



This thread has been locked.  
If you have a related question, please click the "[Ask a related question](#)" button in the top right corner. The newly created question will be automatically linked to this question.

[Go to Wireless connectivity](#)

[Forums](#)

## What is the address range of CC1310 memory ?

Gilbert

Expert 1070 points  
 Community Member

Resolved

Replies: 13

Views: 7533

Hello, everyone!

Where can I find the details of [CC1310](#) memory? I want to know the address range of [CC1310](#) info page, ROM and flash.

Helic

CC1310 memory info page CC1310

Jun 6, 2016 3:08 AM

Gilbert

Expert 1070 points  
 Community Member

Locked

All Responses Suggested Answers

Jun 9, 2016 3:42 PM

Richard W.

Genius 14860 points  
 Texas Instruments

Hi,

please find a [memory map](#) in the CC13XXWare documentation.

Best regards

Richard

Locked

Jun 23, 2016 2:34 AM

Gilbert

Expert 1070 points  
 Community Member

In reply to Richard W.:

Hi, Richard W.

I find that the info page is FCFG1, and there are some byte used by RTOS. Every byte in FCFG2 is 0xFF, so I think that I can save my config info into FCFG2. It is right?

Best Regards,  
Gilbert

Locked

Jun 23, 2016 7:16 AM

Richard W.

Genius 14860 points  
 Texas Instruments

In reply to Gilbert:

Hi Gilbert,

The FCFG1/2 sections contain factory settings are not writeable. The CCFG section contains the application configuration and maps to the last page in flash. You can use this section to store additional configuration settings, but you need to use the flash address. I recommend using the flash section before CCFG in order to not delete the CCFG by accident. Here is a more application-centric memory map:

1	Segment	Begin	End	Size	Description
2	Flash	0x0000 0000	0x0000 0FFF	4 KiB	Free for application. 0x0000 is used as
3	(32/128 KiB)				
4		0x0000 1000	0x0000 14FF	1280 B	When using TI-RTOS in ROM, this section
5					at fixed addresses.
6					
7					
8					
9		0x0000 1500	0x0001 FFA7	1280 B	Free for application
10					
11		0x0001 FFAB	0x0001 FFFF	88 B	Customer configuration (CCFG) area. It c
12					ontains parameters for the ROM bootcode, device
13					Please refer to the Device Configuration
14					Technical Reference Manual.
15					
16	ROM	0x1000 0000	0x1001 FFFF	128 KiB	Pre-built functions from TI-RTOS kernel
17	(128 KiB)				size of applications. See "Rom Functions
18					:ref: using_internal_rom".
19					
20					
21	GPRAM	0x1100 0000	0x1100 1FFF	8 KiB	Used as internal instruction cache by de
22					application purpose. Please refer to the
23					Technical Reference Manual.
24					
25	SRAM	0x2000 0000	0x2004 FFFF	20 KiB	Free for application data.
26					
27	RF_CORE_RAM	0x2100 0000	0x2100 FFFF	4 KiB	Internal RF Core RAM. Not accessible from
28					
29	Peripheral	0x4000 0000	0xFFFF FFFF		Internal registers of peripheral units. I
30	and internal				application. Please refer to "CPU Domain
31	registers				
32					

Best regards

Richard

Locked

Jun 24, 2016 3:45 AM

Gilbert

Expert 1070 points  
 Community Member

In reply to Richard W.:

Hi, Richard W.

I have some questions about CCFG:

1. The size of CCFG is 4KB. Why is the size of the Customer configuration (CCFG) area in flash 88B? Should I use other 4KB-88B?
2. In "SmartRF Flash Programmer2 ver.1.7.2 --> Edit --> CCFG", I change the value in address 0x0001ffb8 from 0xFF to 0x01. After reload new code in CCS, the value in address 0x0001ffb8 has been 0xFF. I hope to know how to do in order to make sure that the value in CCFG won't be changed.

Best Regards,

Gilbert

Locked

Jun 24, 2016 8:47 AM

Richard W.

