AN10968

Using Code Read Protection in LPC1100 and LPC1300

Rev. 1 — 19 August 2010 Application note

Document information

Info	Content
Keywords	LPC11xx, LPC13xx, M0, CRP, ISP, LPCXpresso
Abstract	A comparison of CRP levels supported by the LPC1100/LPC1300 part families, as well as an overview of using CRP in LPCXpresso

Revision history

Rev	Date	Description
1	20100819	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. Introduction

Code Read Protection is a mechanism that allows users to enable different levels of security in the system so they can protect both their software code and hardware.

LPC1100/LPC1300 devices have three different active security levels: CRP1, CRP2, and CRP3. Each mode increases the security level, with CRP3 restricting any access to the device. These devices also feature a new protection level “NO_ISP” which will suppress response to the ISP pin at system startup (but will not prevent debug via SWD, nor will it prevent read access to flash memory). In this application note we examine all these security levels and how to use them according to various security requirements. We also provide an example to test all of these modes. This application note will make use of the LPCXpresso1114 or LPCXpresso1343 Evaluation boards and the LPCXpresso Baseboard. The software will be developed using the LPCXpresso IDE.

CAUTION: Although the example used in this application note was carefully tested, it is recommended that the user initially configure the device to a level lower than CRP3. Once the code is successfully tested, CRP3 can be used with confidence.

2. Flash memory access methods

In general the LPC1100/LPC1300 flash memory can be accessed in two different ways:

- Using the SWD flash programming interface: This is the method that Debug tools use in order to download code into the device and start/stop execution.
- Using In-System Programming (ISP): This method is provided by the boot loader using UART0 serial port (or USB for LPC1300).

3. Understanding CRP security levels

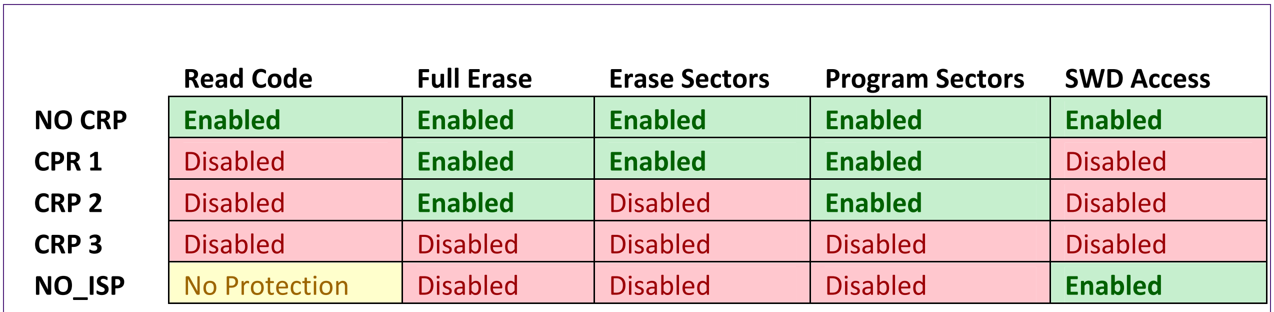
As the name implies, Code Read Protection (CRP) provides a method for users to protect their code from being read from the device flash. In this way, designers can prevent unauthorized users from obtaining their object code which could be disassembled or downloaded onto another hardware platform. Please be aware that all changes to CRP require a power cycle to take place before becoming active.

Notice that from Fig 1 that all active CRP levels will disable Serial-Wire-Debug. By disabling debug access, memory contents cannot be read (or written) by an unauthorized party using widely available debugging equipment. A consequence of this, however, is that should CRP be accidentally enabled during development, debugging cannot take place until the part has been erased via ISP and programmed with code which does not enable CRP. Active CRP levels disable memory read back when LPC1100 and LPC1300 devices are booted into ISP mode as well. CRP settings have no effect on IAP commands (which are executed programmatically by application code).

CRP Level 1 is designed to prevent read access to the device while enabling modification of flash memory on a sector by sector basis. Note that sector 0 can only be erased when fully erasing the device, and cannot be written to when CRP1 is enabled. This effectively prevents modification to sector 0. CRP1 is required, for example, when a design dedicates one or more sectors of flash to storing calibration information or serial numbers, etc. If a full chip erase were to take place during a firmware update, this calibration information would be lost. CRP1 can also be useful, for example, should a design implement a custom secured bootloader. This would program sector 0 and only allow field updates to modify the “application” sectors while enabling the bootloader to persist in an unmodified state.

