AN11388

Using LPC800 In-Application Programming

Rev. 1.1 — 24 September 2013 Application note

Document information

Info	Content
Keywords	LPC810M021FN8; LPC811M001JDH16; LPC812M101JDH16; LPC812M101JD20; LPC812M101JDH20, LPC800 IAP
Abstract	This application note describes how to use IAP commands of the LPC800 device family.

Revision history

Rev	Date	Description
1.1	20130924	Updated demo code package.
1	20130909	Initial version.

Contact information

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

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

1. Introduction

The LPC800 devices have up to 16 kB of flash memory. The smallest operating unit in the flash memory is called a 'page' and the size of each page is 64 bytes.

Users can utilize In-Application Programming (IAP) commands to update flash or to store information in the on-chip flash. Possible uses of IAP include:

- Field upgradable firmware
- EEPROM emulation
- Application configuration storage
- Data storage

The benefits of using IAP are:

- Capability of the microcontroller to reprogram its own flash memory
- No external circuitry is required
- Uses boot ROM code (less code needed in flash)

IAP does have some limitations. For example, IAP has a slow erase time (100 ms) and write time (1 ms), regardless of the flash size that is being updated.

2. Flash specifications

2.1 Flash structure

The flash memory is comprised of 16 sectors, with each sector containing 16 pages. The size of a sector is 1 kB and the size of a page is 64 bytes. Fig 1 shows the flash structure in the LPC800.

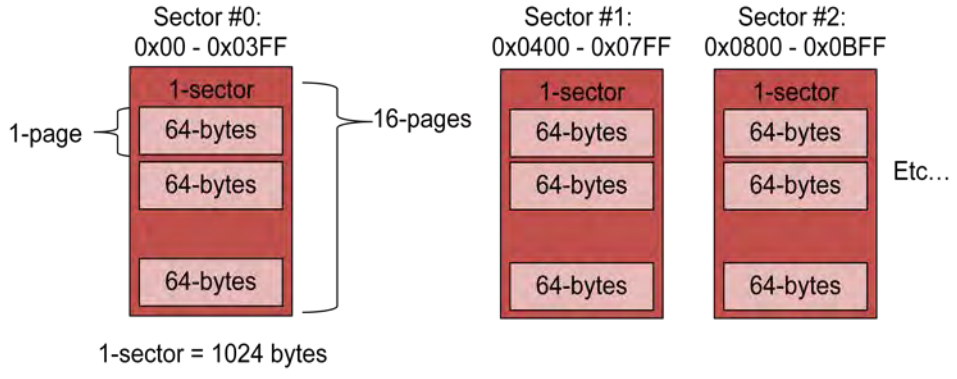


Fig 1. LPC800 flash structure

Most IAP and ISP commands operate on sectors and specific sector numbers. Fig 2 shows the correspondence between page numbers, sector numbers, and memory addresses.

Sector No	Sector Size (kBytes)	Page Number	Address Range	4kB Flash	8kB Flash	16kB Flash
0	1	0 – 15	0x0000 0000 - 0x0000 03FF	Yes	Yes	Yes
1	1	16 – 31	0x0000 0400 - 0x0000 07FF	Yes	Yes	Yes
2	1	32 – 47	0x0000 0800 - 0x0000 0BFF	Yes	Yes	Yes
3	1	48 – 63	0x0000 0C00 - 0x0000 0FFF	Yes	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
9	1	144 – 159	0x0000 2400 - 0x0000 27FF	-	-	Yes
10	1	160 – 175	0x0000 2800 - 0x0000 2BFF	-	-	Yes
11	1	176 – 191	0x0000 2C00 - 0x0000 2FFF	-	-	Yes
12	1	192 – 207	0x0000 3000 - 0x0000 33FF	-	-	Yes
13	1	208 – 223	0x0000 3400 - 0x0000 37FF	-	-	Yes
14	1	224 – 239	0x0000 3800 - 0x0000 3BFF	-	-	Yes
15	1	240 – 255	0x0000 3C00 - 0x0000 3FFF	-	-	Yes

