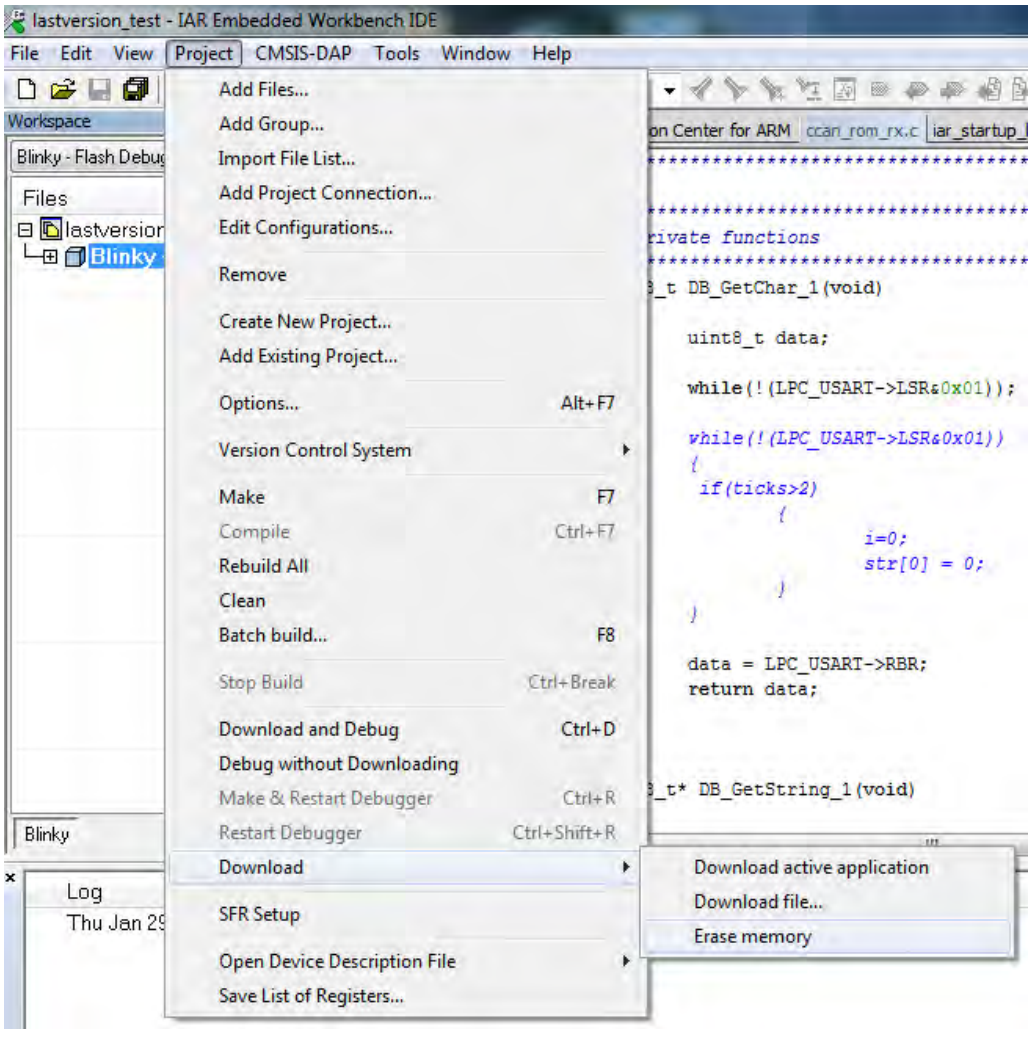

Fig 13. Erase TX board memory before downloading the blinky code.

2) Download the blinky project to the TX board at the memory address 0x4000, see Fig 14 & Fig 15.

The blinky project can be found in the “LPCOpen_CAN_BootLoader” workspace located at

..\AN_LPC11C00_CAN_IAP\keil_iar_can_iap\applications\lpc11xx\iar_ewarm_projects\nxp_lpcxpresso_11c24.