Genius 14860 points  
 Texas Instruments

In reply to Gilbert:

Hi Gilbert,

1. The flash section size (also referred to as page size/ segment size) on the [CC1310](#) is 4KiB. This is only important when erasing, because only whole sections can be deleted on a flash. The CCFG is located at the end of the last section and uses 88 bytes that are memory-mapped to internal registers.
2. When you perform a chip erase, then the whole flash is wiped out, including the CCFG. The default value after erase is 1 for a cell, hence 0xFF for every byte. If you don't erase that section, the content remains. In order to keep the content of this section:
  - a. Uncheck the "Erase" action in Flash programmer
  - b. Do not build the ccfg.c file with your TI-RTOS application, but build it separately and only flash it once. be warned that you can only flip bits from 1 to 0 without erasing. When you erase the section where the CCFG remains, you must immediately restore the content. If something bad happens during that procedure, then your application won't boot after the next reset.

Best regards

Richard

Locked

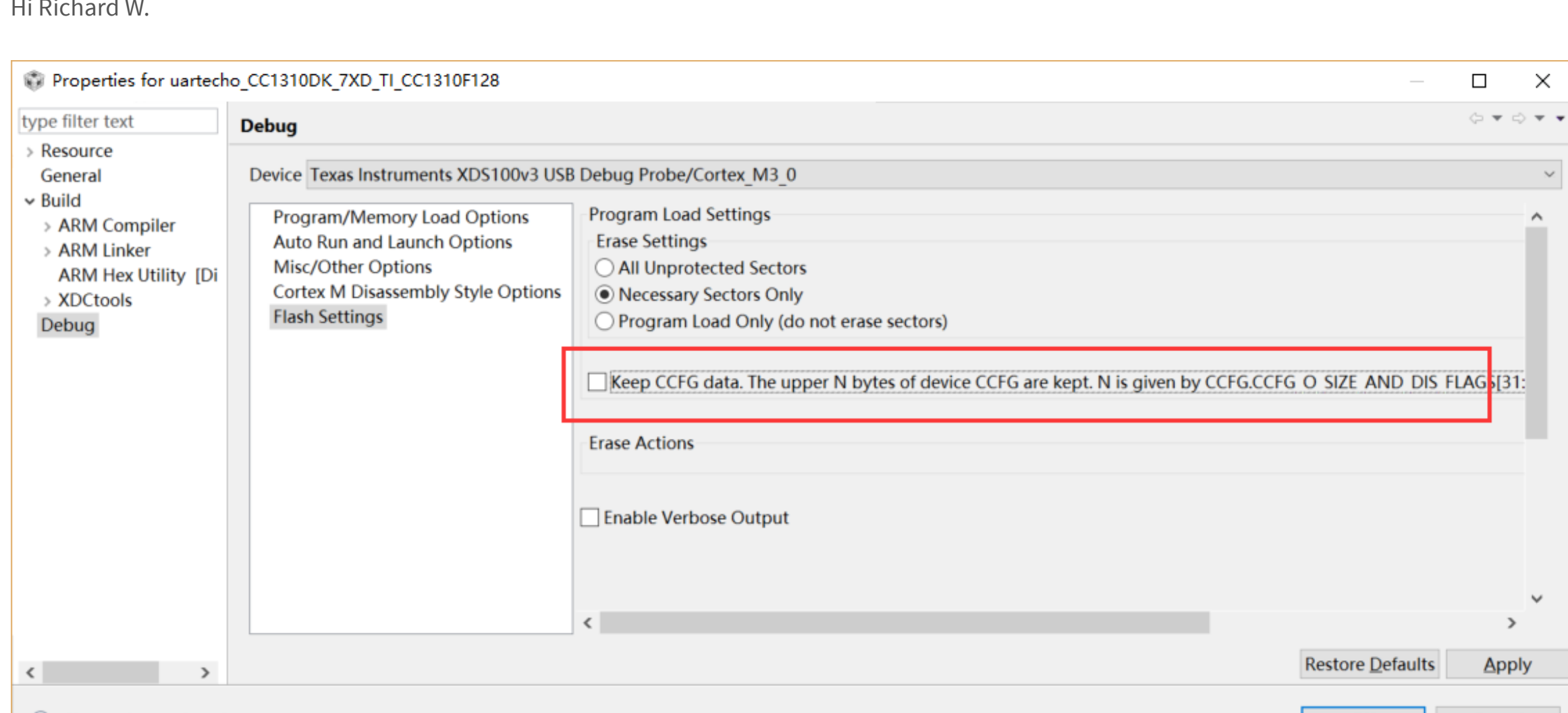
Jun 26, 2016 1:09 PM

Gilbert

Expert 1070 points  
 Community Member

In reply to Richard W.:

Hi Richard W.



Selecting this can help me make sure the content in CCFG won't be changed?

Best Regards,

Gilbert

Locked

Jun 27, 2016 6:52 AM

Richard W.