Fig 2. LPC800 flash address mapping

2.2 Flash specifications

The erase time and programming time of the flash is fixed regardless of the number of sectors and pages. Flash Error Correction Code (ECC) is automatically added during erase and programming. It is also verified when flash is read.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
N_{endu}	Endurance		10000	100000	-	cycles
t_{ret}	Retention time	Powered	10	20	-	years
		Unpowered	20	40	-	Years
t_{er}	Erase time	Sector or multiple consecutive sectors	95	100	105	ms
t_{prog}	Programming Time		0.95	1	1.05	ms

Fig 3. LPC800 flash specification

3. IAP implementation in user code

3.1 IAP parameter passing

In an application, the IAP routines should be called with a word pointer in register r0 pointing to memory (RAM) which contains the command code and parameters. The result of the IAP command is returned in the ‘result table’. Register r1 contains a pointer to the result table. The IAP routine resides at address 0x1FFF 1FF0 and it is thumb code.

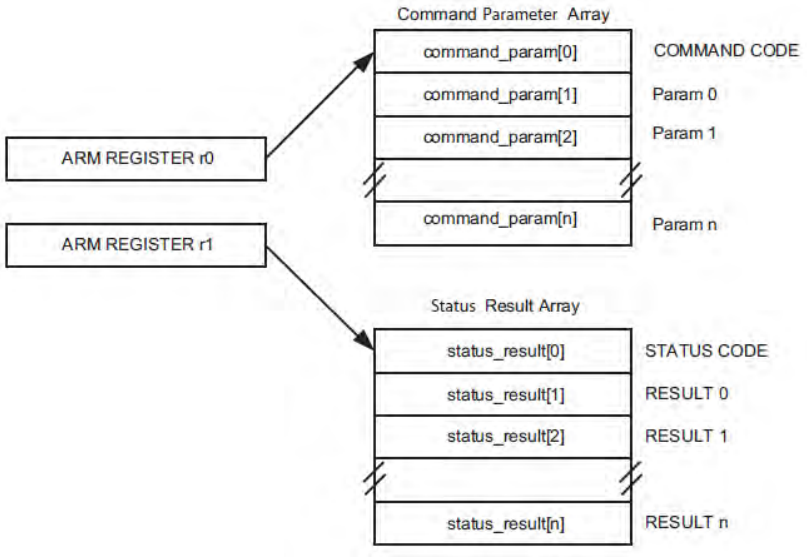


Fig 4. IAP parameter passing

3.2 IAP command summary

There are 10 IAP commands which are listed below. For example, Erase sector command is used to perform an erase operation on the entire sector.

IAP Command	Command Code
Prepare sector(s) for write operation	50 (decimal)
Copy RAM to flash	51 (decimal)
Erase sector(s)	52 (decimal)
Blank check sector(s)	53 (decimal)
Read Part ID	54 (decimal)
Read Boot code version	55 (decimal)
Compare	56 (decimal)
Reinvoke ISP	57 (decimal)
Read UID	58 (decimal)
Erase page	59 (decimal)

Fig 5. IAP command summary

3.3 IAP implementation in user code

3.3.1 Locating the IAP code in Boot ROM

The IAP routines are located in the Boot ROM. To access the IAP routines, the entry point of the IAP must be defined. For the LPC800, the address is 0x1FFF1FF1.

```
1    #define IAP_LOCATION 0x1FFF1FF1
```

