

## FLASH Program and Erasure

### Using HMS99C58S FLASH MCU

#### INTRODUCTION

The HMS99C58S Flash MCU is MCS-51 compatible and has total 32k bytes of user code area, 2kbytes of pre-installed bootloader code area. The user code area is composed of 7 blocks of Flash cells and the erase operation is performed by this block unit while the program operation can be performed by byte unit. The bootloader code is used for ISP(In-Circuit Programming) in the need of user code updating at on-board condition through UART port. The endurance of HMS99C58S is 10,000 write cycle, and data retention time is 10 years. For more information, refer to device manual.

This application note is intended for design engineers who want to implement the Flash programming and erasure of user area on bootloader code area.

#### THE HARDWARE CONNECTION

A typical hardware connection is illustrated in Figure 1. The HMS99C58S pins should be appropriately set up to operate in bootloader mode. The  $\overline{EA}$ ,  $\overline{PROG}$ , and  $\overline{PSEN}$  pins should be initialized to respectively high, high, and low state like Figure 1, that is necessary for bootloader code operation and prevention of user code activation. The Power, Reset and Oscillation clock lines should be applied for proper MCU operation, and 4MHz oscillator is used in this example.

#### THE SOFTWARE STRUCTURE

An example programming and erasure algorithm is listed in Appendix A. A flow diagram is given in Figure 2. The function of example source of HMS99C58S is to erase all the user block and write data 55h fully.

**Note:** At Erase and Program mode, the combination of target block in user code area and source code in bootcode area is possible. But the vice versa is impossible.

The user should take the following steps when implementing Erase and Program operation.

##### A) Erase operation

1. Prewrite all the user area block with data 00h. The 00h prewrite operation makes the all the flash cells get equal electrical charge level so that the overerase may not occur. The overerase can cause unwanted cells to be erased. In this Prewrite case, the Program operation can be done by 8 byte unit because of the programming of the only data 00h, differently from the normal programming.

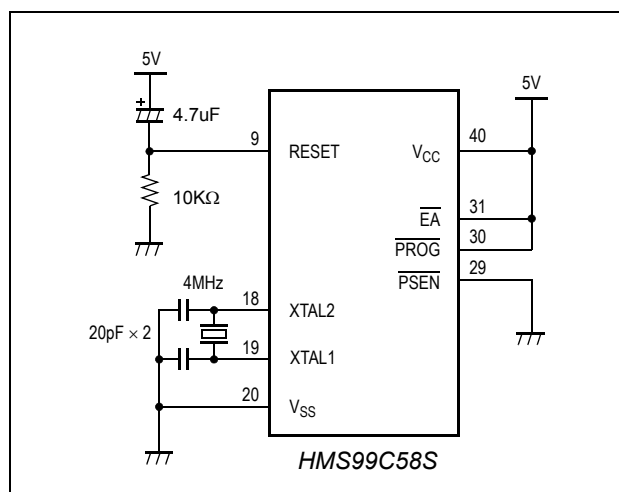


Figure 1. Bootloader mode entry

2. Check the data from the first address in each block to be blank data or not. If not blank, perform Erase operation with 10ms of erase time.
3. Check the the data from the latest address to be blank data or not. If not blank, perform Erase operation at this block again.
4. The maximum repeat count of Erase operation is 40 times in this case. If the retrial count becomes greater than 40, the Erase operation results in failure.

##### B) Program operation

1. Program the data 55h at the designated address in block. The Program operation is performed with 30us of write time.
2. Check the written data to be the wanted data. If not, perform the Program operation again.
3. The maximum repeat count of Program operation is 5 times in this case. If the retrial count becomes greater than the 5, the Program operation results in failure.

#### MAIN FUNCTIONS

The example source has a few following useful functions that deal with Erase and Program operation. The respective function's parameter and work are as follows.

- hex\_erase\_with\_prewrite :

parameter : ACC(Block Index)

The 00h Prewrite and Erase operation is performed according to the ACC register that represents the corresponding block index with its bit position.

Code size: 0.5k bytes of program code

- erase\_2k\_block, erase\_8k\_block

parameter : R3(Erase Block Index), DPTR(Block Start Address)

The erasure of one block is performed in corresponding block area designated by R3.

*Author: Vince Jeong  
MCU application team  
e-mail: byoungsu.jeong@hynix.com*

- flash\_erase\_start :

parameter : ACC(Erasure Block Index)

The actual block erasure is executed by setting of erase control register in this function.