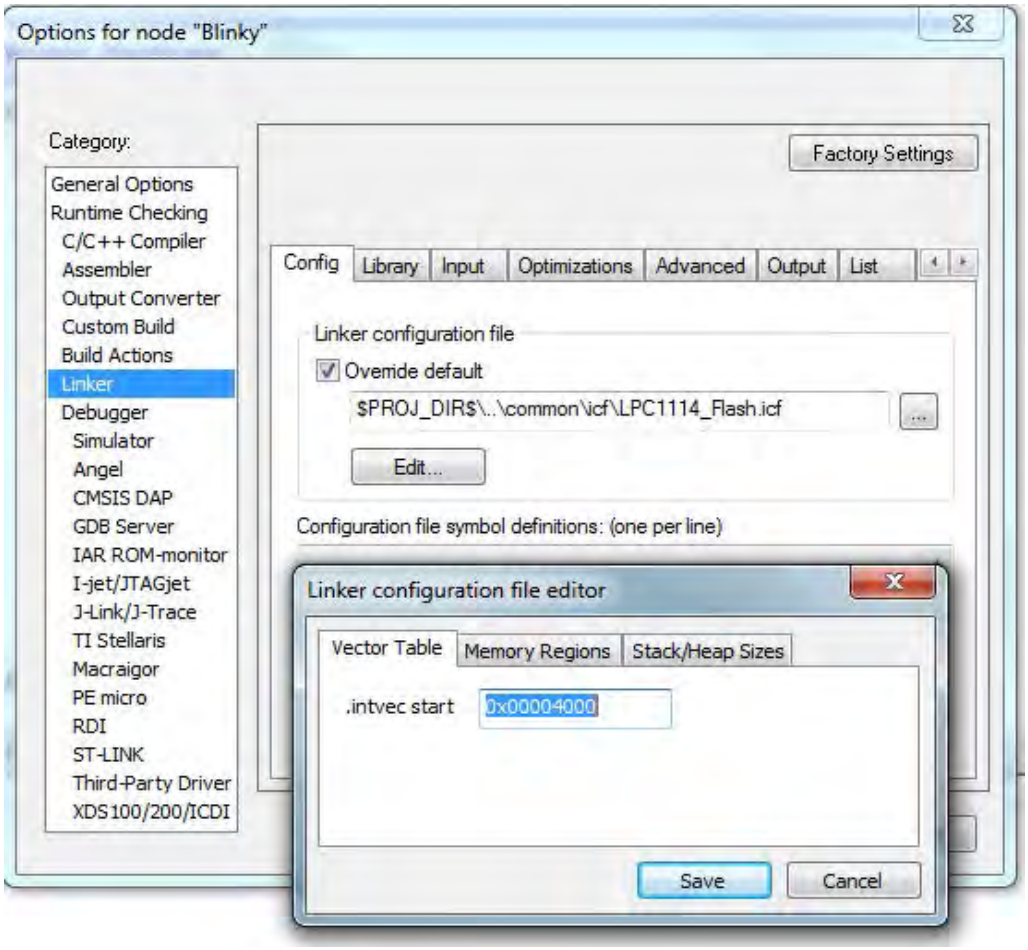

Fig 14. Memory configuration for the APP under IAR – vector table

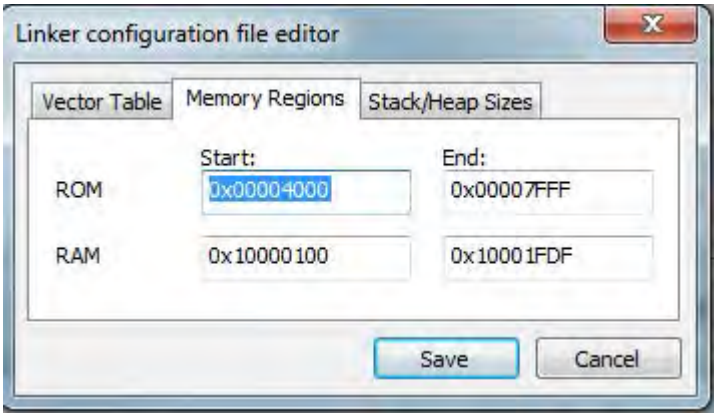

Fig 15. Memory configuration for the APP under IAR – memory regions

3) Open the TX and RX project named 'LPCOpen_CAN_BootLoader' for IAR, located at: ..