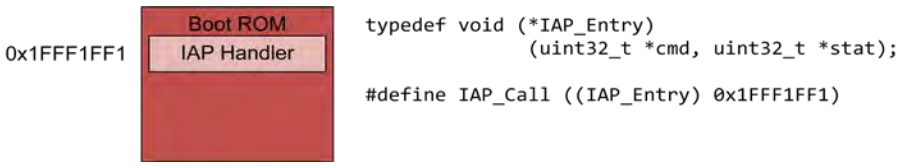


Fig 6. Locating the IAP code in Boot ROM

3.3.2 IAP parameters

The IAP routines take two unsigned 32-bit integer arrays, the command_param and status_result, as input. The command_param is a 5 element array and the status_result is a 4 element array. The arrays can be defined as:

```
2    unsigned int command_param[5];
3    unsigned int status_result[4];
```

or they can be defined as:

```
4    unsigned int * command_param;
5    unsigned int * status_result;
6    command_param = (unsigned int *) <address>
7    status_result = (unsigned int *) <address>
```

The demo code uses a data structure to define the IAP parameters as shown in Fig 7.

To make a call to the IAP routines, the following code can be used:

```
8    Iap_entry (command_param, status_result);
```

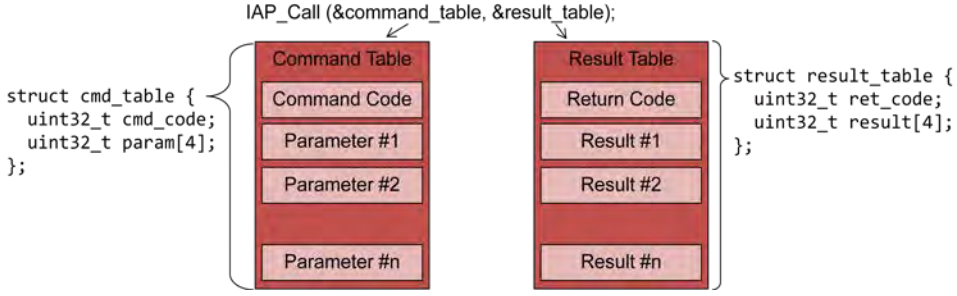
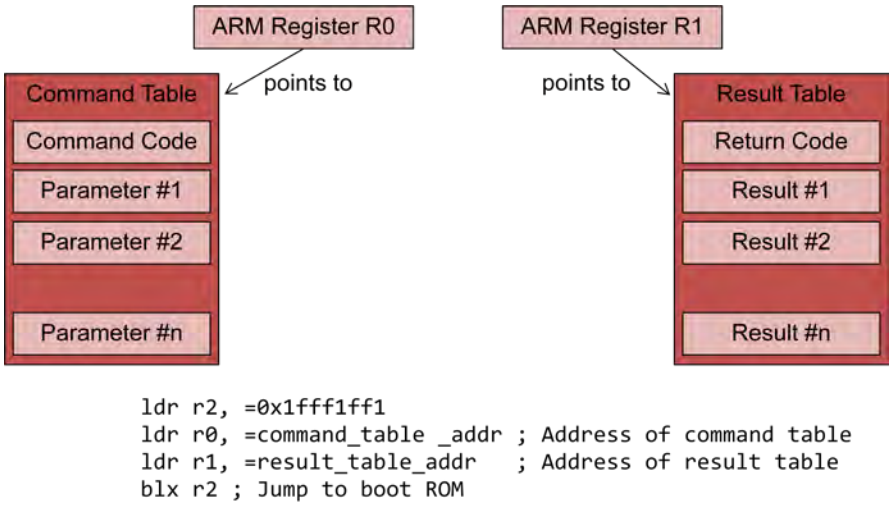


Fig 7. IAP parameters as a structure

3.3.3 Calling IAP in assembly code

The IAP routine can also be called in assembly language as shown in Fig 8.




```
ldr r2, =0x1fff1ff1
ldr r0, =command_table _addr ; Address of command table
ldr r1, =result_table_addr   ; Address of result table
blx r2 ; Jump to boot ROM
```

Fig 8. Calling IAP in assembly code

3.4 IAP routines

The list of IAP routines is listed below.