- flash\_programming\_start :

parameter : ACC(write data), DPTR(program address)

The actual byte programming is executed by setting of program control register in this function.

## SUMMARY

The erasure operation of HMS99C58S needs the 00h pre-writing for protection of overerase and maximum 40 times of erasure operation that includes blank check. The programming operation needs 5 times of programming operation that includes verify operation. The generic code in Appendix A makes it easy to incorporate in HMS99C58S series FLASH MCU application. Any of the HMS99C58S Series MCU can be applied to this example.

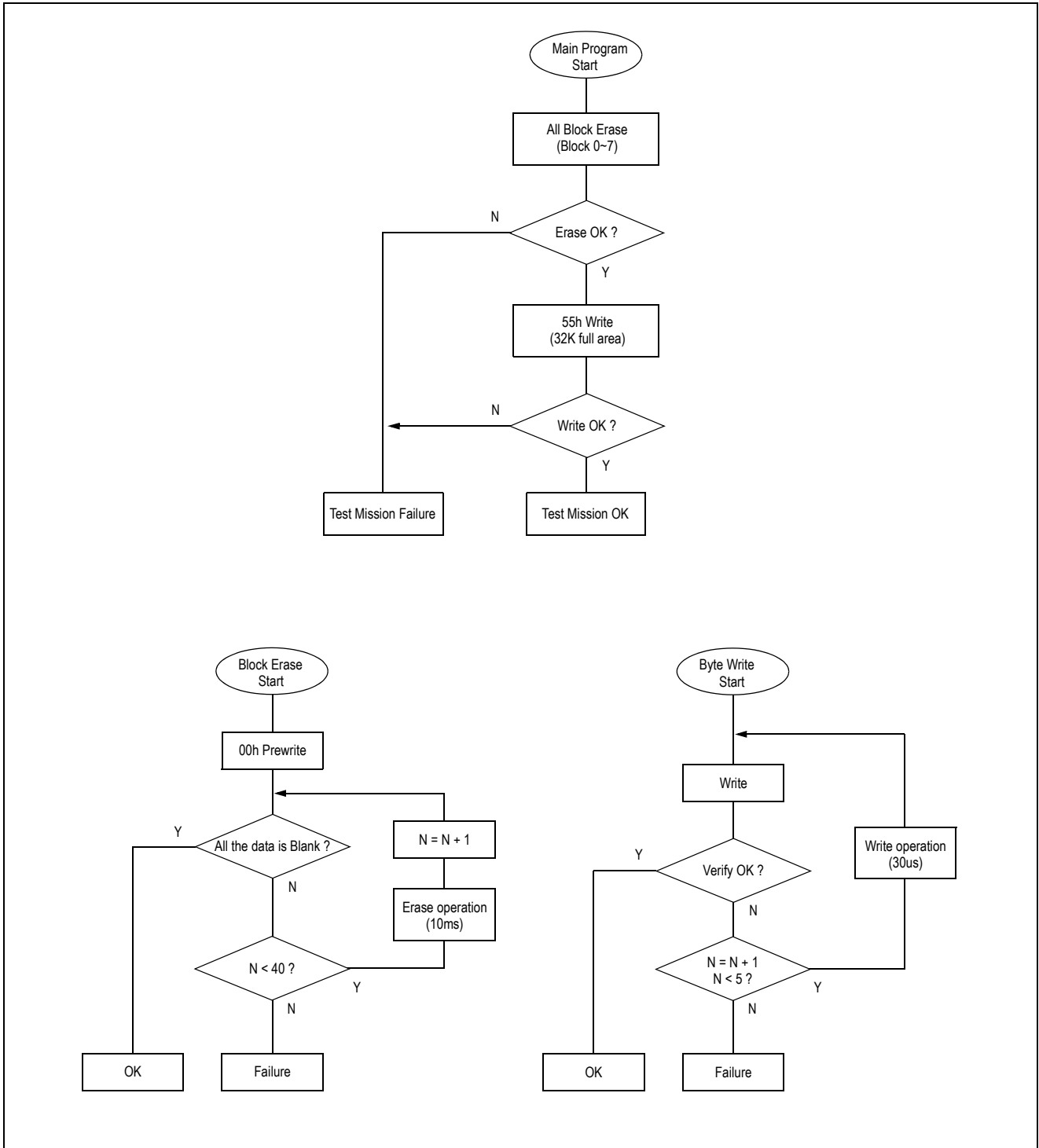


Figure 2. Flow Chart of Flash Erase and Program Operation

## APPENDIX A

```

;*****
; NAME:          bootloader.a51
;-----
;               It supports 32 Kbytes USER memory

$TITLE      (** Boot Loader **)

BOOTLOADER_MODULE  SEGMENT    CODE
BOOTLOADER_DATA    SEGMENT    DATA

;_____ I N C L U D E S _____

$include (register.inc)

;_____ D E F I N I T I O N _____

#define          ERASE_MAX_COUNT      40
#define          WRITE_MAX_COUNT      5

#define          OK_MESSAGE           055h
#define          FAIL_MESSAGE        0aah

; *** DATA RAM INTERNAL ***

RSEG          BOOTLOADER_DATA
write_buffer:      DS          1
hex_buffer:       DS          16
stack:           DS          40h

;_____ D E C L A R A T I O N _____

RSEG          BOOTLOADER_MODULE

;F*****
; NAME:          boot_loader
;*****
; NOTE:         located at address BOOT_ENTRY
;               Boot entry process:
;*****
boot_loader:

;erase test
        mov     A,#7fh                ;block index for erase, all block erase
        acall  hex_erase_with_prewrite
        cjne   A, #OK_MESSAGE, hex_erase_fail
        sjmp   program_test

hex_erase_fail:
        sjmp   test_mission_fail

;*****
program_test:
        mov     DPH,#00h              ; DPTR = programming start address
        mov     DPL,#00h
        acall  flash_programming_setup;programming setup

        mov     R0,#128                ;128, 128 * 256 = 32K
pgm_loop:
        mov     R1,#00h                ;256 * 32 = 8K
pgm_loop_2:
        mov     A,#55h
        acall  flash_programming_start

```

```