Though it is unlikely, it is conceivable that an attacker with knowledge about a system could partially overwrite firmware in such a way as to gain read access to internal flash memory. To prevent this, CRP Level 2 further increases security by only supporting full chip erasure. This ensures that devices are entirely blank prior to updates, and therefore not susceptible to modification by an attacker.

CRP Level 3 is the highest level supported by the LPC1100/LPC1300 devices. In sophisticated reverse engineering, an intellectual property thief may be able to learn proprietary information about a design by fully erasing and then programming the target device with custom test code and analyzing how the PCB behaves. This would then enable the thief to implement a counterfeit design without ever having read original object code. CRP Level 3 effectively disables ISP functionality[1] and SWD, thus the device’s flash memory can no longer be modified. Should there be instances where hardware cannot be allowed to run unauthorized code, CRP Level 3 should be used. **Be aware that there is no built-in recovery for designs once CRP Level 3 is enabled.** The most simplistic way of implementing a custom recovery mechanism involves the “Re-Invoke ISP” IAP call.



	Read Code	Full Erase	Erase Sectors	Program Sectors	SWD Access
NO CRP	Enabled	Enabled	Enabled	Enabled	Enabled
CPR 1	Disabled	Enabled	Enabled	Enabled	Disabled
CRP 2	Disabled	Enabled	Disabled	Enabled	Disabled
CRP 3	Disabled	Disabled	Disabled	Disabled	Disabled
NO_ISP	No Protection	Disabled	Disabled	Disabled	Enabled

Note that Sector 0 is not erasable when CRP1 is enabled unless a full device erase takes place

Fig 1. Comparison of CRP levels

4. Secondary/encrypted bootloaders

If a designer is going to take the additional effort to protect their devices from design theft, dealing with field updates should be taken into consideration. If an application will enable CRP, it is usually a good practice to develop a custom encrypted bootloader to allow field updates. Distributing unencrypted object code for the updating process defeats the notion of protecting the design.

Designs making use of the USB features of the LPC1300 part families may also consider implementing an encrypted USB based secondary bootloader, as is typically done in DFU applications.

5. NO_ISP

The LPC1100/LPC1300 families of parts feature a new CRP mode, “NO_ISP”. This will prevent the on chip bootloader from sampling the ISP pin at start up. There are cases in which external devices may drive a signal on the ISP pin of a device. In the event of a power failure that causes a brown out event, the LPC1100/LPC1300 device may restart

[1] Designs with CRP3 can allow reprogramming by implementing a custom bootloader which uses IAP commands to modify flash memory contents.

and find that the ISP pin is erroneously being asserted. To prevent the possibility of an accidental invocation of ISP, designers can enable the “NO_ISP” mode. Be aware that NO_ISP does not prevent theft of intellectual property, as debugger access remains enabled, and flash contents can be read back via this mechanism.

6. Example: Using CRP with LPCXpresso

LPCXpresso is distributed with projects for both LPC1100 and LPC1300 families which make use of CRP. Designers who anticipate using CRP in their applications are encouraged to use these example projects as starting points for their designs.

When using UART based ISP with the LPC1300 and FlashMagic, refer to the LPCXpresso Base Board’s User Manual for details about proper jumper configuration. With all jumpers in the default locations, the LPC1300 evaluation board will operate in USB based ISP mode.

The LPC1100 is able to operate with FlashMagic with all jumpers in their default locations.

Controlling CRP in LPCXpresso projects is done in two parts:

- Defining a constant which sets the appropriate CRP level
- Configuring the linker with custom scripts to place this constant at the CRP memory address (0x2FC)