IAP Command	Code (base 10)	Functional description	Precautions
Prepare sector(s) for write operation	50	Turns off the write protection for the specified flash sectors.	This function must be called prior to executing “Copy RAM to Flash” or “Erase Sector(s)” commands.
Copy RAM to Flash	51	Performs a write operation from RAM to flash memory.	A flash sector must be prepared for write operation before contents can be written. Ensure no other flash accesses are performed during the copy procedure. Source data must be located in RAM.
Erase Sector(s)	52	Erases the contents of the entire flash sector(s). Erased flash sector(s) will read back with all bits set to 1’s.	A flash sector must be prepared for write operation before it can be erased. Ensure no other flash accesses are performed during the erase procedure.
Blank check sector(s)	53	Determines if flash sector(s) is (are) erased.	None
Read part identification number	54	Returns the identification number of a particular part. See the user manual for the specific part identification	None

IAP Command	Code (base 10)	Functional description	Precautions
		numbers.	
Read boot code version number	55	Returns the boot ROM version number.	None
Compare (memory)	56	Compares memory contents at two locations.	None
Re-invoke ISP	57	This function call will invoke the ISP routine located on the boot ROM.	Calling this function will remap the boot vectors, enable UART0 and Timer1 and change their PCLK values to CCLK/4.
Read device serial number	58	Returns the part's unique serial number.	None
Erase Page	59	Erases a page or multiple pages of on-chip flash memory.	The page has to be prepared for write operation before it can be erased. Ensure no other flash accesses are performed during the erase procedure.

3.4.1 Erase page feature

The erase page is a new feature introduced in LPC800. Instead of erasing each and every sector, this feature helps in erasing a single page (64 bytes) or multiple pages of the on-chip flash memory. In the demo code, page 160 has been erased.

3.5 IAP precautions

The IAP manipulates the memory during run-time. Therefore, certain precautions must be taken to ensure proper operations.

3.5.1 Interrupts

When the IAP routines are used any access to the flash memory must be avoided during the erase and write operations. If the vector interrupt table is located in the flash, all the interrupts must be disabled prior to erase and write.

The LPC800 also has the ability to remap the interrupt vector table to the RAM by changing the MAP bits in the SYSMEMREMAP register. This allows interrupts to occur even during the erase and write operations. But as the flash cannot be accessed during this time, the interrupt handlers must be executed from the RAM. Hence, all code related to the interrupt handlers must be copied from Flash into the RAM.

3.5.2 RAM usage

The IAP routines utilize 32 bytes of space in the top portion of the on-chip RAM for execution and up to 128 bytes of stack space. The user program should not use this space if IAP flash programming is permitted in the application. Furthermore, if the interrupt vector table is remapped to the SRAM, the bottom 512 bytes of the memory map should not be used.

4. Software setup

4.1 Interrupt remapping

The system remap register SYSMEMREMAP on NXP's LPC 8xx MCU families selects whether the exception vectors are read from boot ROM, flash or SRAM. By default, the flash memory is mapped to the address 0x0000 0000. When the MAP bits in the SYSMEMREMAP register are set to 0x0 or 0x1, the boot ROM or RAM respectively are mapped to the bottom 512 bytes of the memory map (addresses 0x0000 0000 to 0x0000 0200).

Bit	Symbol	Value	Description	Reset value
1:0	MAP		System memory remap. Value 0x3 is reserved.	0x2
		0x0	Boot Loader Mode. Interrupt vectors are re-mapped to Boot ROM.	
		0x1	User RAM Mode. Interrupt vectors are re-mapped to Static RAM.	
		0x2	User Flash Mode. Interrupt vectors are not re-mapped and reside in Flash.	
31:2	-	-	Reserved	-

Fig 9. SYSMEMREMAP register

So for interrupt handling during IAP on these MCU families, user code should copy the interrupt vector table from 0x0000 0000 to 0x1000 0000 and then set the MAP bits to be 0x1 to select the exception vector from RAM. The entire lower 512 byte flash block should be copied to RAM.