        cjne        A, #OK_MESSAGE, hex_program_fail

hex_program_OK:
        inc        DPTR
        djnz       R1, pgm_loop_2
        djnz       R0, pgm_loop
        sjmp       test_mission_ok

hex_program_fail:
        sjmp       test_mission_fail

test_mission_ok:
        sjmp       test_mission_ok
test_mission_fail:
        sjmp       test_mission_fail

;*****
hex_erase_with_prewrite:
        mov        hex_buffer, A                ;block index for erase
        acall      flash_programming_setup
hex_block_prewrite_0_parallel:
        mov        A,hex_buffer                ;block index for erase
        jnb        ACC.0, hex_block_prewrite_1_parallel
        acall      block0_prewrite
hex_block_prewrite_1_parallel:
        mov        A,hex_buffer                ;block index for erase
        jnb        ACC.1, hex_block_prewrite_2_parallel
        acall      block1_prewrite
hex_block_prewrite_2_parallel:
        mov        A,hex_buffer                ;block index for erase
        jnb        ACC.2, hex_block_prewrite_3_parallel
        acall      block2_prewrite
hex_block_prewrite_3_parallel:
        mov        A,hex_buffer                ;block index for erase
        jnb        ACC.3, hex_block_prewrite_4_parallel
        acall      block3_prewrite
hex_block_prewrite_4_parallel:
        mov        A,hex_buffer                ;block index for erase
        jnb        ACC.4, hex_block_prewrite_5_parallel
        acall      block4_prewrite
hex_block_prewrite_5_parallel:
        mov        A,hex_buffer                ;block index for erase
        jnb        ACC.5, hex_block_prewrite_6_parallel
        acall      block5_prewrite
hex_block_prewrite_6_parallel:
        mov        A,hex_buffer                ;block index for erase
        jnb        ACC.6, erase_after_prewrite_check
        acall      block6_prewrite
erase_after_prewrite_check:
        mov        A,hex_buffer                ;block index for erase
        anl        A, #07fh
        cjne       A, #00h, erase_after_prewrite
        sjmp       hex_block_prewrite_null
erase_after_prewrite:
        ajmp       hex_erase_start            ;go to erase
hex_block_prewrite_null:
        mov        A, #FAIL_MESSAGE
        ret

;*****
hex_erase_start:

```

```

        acall        flash_erase_setup
hex_block_erase_0_parallel:
    mov            A,hex_buffer            ;block index for erase
    jnb           ACC.0, hex_block_erase_1_parallel
    acall         block0_erase
    cjne          A, #OK_MESSAGE, hex_erase_block_fail
hex_block_erase_1_parallel:
    mov            A,hex_buffer            ;block index for erase
    jnb           ACC.1, hex_block_erase_2_parallel
    acall         block1_erase
    cjne          A, #OK_MESSAGE, hex_erase_block_fail
hex_block_erase_2_parallel:
    mov            A,hex_buffer            ;block index for erase
    jnb           ACC.2, hex_block_erase_3_parallel
    acall         block2_erase
    cjne          A, #OK_MESSAGE, hex_erase_block_fail
hex_block_erase_3_parallel:
    mov            A,hex_buffer            ;block index for erase
    jnb           ACC.3, hex_block_erase_4_parallel
    acall         block3_erase
    cjne          A, #OK_MESSAGE, hex_erase_block_fail
hex_block_erase_4_parallel:
    mov            A,hex_buffer            ;block index for erase
    jnb           ACC.4, hex_block_erase_5_parallel
    acall         block4_erase
    cjne          A, #OK_MESSAGE, hex_erase_block_fail
hex_block_erase_5_parallel:
    mov            A,hex_buffer            ;block index for erase
    jnb           ACC.5, hex_block_erase_6_parallel
    acall         block5_erase
    cjne          A, #OK_MESSAGE, hex_erase_block_fail
hex_block_erase_6_parallel:
    mov            A,hex_buffer            ;block index for erase
    jnb           ACC.6, hex_erase_block_ok
    acall         block6_erase
    cjne          A, #OK_MESSAGE, hex_erase_block_fail
hex_erase_block_ok:
    mov            A, #OK_MESSAGE
    sjmp          hex_erase_without_prewrite_end
hex_erase_block_fail:
    mov            A, #FAIL_MESSAGE
hex_erase_without_prewrite_end:
    ret

;*****
prewrite_8K_block_00:
;parameter : DPH, DPL
    mov            R6, DPH
    mov            R7, DPL
    acall         prewrite_8B_ready
    mov            DPH,R6                ; DPTR = programming address
    mov            DPL,R7
    mov            R0,#4                ;8K bytes(4*8*256=8192) writing
    ajmp          first_zero_preprogram_loop

;=====
prewrite_2K_block_00:
;parameter : DPH, DPL
    mov            R6, DPH
    mov            R7, DPL
    acall         prewrite_8B_ready
    mov            DPH,R6                ; DPTR = programming address

```

```

        mov     DPL,R7
        mov     R0,#1                ;8K bytes(1*8*256=2048) writing

first_zero_preprogram_loop:
        mov     R1,#00h              ;256; R1*R2= number of byte to write
zero_preprogram_loop_256:
        mov     FCON,#8Fh           ;flash 8Byte write
        inc     DPTR
        inc     DPTR
        inc     DPTR
        inc     DPTR
        inc     DPTR
        inc     DPTR
        inc     DPTR
        inc     DPTR
        djnz    R1,zero_preprogram_loop_256
        djnz    R0,first_zero_preprogram_loop
        ret

prewrite_8B_ready:
        mov     DPL,#0
        mov     A,#0
        mov     R0,#8

load_8B:
        mov     FCON,#8Ch
        inc     DPL
        djnz    R0,load_8B
        ret

;*****
erase_2K_block:
;parameter : ACC(EBR), DPH, DPL
        mov     R3,A                 ;EBR, block index
        mov     R4,#ERASE_MAX_COUNT
        mov     R0,#08h             ;8,8*256=2K
        sjmp    erase_blank_check_loop
;=====
erase_8K_block:
;parameter : ACC(EBR), DPH, DPL
        mov     R3,A                 ;EBR, block index
        mov     R4,#ERASE_MAX_COUNT
        mov     R0,#20h             ;32, 32 * 256 = 8K
erase_blank_check_loop:
        mov     R1,#00h              ;256 * 32 = 8K
erase_blank_check_loop_2:
        mov     A,#00h
        movc    A,@A+DPTR
        cjne    A,#0FFh,erase_operation
        inc     DPTR
        djnz    R1,erase_blank_check_loop_2
        djnz    R0,erase_blank_check_loop
;flash_erase_sector_ok:
        mov     A,#OK_MESSAGE
        ret

erase_operation:
        mov     A,R4
        jz      flash_erase_sector_fail
        mov     A,R3                 ;EBR
        acall   flash_erase_start
        sjmp    erase_blank_check_loop_2

