EEPROM 功能使用方法

1 適用產品：

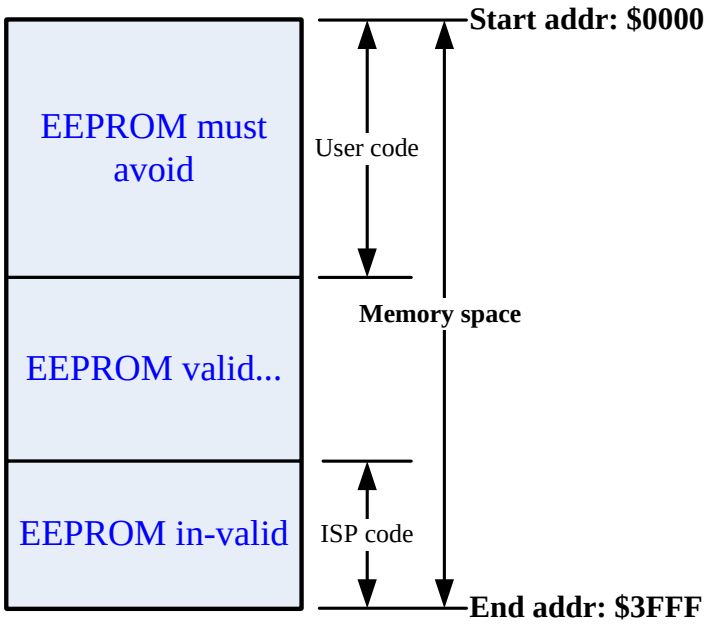
1.1 SM39R16A2/ SM39R12A2/ SM39R08A2
1.2 SM39R4051/ SM39R2051
1.3 SM39R04G1/ SM39R02G1

2 EEPROM 功能概述：

2.1 使用 code flash 模擬爲 Internal EEPROM，在程序執行時，可將 code flash 作爲 data flash 儲存資料使用。

2.2 EEPROM command 包括：Chip erase、page erase 及 byte program。

2.3 EEPROM 必須避免定址在主程序(user code)區域；而爲保護 ISP code，在 ISP 保護區(ISP block N)所保護的位置，EEPROM command 將視爲無效。



2.4 MCU 空片及 erase 後 code flash 皆爲 FFh。Flash 每一 page=256 byte。

2.5 Page erase 及 byte program 時間紀錄：

EEPROM 執行 command 時間，爲系統內振，與 user 設定的系統晶振頻率無關	
page erase (ms)	byte program(ms)
about 28~35ms	about 0.065~0.075

2.6 EEPROM 相關的特殊暫存器 Special Function Register (SFR)

2.6.1 EERPOM register – TAKEY, IFCON, ISPFAH, ISPFAL, ISPFD and ISPFC

Mnemonic	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
ISP function											
TAKEY	Time Access Key register	F7h	TAKEY [7:0]								00H
IFCON	Interface Control register	8Fh	-	CDPR	-	-	-	-	-	ISPE	00H
ISPFAH	ISP Flash Address - High register	E1h	ISPFAH [7:0]								FFH
ISPFAL	ISP Flash Address - Low register	E2h	ISPFAL [7:0]								FFH
ISPFD	ISP Flash Data register	E3h	ISPFD [7:0]								FFH
ISPFC	ISP Flash Control register	E4h	EMF1	EMF2	EMF3	EMF4	-	ISPF.2	ISPF.1	ISPF.0	00H

Mnemonic: TAKEY **Address: F7H**

7	6	5	4	3	2	1	0	Reset
TAKEY [7:0]								00H

ISP 致能位元(ISPE)預設爲***唯讀屬性***，爲避免程序錯誤導致 ISP 誤動作，使用者必須依序寫入三筆資料到(55h, AAh, 5Ah)TAKEY，才可將 ISP 致能位元(ISPE)改變爲***可寫入屬性***，程序參考如下：

```
MOV TAKEY, #055h
MOV TAKEY, #0AAh
MOV TAKEY, #05Ah      ; enable ISPE write attribute
ORL IFCON, #001H      ; enable ISP function
```

注意：以上程序，中間不可有任何程序執行，包括不可有程序中斷(Interrupt)或設定模擬器中斷(break point)干擾其操作流程，必須依序且完整的連續寫入，否則該功能將無法致能。完整程序可參考 Assembly 或 C 語言範例程序。

Mnemonic: IFCON **Address: 8FH**

7	6	5	4	3	2	1	0	Reset
ITS	CDPR	F12K	F8K	ALEC[1]	ALEC[0]	EMEN	ISPE	00H

ISP 致能位元(ISPE)預設爲**唯讀屬性**，爲避免程序錯誤導致 ISP 誤動作，使用者必須依序寫入三筆資料到(55h, AAh, 5Ah)TAKEY，才可將 ISP 致能位元(ISPE)改變爲**可寫入屬性**：