In addition to the SYSMEMREMAP register, the LPC8xx family implements a Vector Table Offset Register (VTOR) which provides more flexibility than the SYSMEMREMAP register. The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. On system reset, the vector table is located at address 0x0000 0000. The VTOR register allows the interrupt vector table to be relocated at an address other than the default.

4.2 SRAM memory mapping

The demonstration code relocates the interrupt vector to SRAM and uses the IAP code. This means that the compiler must be configured such that the bottom 512 bytes and the top 32 bytes of the memory cannot be touched. In the Keil environment, the IRAM1 section should be specified to be smaller than the actual SRAM size to prevent the compiler from using these areas.

The SRAM starts at address 0x1000 0000. Since the interrupt vector table uses 512 bytes of the bottom of the SRAM, the start location is now set to 0x1000 0200. The total SRAM size of LPC800 is 4 kB. With IAP using 32 bytes in the top of the SRAM, this means the usable SRAM size is 4 kB – 32 bytes i.e. 4064 bytes. But since 512 bytes is also being used by the interrupt vector table, the SRAM size now becomes:
(4096 – 32 – 512) bytes = 3552 bytes = 0x0DE0.

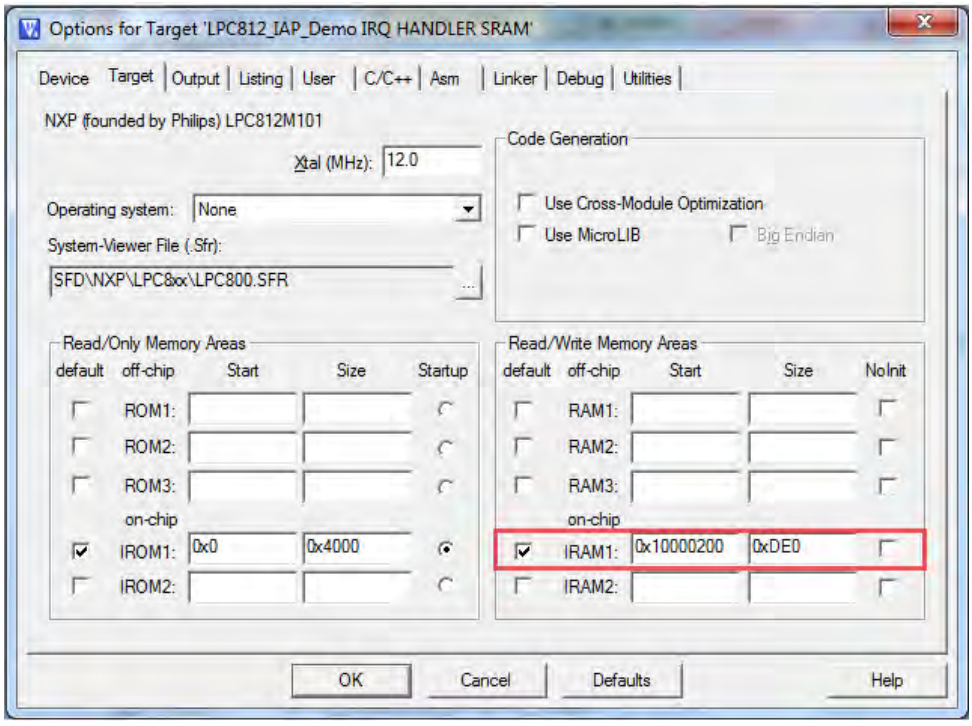


Fig 10. Keil IROM and SRAM remapping

In the LPCXpresso IDE, the same task is accomplished by changing the MCU settings.