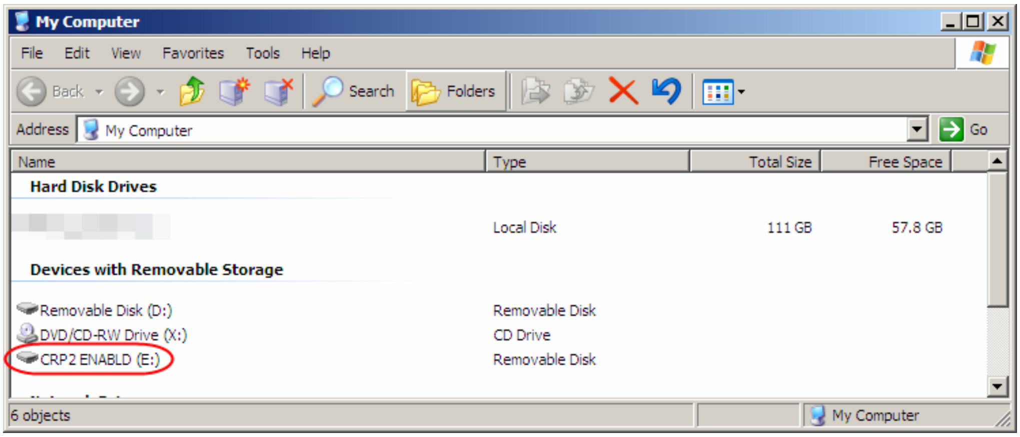
Fig 2 shows how a developer can go about defining a constant which is assigned with their desired CRP level. The attribute directive ensures that the constant (in this example *CRP_WORD*) is placed in a special “.crp” section of memory by the linker.

```
#define NO_CRP          0xFFFFFFFF
…
#define NO_ISP_MAGIC    0x4E697370
…
#define CRP1_MAGIC      0x12345678
…
#define CRP2_MAGIC      0x87654321
…
/**** DANGER CRP3 WILL LOCK PART TO ALL READS and WRITES ****/
/*********** #define CRP3_MAGIC xxxx 0x43218765 *************/
…
#define CURRENT_CRP_SETTING NO_CRP
…
__attribute__ ((section(".crp"))) const uint32_t CRP_WORD =
CURRENT_CRP_SETTING;
…
```

Fig 2. Excerpt from CRP example (C source code)

The second part of the two step process involves configuring the linker to place the constant at the proper memory location. This is done via custom linker scripts. The process for manually configuring the linker in LPCXpresso is outlined in Appendix A.

The major difference between the custom linker script and the automatically generated one can be seen in Fig 5. Notice that the “.crp” section is defined in the script, and begins at address 0x2FC, the CRP memory address for LPC1100/LPC1300 devices.