Genius 14860 points  
 Texas Instruments

In reply to Gilbert:

This makes sure that the debugger won't overwrite the .ccfg section when it uploads an application even though it is included in the application image. If you don't build the ccfg.c file in your project, then this option is not necessary because a ccfg section is not included. However, TI-RTOS requires this section to be present when it starts. So you must flash it at least once. After that you can reflash your device as many times as you want as long as the .ccfg section is not deleted.

I don't know about the internal ROM bootloader, maybe it requires some of the bytes in the .ccfg, too.

In my previous post, I talked about [FlashProgrammer2](#) which is a separate programming tool.

Best regards

Richard

Locked

Jun 29, 2016 8:09 AM

Gilbert

Expert 1070 points  
 Community Member

In reply to Richard W.:

Hi, Richard.

Another question.  
In my [CC1310](#) project, executing malloc(n) must failed when n is more than 1000. And I find that malloc is defined in cfg\_pem3.c file. I guess that it will execute successfully if calling malloc declared in stdlib.h file. But I don't know how to call the malloc declared in stdlib.h file.

Best Regards,  
Gilbert

Locked

Jul 22, 2017 9:00 AM

Omkar Inamdar

Intellectual 870 points  
 Community Member

In reply to Richard W.:

Hello Richard,

I am trying to write my data into the flash using following code :

```
1  if( FlashProtectionGet( FLASHMEM_BASE_ADDR ) == FLASH_NO_PROTECT)
2  {
3      if(FlashSectorErase(FLASHMEM_BASE_ADDR ) != FAPI_STATUS_SUCCESS)
4      {
5          while(1);
6      }
7      FlashProgram( (uint8_t *)pbuffer, FLASHMEM_BASE_ADDR, 16);
8  }
9  }
```

FLASHMEM\_BASE\_ADDR = 0xF00.

However I am not able to erase the flash and FlashSectorErase function is returning FAPI\_STATUS\_INCORRECT\_DATABUFFER\_LENGTH.  
Am I using wrong flash address to write the user data ?

Locked

Jul 24, 2017 7:13 AM

Richard W.

Genius 14860 points  
 Texas Instruments

In reply to Omkar Inamdar:

Hi,

Omkar Inamdar  
However I am not able to erase the flash and FlashSectorErase function is returning FAPI\_STATUS\_INCORRECT\_DATABUFFER\_LENGTH.  
Am I using wrong flash address to write the user data ?

The flash sector/page size is 4 KiB. Hence the address provided to FlashSectorErase() must be a multiple of 4 KiB (4 \* 1024 bytes), e.g. 0x0000, 0x1000, 0x2000, ...

Best regards

Richard

Locked



This thread has been locked.  
If you have a related question, please click the "[Ask a related question](#)" button in the top right corner. The newly created question will be automatically linked to this question.



TI E2E™ support forums

Other support forums

Educational resources

Other resources

Forum list

E2E China

TI training

WEBENCH® Power Designer

Site support

beagleboard.org

TI university program

Customer support center

Settings

TI technical articles

All content and materials on this site are provided "as is". TI and its respective suppliers and providers of content make no representations about the suitability of these materials for any purpose and disclaim all warranties and conditions with regard to these materials, including but not limited to all implied warranties and conditions of merchantability, fitness for a particular purpose, title and non-infringement of any third party intellectual property right. No license, either express or implied, by estoppel or otherwise, is granted by TI. Use of the information on this site may require a license from a third party, or a license from TI.

Content on this site may contain or be subject to specific guidelines or limitations on use. All postings and use of the content on this site are subject to the [Terms of use](#) of the site; third parties using this content agree to abide by any limitations or guidelines and to comply with the [Terms of use](#) of this site. TI, its suppliers and providers of content reserve the right to make corrections, deletions, modifications, enhancements, improvements and other changes to the content and materials, its products, programs and services at any time or to move or discontinue any content, products, programs, or services without notice.

TI worldwide | [Contact us](#) | [myTI login](#) | [Corporate citizenship](#)

Follow us [f](#) [t](#) [in](#) [in](#)

TI is a global semiconductor design and manufacturing company. Innovate with 100,000+ analog ICs and embedded processors, along with software, tools and the industry's largest sales/support staff.

© Copyright 1995-2020 Texas Instruments Incorporated. All rights reserved.  
[Trademarks](#) | [Privacy policy](#) | [Terms of use](#)