HDSC 华大半导体 HUADA SEMICONDUCTOR

32 Bit Microcontrollers

FLASH Operating Instructions and Precautions

Applicable objects

Series	Product Model	Series	Product Model	Series	Product Model
HC32L110	HC32L110C6UA HC32L110C6PA HC32L110C4UA HC32L110C4PA HC32L110B6PA HC32L110B4PA HC32L110B6YA	**HC32F00**	HC32F003C4UA HC32F003C4PA HC32F005C6UA HC32F005C6PA HC32F005D6UA	**HC32L13**	HC32L130E8PA HC32L130F8UA HC32L130J8TA HC32L136J8TA HC32L136K8TA
HC32F03	HC32F030E8PA HC32F030F8UA HC32F030F8TA HC32F030H8TA HC32F030J8TA HC32F030K8TA	**HC32L07**	hc32l072pata hc32l072kata hc32l072jata hc32l073pata hc32l073kata HC32L073JATA	**HC32F07**	hc32f072pata hc32f072kata hc32f072jata
HC32L17	HC32L176PATA HC32L176MATA HC32L176KATA HC32L176JATA HC32L170JATA HC32L170FAUA	**HC32F17**	hc32f176pata hc32f176mata hc32f176kata hc32f176jata hc32f170jata HC32F170FAUA	**HC32L19**	HC32L196PCTA HC32L196MCTA HC32L196KCTA HC32L196JCTA HC32L190JCTA HC32L190FCUA
HC32F19	HC32F196PCTA HC32F196MCTA HC32F196KCTA HC32F196JCTA HC32F190JCTA HC32F190FCUA				

Table of Contents

1 Abstract

This application note introduces the FLASH security features, operating instructions and application methods of the UW MCU*. Caution.

- This application note is a supplement to the application of the UW MCU* and cannot replace the user's manual. Please refer to the user's manual for specific functions and register operations and other related matters.

2 FLASH Introduction

What is FLASH?

A type of flash memory device, flash memory is a non-volatile memory that can retain data for a long time without current supply. Its storage characteristics are equivalent to a hard disk, and this characteristic is the basis for flash memory to become a storage medium for various portable digital devices.

(Quoted from 'Baidu Encyclopedia', 'Interactive Encyclopedia', 'Wikipedia')

FLASH Features?

Flash memory is non-volatile memory and can be erased and reprogrammed in blocks of memory cells called blocks. A write operation to any Flash device can only be performed to an empty or erased cell, so in most cases, an erase must be performed before a write operation can be performed.

What are the applications of FLASH?

FLASH is widely used in emerging digital devices such as mobile storage, MP3 players, digital cameras, and PDAs.

*See cover for supported models.

3 FLASH Application Notes

3.1 Introduction

The UW MCU* covers FLASH memory with 16/32/64/128/256/512K bytes (Byte)512 capacity depending on the model. This memory supports erase (slice/page erase), program and read operations. In addition, the module supports protection of FLASH memory erasure and write protection of control registers.

3.2 Security Features

3.2.1 Operating source protection

WUTA MCU* adopts high security hardware design for FLASH with capacity over 32K, and has FLASH operation source defense function: FLASH erase operation can be correctly executed only when the address of FLASH operation function is in 0~32K.

FLASH address 0~32K has higher security, important functions must be placed in this area, such as important program entry, interrupt entry function, high security algorithm module, UID, AES, true random number, RTC's algorithm to cooperate and form a high security authentication system.

3.2.2 Operation target protection

The entire 64K byte FLASH memory is divided into 128pages, each 4page shares a single erase protection bit. When a page is protected, all erase operations on that page are invalid and an alarm flag and interrupt signal are generated. When any page in the FLASH memory is protected, the full erase of the FLASH is invalid, and an alarm flag and interrupt signal are generated.

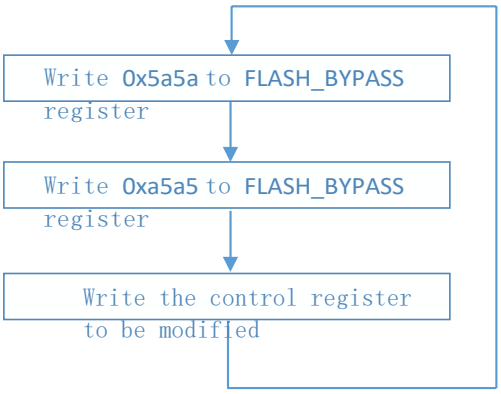
3.2.3 PC address erase protection

When the CPU runs a program in FLASH, if the current PC pointer falls within the range of the page address to be erased, the erase operation is invalid and an alarm flag and interrupt signal are generated.