```

```

flash_erase_sector_fail:
    mov     A, #FAIL_MESSAGE
    ret

;=====
block0_prewrite:
    mov     DPH, #00h
    mov     DPL, #00h
    acall   prewrite_2K_block_00
    ret

block1_prewrite:
    mov     DPH, #08h
    mov     DPL, #00h
    acall   prewrite_2K_block_00
    ret

block2_prewrite:
    mov     DPH, #10h
    mov     DPL, #00h
    acall   prewrite_2K_block_00
    ret

block3_prewrite:
    mov     DPH, #18h
    mov     DPL, #00h
    acall   prewrite_2K_block_00
    ret

block4_prewrite:
    mov     DPH, #20h
    mov     DPL, #00h
    acall   prewrite_8K_block_00
    ret

block5_prewrite:
    mov     DPH, #40h
    mov     DPL, #00h
    acall   prewrite_8K_block_00
    ret

block6_prewrite:
    mov     DPH, #60h
    mov     DPL, #00h
    acall   prewrite_8K_block_00
    ret

;=====
block0_erase:
    mov     DPH,#00h           ;0000h
    mov     DPL,#00h
    mov     a, #01h           ;setting EBR
    acall   erase_2K_block
    ret

block1_erase:
    mov     DPH,#08h           ;0800h
    mov     DPL,#00h
    mov     a, #02h           ;setting EBR
    acall   erase_2K_block
    ret

block2_erase:
    mov     DPH,#10h           ; 1000h
    mov     DPL,#00h
    mov     a, #04h           ;setting EBR
    acall   erase_2K_block
    ret

block3_erase:
    mov     DPH,#18h           ; 1800h

```

```

        mov     DPL,#00h
        mov     a, #08h           ;setting EBR
        acall  erase_2K_block
        ret
block4_erase:
        mov     DPH,#20h         ;2000h
        mov     DPL,#00h
        mov     a, #010h        ;setting EBR
        acall  erase_8K_block
        ret
block5_erase:
        mov     DPH,#40h         ;4000h
        mov     DPL,#00h
        mov     a, #020h        ;setting EBR
        acall  erase_8K_block
        ret
block6_erase:
        mov     DPH,#60h         ;6000h
        mov     DPL,#00h
        mov     a, #040h        ;setting EBR
        acall  erase_8K_block
        ret

;F*****
; NAME:      flash_programming_setup
;-----
flash_programming_setup:
        acall  flash_pgm_erase_setup
        mov     th2,#0ffh        ;(fff5h,4MHz,30us)
        mov     tl2,#0f5h
        mov     RCAP2H,#0ffh
        mov     RCAP2L,#0f5h
        ret

;*****
flash_programming_start:
        mov     write_buffer, A   ;save for verify
        mov     R7,#WRITE_MAX_COUNT

flash_programming_loop:
        mov     A, write_buffer   ;save for verify
        mov     FCON,#8Dh

;write_verify
        mov     A,#00h
        movc   A,@A+DPTR
        cjne   A,write_buffer,hex_program_count_check
        mov     A, #OK_MESSAGE
        ret
hex_program_count_check:
        djnz   R7, flash_programming_loop
        mov     A, #FAIL_MESSAGE
        ret

;F*****
flash_pgm_erase_setup:
        mov     t2con, #00h
        mov     ip, #00h
        mov     ie, #00h
        mov     a, #03h           ;PGM Power setup
        mov     FCON, #0ach       ;acc => CONREG
        mov     a, #80h           ;PGM Vpp voltage select
        mov     FCON, #0ech       ;acc => PWRREG

```

```
ret

;F*****
; NAME:    flash_erase_setup
;-----
flash_erase_setup:
    acall    flash_pgm_erase_setup
    mov     th2, #0f2h          ;f2fah,4MHz(10ms)
    mov     tl2, #0fah
    mov     RCAP2H,#0f2h
    mov     RCAP2L,#0fah
    ret

;*****
flash_erase_start:
;ACC(ERASE BLOCK)
    mov     FCON, #0cch        ;acc => EBR
    mov     FCON, #01ch        ;SEC erase,
    ret

END
```