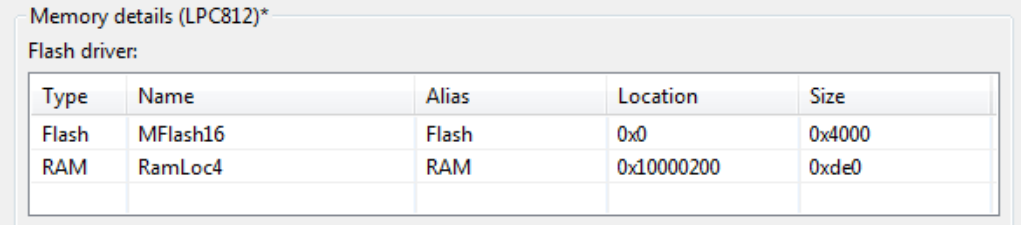


Fig 11. LPCXpresso flash and SRAM remapping

In IAR Embedded Workbench, the SRAM remapping is achieved by changing the linker configuration file settings. The RAM start address is set to 0x1000 0200 and the end address to 0x1000 0FE0 (0x1000 0200 + 0xDE0).

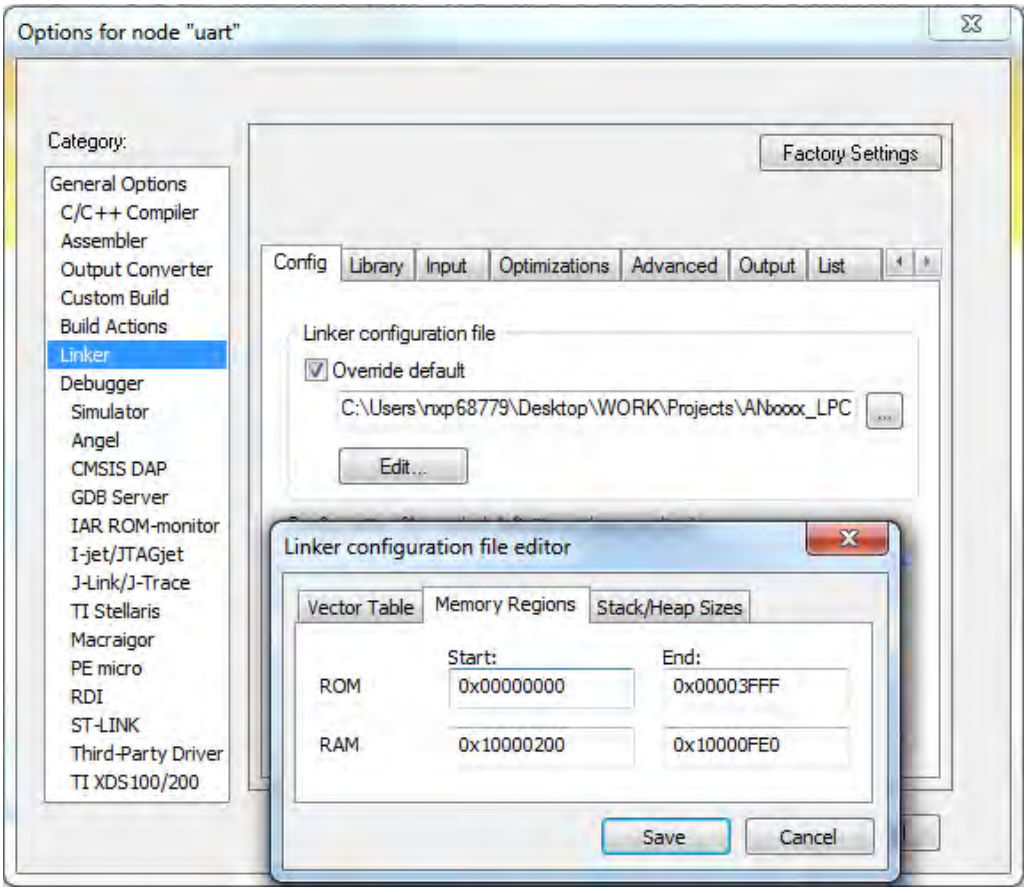


Fig 12. IAR Embedded Workbench ROM and SRAM remapping

4.3 Systick interrupt

The systick is used to create a periodic interrupt while the software is running. On each interrupt, PIO0_16 is toggled. Since during the IAP call the flash is not accessible to the software, the SysTick interrupt handler is relocated to the SRAM.