   \keil_iar_can_iap\applications\lpc11xx\iar_ewarm_projects\nxp_lpcxpresso_11c24.

4) Rebuild the Chip library and Board library projects respectively before rebuilding the TX project.

5) Download the TX project to the TX board at the 0x0000 memory location.

6) Rebuild the RX project.

7) Erase RX board's memory before downloading the RX code, see Fig 13.

8) Download RX project to the RX board at 0x0000 memory location.

Other steps are same as '3.3 Demo under MDK' after downloading the projects to both the boards step 3) to step 10) in section 3.4.

3.6 Demo under LPCXpresso

LPCXpresso version is v7.5.0.

Demo for LPCXpresso is similar to MDK and IAR. However, LPCXpresso generates larger compiled files than Keil and IAR. Download the APP project to the 0x6000 memory address to give enough space to accommodate SBL. Modify "run 0x4000" command to "run 0x6000", modify "blkchk 4" command to "blkchk 6", modify "erase 4" command to "erase 6".

APP project "nxp_lpcxpresso_11c24_periph_blinky" is included in the demo TX and RX project workspace.

1) Import 'lpcxpresso_can_iap.zip' to the LPCXpresso from QuickStart panel.

2) Build all the projects that are available in the 'lpcxpresso_can_iap.zip' folder.

3) Go to the project properties to check if the memory location for the flash is set to 0x6000.

NOTE: FLASH/RAM settings information for each project is displayed in the Fig 16.

4) Click on the ‘program flash’ button from the panel on top, see Fig 18, and select an output .axf file for the project in ‘Select file’ input box under ‘Program flash memory’ tab to download the project to the flash memory of the TX board.

NOTE: The output .axf file can be found in the ‘Debug’ folder in the project’s workspace.