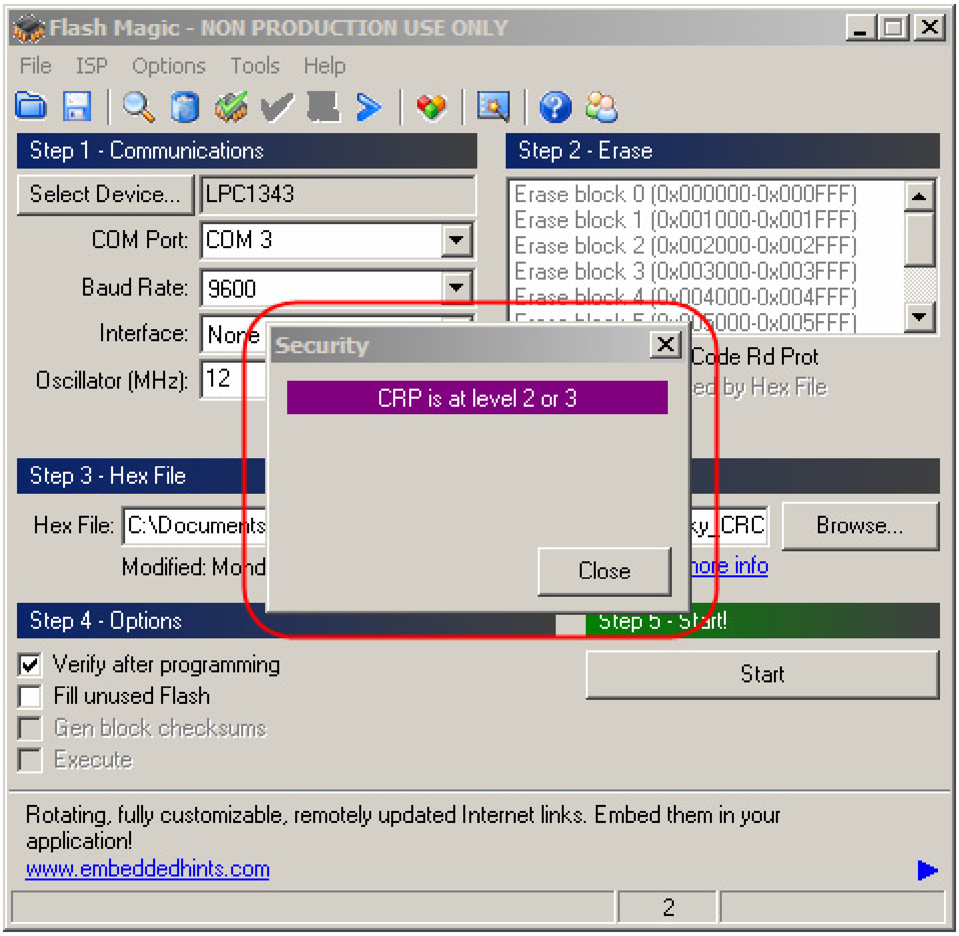
Address: My Computer

Name	Type	Total Size	Free Space
Hard Disk Drives			
[redacted]	Local Disk	111 GB	57.8 GB
Devices with Removable Storage			
Removable Disk (D:)	Removable Disk		
DVD/CD-RW Drive (X:)	CD Drive		
CRP2 ENABLD (E:)	Removable Disk		

6 objects

My Computer

Fig 3. LPC1343 with CRP2 enabled (USB based ISP mode)



Select Device... LPC1343

COM Port: COM 3

Baud Rate: 9600

Interface: None

Oscillator (MHz): 12

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

Code Rd Prot

ed by Hex File

CRP is at level 2 or 3

Hex File: C:\Documents ky_CRC

Modified: Mond

☑ Verify after programming
☐ Fill unused Flash
☐ Gen block checksums
☐ Execute

Rotating, fully customizable, remotely updated Internet links. Embed them in your application!
www.embeddedhints.com

2

Fig 4. LPC1343 with CRP2 enabled (UART based ISP and FlashMagic)

```
…
ENTRY(ResetISR)

SECTIONS
{
   .text :
   {
     KEEP(*(.isr_vector))
     . = 0x000002FC;
     KEEP(*(.crp))
     *(.text*)
     *(.rodata*)
    } > MFlash32
…
```

(1) Bold text indicates changes from automatically generated linker script

Fig 5. Excerpt from CRP example (custom linker script)

7. Conclusion

By learning how to enable CRP, designers can prevent the theft of their intellectual property. This application note has detailed the differences between each of the CRP levels, and has shed light onto some of the conditions that might lead a developer to choose one level rather than another. Details of potential pitfalls that can occur when using CRP have been provided in the hopes that developers might avoid them. When using the LPCXpresso IDE developers can add CRP to their designs by creating a special segment of memory, and placing a constant in it.

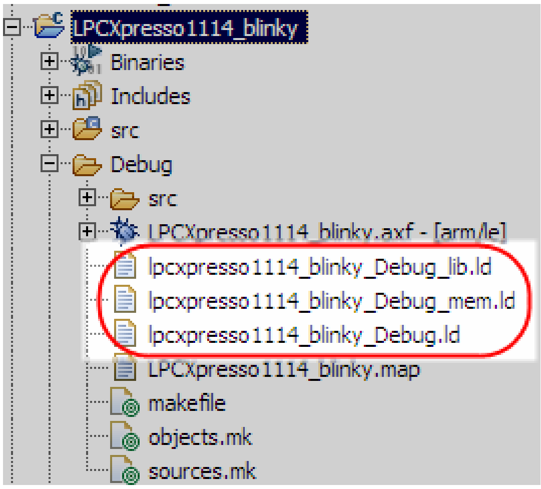
8. Appendix A

Several steps are required to specify a custom linker script[2] set. First make a new folder in the project named “custom_ld”. This will store the customized script files. Next copy the automatically generated linker scripts with “.ld” extensions from the active target folder (typically “Debug” or “Release”) into the “custom_ld” folder. Fig 6 illustrates where the automatically generated scripts are stored in the project. Rename the file as per Table 1. Before you modify the content of the scripts, you need to ensure that the toolchain is configured to use them as seen in Fig 7. Add the “.crp” section to Custom.ld as show in Fig 5. Finally update Custom.ld to point to the Custom_mem.ld and Custom_lib.ld scripts in the newly created custom_ld folder. An example of this is shown in Fig 8.