*See cover for supported models.

3.2.4 Register write protection

The controller of this module blocks the ordinary write operation and must be modified by the write sequence method. The specific operation steps are shown in the following figure.



Caution.

- Write 0x5a5a, 0xa5a5, write the target register. No write operation (write ROM, RAM, REG) can be inserted between these three write operations, otherwise the value of the target register cannot be rewritten. If the rewrite fails, you need to perform these three steps again.

3.3 Function Description

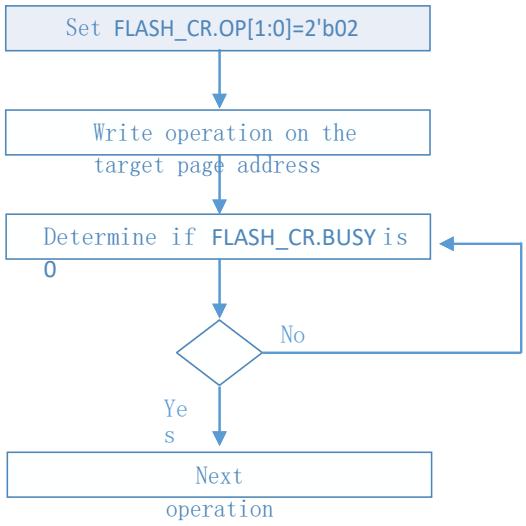
This section describes FLASH controller module functions, workflow and programming methods based on security features.

This FLASH controller supports read/write operations of Byte (8bits), Half-word (16bits), and Word (32bits) bit widths for eFLASH. Note that the address of Byte operation must be aligned by Byte, the target address of Half-word operation must be aligned by Half-word (the lowest bit of the address is 1'b0), and the address of Word operation must be aligned by Word (the lowest two bits of the address are 2'b00). If the address of a read/write operation is not aligned according to the bit-width specification, the operation is invalid and the system will enter a Hard Fault error interrupt.

3.4 Workflow Introduction

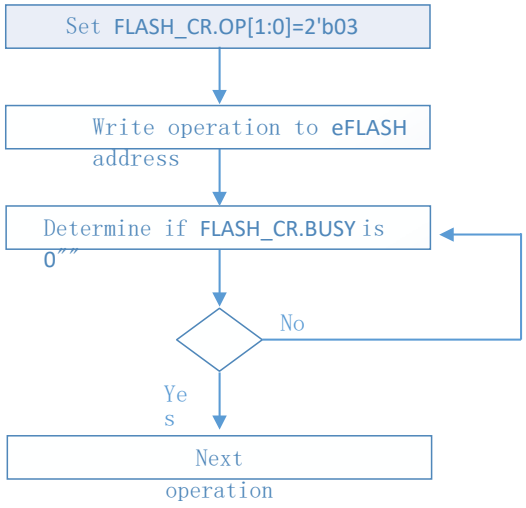
For detailed operation procedures, please refer to the user manual of the corresponding series.

3.4.1 Sector Erase



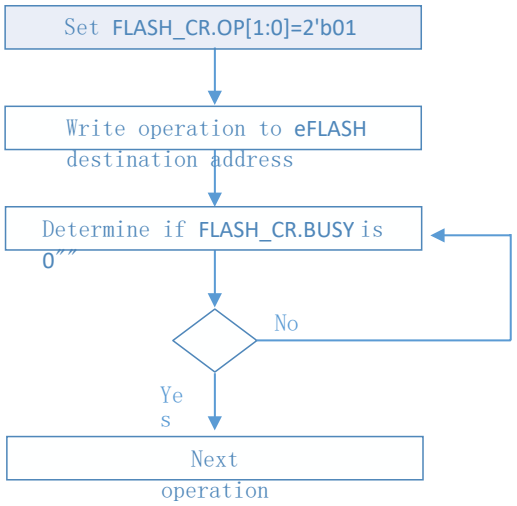
Caution.
1. The controller ignores the low9 bits of the target address as long as the target address falls within the address range of that page.
2. This write operation is used to trigger a page erase operation, and the data written is also ignored by the controller.
3. If the current erase instruction is executed in eFLASH, the CPU fetch will stop and the hardware will automatically wait for the BUSY state of eFLASH to end.
4. If the current erase instruction is executed in RAM, the CPU fetch will not stop and the software must determine if the BUSY state of eFLASH is finished before performing any operation on eFLASH.