The above operation is done within the Keil environment as opposed to manually editing the scatter file.

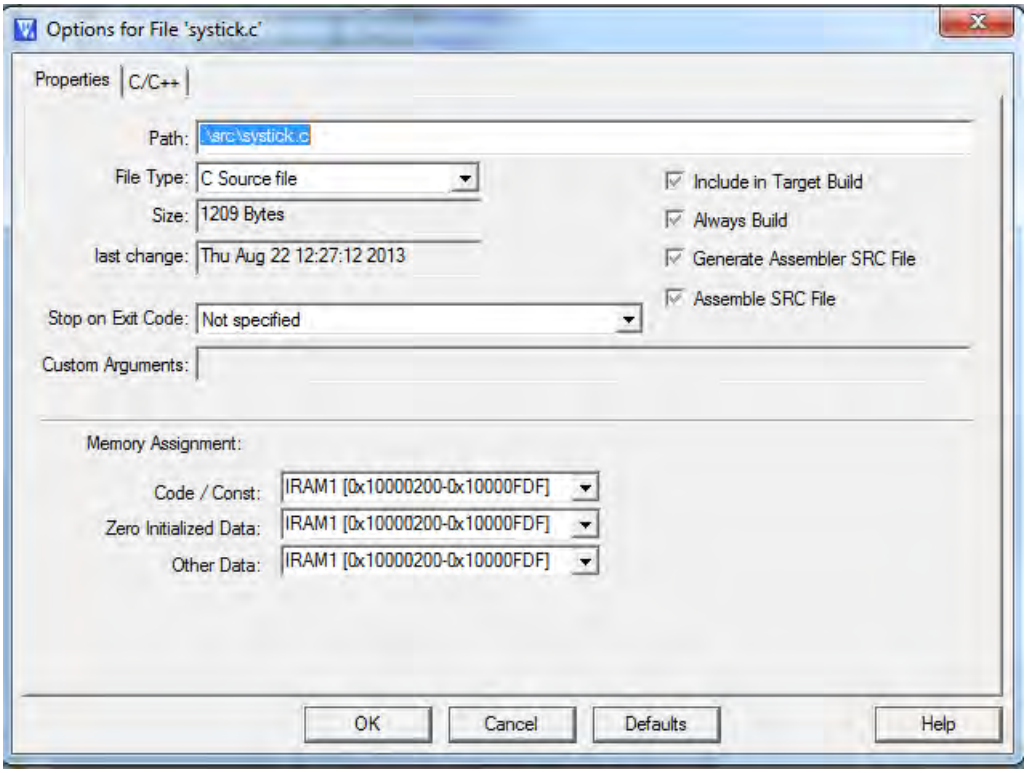


Fig 13. SysTick interrupt handler settings in Keil

For the LPCXpresso IDE, the systick handler function is directed to by placed into the SRAM by the using the **.data.ramfunc** directive.

```
9     __attribute__ ((__section__(".data.ramfunc")))
10    void SysTick_Handler(void){
11          LPC_GPIO_PORT->NOT0 = (1<<7);}
```

In the IAR Embedded Workbench, the systick handler function is placed into the SRAM by using the compiler directive **__ramfunc.**

```
12    __ramfunc void SysTick_Handler(void){
13          LPC_GPIO_PORT->NOT0 = (1<<7);}
```

5. Handling interrupts during IAP

The LPC800's flash is not accessible when IAP routines are being called. This can be dealt with by using two methods.