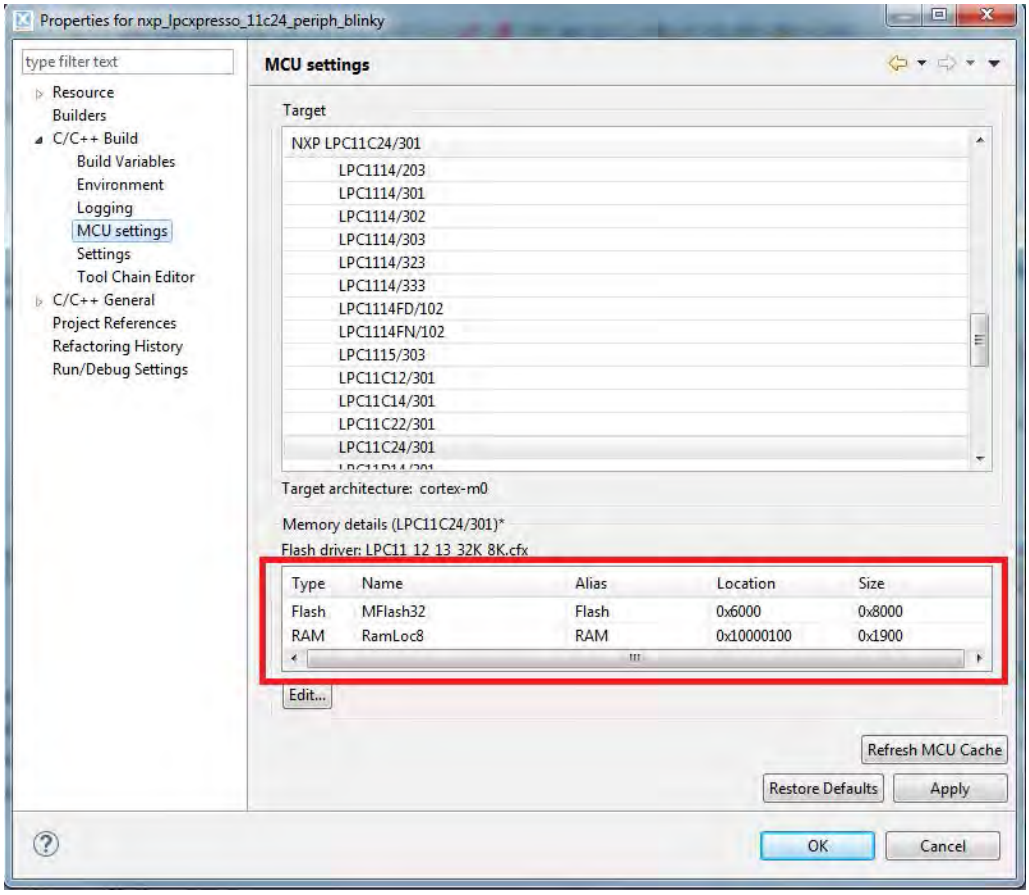

Fig 16. Memory configuration for the user application under LPCXpresso

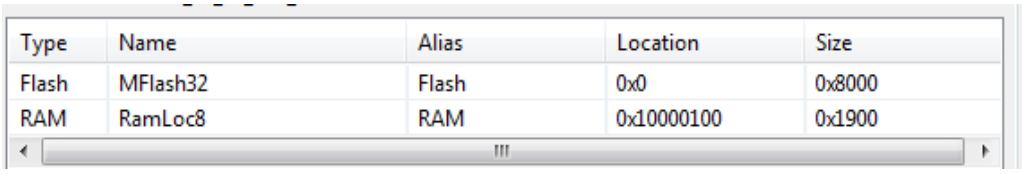

Fig 17. Memory configuration for TX and RX project under LPCXpresso

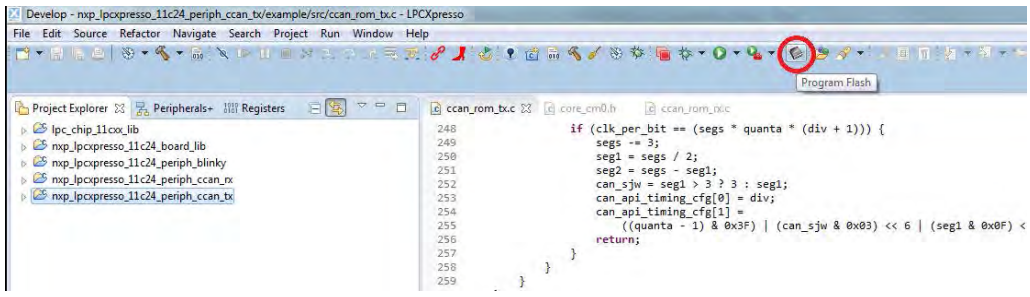

Fig 18. Flash operation under LPCXpresso

5) After downloading the periph_blinky project to the TX board flash memory, download TX and RX projects to the respective boards at 0x0000 memory location.

6) Other steps are same as '3.3 Demo under MDK' after downloading the projects to both the boards from steps 3) to 10) in section 3.4.

NOTE: Modify "run 0x4000" command to "run 0x6000", modify "blkchk 4" command to "blkchk 6", modify "erase 4" command to "erase 6".

4. Conclusion

The LPC11Cxx provides users a convenient way to update the flash content in real time using an In-Application Programming (IAP) via SBL and CAN bus. This flexibility allows users to update the system firmware without using the traditional approach to program the flash. Use of this SBL eliminates the need to restart the device to write and execute the updated firmware. The CPU jumps to the flash sectors with the updated firmware with the help of the SBL after the firmware update.

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. 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.

5.3 Trademarks

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

6. List of figures

7. Contents

Date of release: 3 December 2015
Document identifier: AN11772