3.4.2 Chip Erase



Caution.
1. The controller ignores the low15 bits of the target address as long as the target address falls within the eFLASH address range.
2. This write operation is used to trigger a page erase operation, and the data written is will be ignored by the controller.
3. If the current erase instruction is executed in eFLASH, the CPU fetch will stop and the hardware will automatically wait for the BUSY state of eFLASH to end.
4. If the current erase instruction is executed in RAM, the CPU fetch will not stop and the software must determine if the BUSY state of eFLASH is finished before performing any operation on eFLASH.

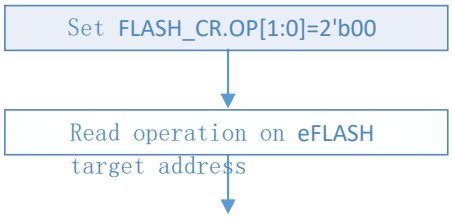
3.4.3 Write operation



Caution.

1. If the current erase instruction is executed in eFLASH, the CPU fetch will stop and the hardware will automatically wait for the BUSY state of eFLASH to end.
2. If the current erase instruction is executed in RAM, the CPU fetch will not stop, and the software must determine if the BUSY state of eFLASH is finished before any operation is performed on eFLASH.

3.4.4 Read operation



Caution.

1. The first step of setting FLASH_CR.OP[1:0] can actually be omitted, and the read operation can be performed regardless of the value of FLASH_CR.OP[1:0

3.5 Programming method based on FLASH security features

In practical application, for MCU with capacity larger than 32K, if you need to place the operation function and safety function of FLASH in the 32K safety area of FLASH, you can achieve it in the following convenient way.

Description.

- This example is to illustrate the need for the main example of the function "Flash_SectorErase()" placed in the safe area "0x400" address method, the actual application can be "example function " and "address" can be replaced according to your needs.

3.5.1 Keil MDK-based programming method

In the Keil MDK, the mapping of execution addresses for safe functions can be implemented simply as follows.

Add the following code to the declaration of the target function.

```
en_result_t Flash_SectorErase(uint32_t u32SectorAddr) attribute ((section(".ARM. at_0x400")));
```

3.5.2 IAR-based programming approach

1, add the following code to the definition of the objective function.

```
en_result_t Flash_SectorErase(uint32_t u32SectorAddr) @".Flash_SectorErase"
```

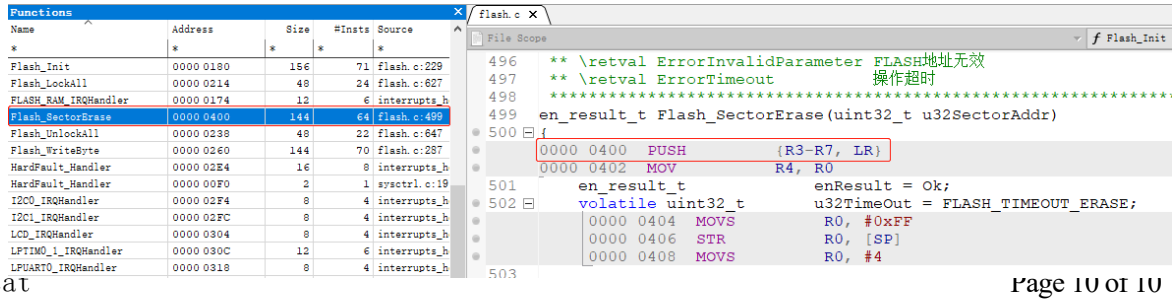
2Add the following code to the project "*.icf" file.

```
place at address mem:0x00000400 { readonly section .Flash_SectorErase};
```

3.5.3 Results view and examples

If you are interested in specific information about the results produced by this method, you can observe the execution address space of the code through a map file, a debug file, or during a debug run.

The example is as follows (you can confirm that the target function is indeed placed at the expected [0x400] address).



4 Summary

The above section introduces the security features, workflow and programming methods of FLASH based on the security features of UW MCU*, which can be modified or extended by users to meet their own applications in the actual development.

*See cover for supported models.

5 Version Information & Contact

Date	Versions	Modify records
2019/8/19	Rev1.0	Initial Release

If you have any comments or suggestions in the process of purchase and use, please feel free to contact us.

Email: mcu@hdsc.com.cn

Website: www.hdsc.com.cn

Address: No. Lane39, Bebo Road, Zhangjiang Hi-Tech Park, Shanghai,572 China

Zip code.201203