5.1 Method 1: Disable interrupts before calling IAP functions

All interrupts are disabled prior to any IAP operations.

```
__disable_irq();
iap_call(&cmd_table, &result_table);
__enable_irq();
```

Fig 14. Method 1: Disable interrupts

5.2 Method 2: Relocate the interrupt table to SRAM

The interrupt vector table is moved to the SRAM and the VTOR register is set to the starting address of the remapped vector table. The MAP bits in the SYSMEMREMAP register is set to 0x1, indicating the vector table is located in the SRAM, not in the flash area.

```
#if defined IRQ_HANDLER_IN_SRAM
  CopyInterruptToSRAM();    //remap interrupt vector to SRAM
  SCB->VTOR =0x10000000;
  LPC_SYSCON->SYSMEMREMAP = 0x01; //change memory map
#endif
```

Fig 15. Method 2: Relocate the interrupt table to SRAM

The interrupt vector table is copied to SRAM using the function call 'CopyInterruptToSRAM() '. The function call is hardcoded to copy from flash address 0x00 to SRAM address of 0x1000 0000.

```
void CopyInterruptToSRAM(void)
{
  unsigned int * flashPtr, * ramPtr;
  unsigned int * uLimit = (unsigned int *) 0x200;

    ramPtr = (unsigned int *)0x10000000;  //load RAM starting at 0x10000000,
    flashPtr = (unsigned int *)0x00;      //start of interrupt vector table
    while(flashPtr < uLimit)
      {
        *ramPtr = *flashPtr;
        ramPtr++;
        flashPtr++;
      }
}
```

Fig 16. Copy the IRQ handler to SRAM

6. Design tips for IAP

6.1 Flash lookup sheet

Fig 17 shows the flash lookup sheet for the IAP command usage.

Operations	IAP Commands
Erase page (64-bytes)	Prepare sector for write (**CMD = 50**) Erase page (**CMD = 59**)
Erase sector (1024-bytes)	Prepare sector for write (**CMD = 50**) Erase sector (**CMD = 52**)
Write page / sector (64, 128, 256, 512, 1024 bytes)	Prepare sector for write (**CMD = 50**) Copy RAM to Flash (**CMD = 51**)
Verify data	Compare (**CMD = 56**)

Read Flash data	Read like ordinary memory. E.g.:
	`#define STORAGE_ADDR 0x1000`
	`uint32_t stored_value =`
	`* ((uint32_t *) STORAGE_ADDR);`

Check if Flash is empty	Blank check sector (**CMD = 53**)

Fig 17. Flash lookup sheet

The above figure shows the commands to be used to perform various operations. For example, in order to erase a sector, the sectors should be prepared for a write operation by the use of 'Prepare sector for write' command and the page is erased using the 'Erase sector' command. Every Erase and Write command must be preceded by a Prepare command.

6.2 Set correct CCLK parameter

The input parameter 'CCLK' of an IAP command should be equal to the CPU clock frequency in kilohertz (kHz). If the CCLK parameter is less than the CPU clock, the flash operation may be unstable. If the CCLK parameter is higher than the CPU clock, the flash operation may be slower than expected. If the CCLK is not equal to the CPU clock, the flash reliability cannot be guaranteed.

7. Conclusion

This application note provides example implementation for In-Application Programming (IAP) in LPC800 MCU families. The IAP routines available on the LPC800 provide an easy and simple way for data storage or for program updates. As these routines are stored on the on-chip ROM, the user application's code size is minimized.

For additional details on how the IAP routines operation, refer to the LPC800 user manual.

8. Application Example

The demo software sends a menu system to the PC through the UART at 9600, 8, N, 1. The IAP operations were performed on sector 10 and page 160.

The demo software demonstrates the IAP calls of:

- Reading the part ID of the device
- Reading the Boot version
- Reading the device serial number
- Preparation of a sector
- Erase page
- Write on a page.

The following figure shows the UART menu display using Tera Term on the PC.

Fig 18. UART window on PC

9. Legal information

9.1 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.

9.2 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.

9.3 Trademarks

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

10. List of figures

11. Contents

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

Date of release: 24 September 2013
Document identifier: AN11388