Table 1. Renamed linker scripts

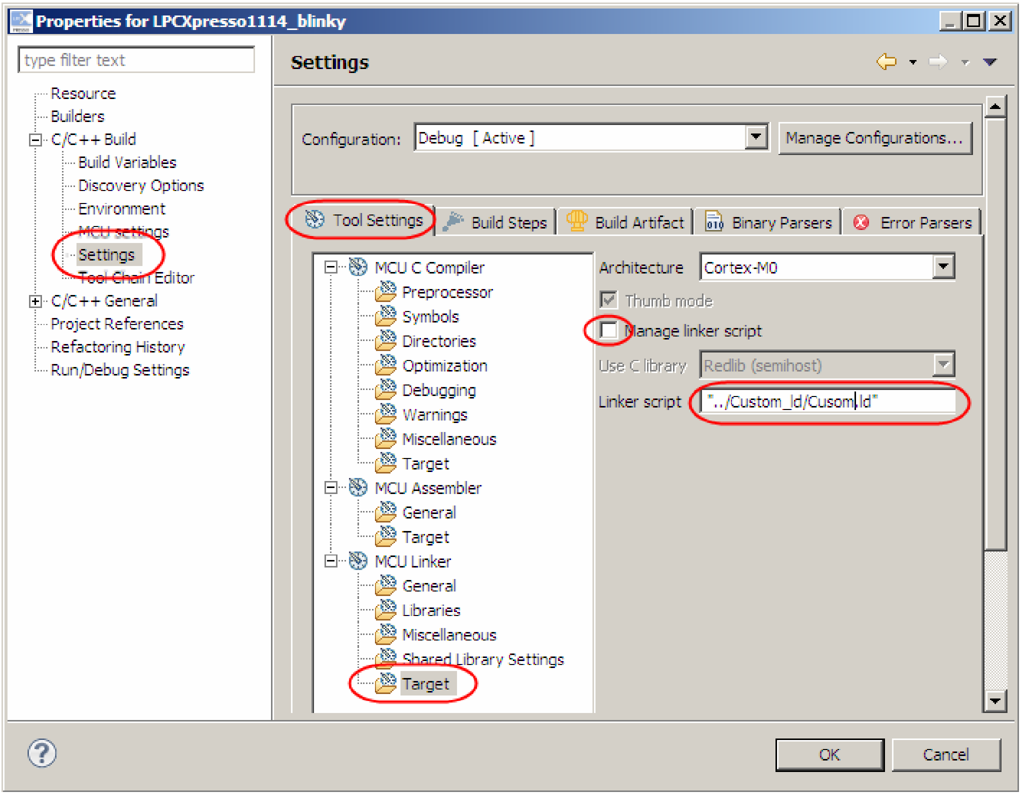
Original name	Renamed copy
*PROJECTNAME*_Debug.ld	Custom.ld
*PROJECTNAME*_Debug_mem.ld	Custom_mem.ld
*PROJECTNAME*_Debug_lib.ld	Custom_lib.ld

[2] http://lpcxpresso.code-red-tech.com/LPCXpresso/node/31



LPCXpresso1114_blinky
- Binaries
- Includes
- src
- Debug
  - src
  - LPCXpresso1114_blinky.axf - [arm/le]
  - lpcxpresso1114_blinky_Debug_lib.ld
  - lpcxpresso1114_blinky_Debug_mem.ld
  - lpcxpresso1114_blinky_Debug.ld
  - LPCXpresso1114_blinky.map
  - makefile
  - objects.mk
  - sources.mk

Fig 6. Automatically generated linker scripts



Configuration: Debug [Active]

Architecture: Cortex-M0

☑ Thumb mode

☐ Manage linker script

Use C library: Redlib (semihost)

Linker script: "../Custom_ld/Cusom.ld"

Fig 7. Specifying custom linker file

```
/*
 * GENERATED FILE - DO NOT EDIT
 * (C) Code Red Technologies Ltd,
 * Generated C linker script file for LPC1114
*/
INCLUDE "../custom_ld / Custom_lib.ld "
INCLUDE "../custom_ld /Custom_mem.ld "

ENTRY(ResetISR) …
```

(1) Underline indicates changes from original file

Fig 8. Modifications to “Custom.ld” linker script

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.

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 accepts 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 national authorities.

9.3 Trademarks

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

10. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

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

Date of release: 19 August 2010

Document identifier: AN10968