ISPE: = 1, ISP致能，ISP暫存器(ISPFAH, ISPFAL, ISPFD , ISPFC)設爲可寫入。

= 0, ISP禁能，ISP暫存器(ISPFAH, ISPFAL, ISPFD and ISPFC)爲唯讀 (預設)。

程序範例，ISP byte program #22H 到 program flash 位置$1005H，如下：

```
MOV TAKEY, #055h
MOV TAKEY, #0AAh
MOV TAKEY, #05Ah      ; enable ISPE write attribute
ORL   IFCON, #001H    ; enable ISP function
MOV ISPFAH, #010H     ; set flash address-high, 10H
MOV ISPFAL, #005H     ; set flash address-low, 05H
MOV ISPFD, #022H      ; set flash data to be programmed, data = 22H
MOV ISPFC, #000H      ; start to program #22H to the flash address $1005H
```

Mnemonic: ISPFAH **Address: E1H**

7	6	5	4	3	2	1	0	Reset
ISPFAH7	ISPFAH6	ISPFAH5	ISPFAH4	ISPFAH3	ISPFAH2	ISPFAH1	ISPFAH0	FFH

ISPFAH [7:0]: ISP 共提供 16 位元定址，此爲高位元 8~15 位置。

Mnemonic: ISPFAL **Address: E2H**

7	6	5	4	3	2	1	0	Reset
ISPFAL7	ISPFAL6	ISPFAL5	ISPFAL4	ISPFAL3	ISPFAL2	ISPFAL1	ISPFAL0	FFH

ISPFAL [7:0]: ISP 共提供 16 位元定址，此爲低位元 0~7 的位置。

Mnemonic: ISPFD **Address: E3H**

7	6	5	4	3	2	1	0	Reset
ISPFD7	ISPFD6	ISPFD5	ISPFD4	ISPFD3	ISPFD2	ISPFD1	ISPFD0	FFH

ISPFD [7:0]: ISP 資料暫存器。

Mnemonic: ISPFC **Address: E4H**

7	6	5	4	3	2	1	0	Reset
EMF1	EMF2	EMF3	EMF4	-	ISPF[2]	ISPF[1]	ISPF[0]	00H

EMF1: Entry mechanism (1) flag, clear by reset. (Read only)
ISP 進入記錄旗標(1)，唯讀，可由晶片復位清除

EMF2: Entry mechanism (2) flag, clear by reset. (Read only)
ISP 進入記錄旗標(2)，唯讀，可由晶片復位清除

EMF3: Entry mechanism (3) flag, clear by reset. (Read only)
ISP 進入記錄旗標(3)，唯讀，可由晶片復位清除

EMF4: Entry mechanism (4) flag, clear by reset. (Read only)
ISP 進入記錄旗標(4)，唯讀，可由晶片復位清除

ISPF [2:0]: ISP function select bit.
ISP 功能選擇元位，提供七組功能

ISPF[2:0]	ISP function
000	Byte program
001	Chip protect
010	Page erase
011	Chip erase
100	Write option
101	Read option
110	Erase option
111	Finish Flag*(註一)

註一：只有Write option或Erase option後，才須執行Finish flag指令以示完成，否則寫入值皆視爲無效。EEPROM不須執行Finish Flag。

2.7 EEPROM Assembly 語言範例程式：

Description	此範例展示有 EEP_Byte_Program、EEP_Chip_Protect、EEP_Page_Erase、EEP_Byte_Read、EEP_Enable 及 EEP_Disable 基本功能，提供使用者自行開發 ISP code 或 EEPROM 應用參考。

Main program

```
//=====================================================================
//
//              S Y N C M O S   T E C H N O L O G Y
//
//=====================================================================
#include "SM39R16A2.h"

//ISPFC.ISPF[2:0]
#define d_COMMAND_BYTE_PROGRAM 0
#define d_COMMAND_CHIP_PROTECT 1
#define d_COMMAND_PAGE_ERASE   2
//#define d_COMMAND_CHIP_ERASE 3

dat     equ  30h
addh equ  31h
addl equ  32h

    org      0000h        //Start
Start:
    //test1======================
    mov      addh,    #0x01
```

```
     mov      addl,    #0x00
     mov      dat,     #0x10
     call     EEP_Byte_Program
     call     EEP_Byte_Read
     mov      P0,      dat
     //test2=====================
     mov      addh,    #0x02
     mov      addl,    #0x00
     mov      dat,     #0x20
     call     EEP_Byte_Program
     call     EEP_Byte_Read
     mov      P1,      dat
     //==========================
     jmp      $

EEP_Byte_Program:
     call     EEP_Enable
     mov      ISPFD,   dat
     mov      ISPFAH,  addh
     mov      ISPFAL,  addl
     mov      ISPFC,   #d_COMMAND_BYTE_PROGRAM
     call     EEP_Disable
     ret
EEP_Chip_Protect:
     call     EEP_Enable
     mov      ISPFC,   #d_COMMAND_CHIP_PROTECT;
     call     EEP_Disable
     ret
EEP_Page_Erase://256-Byte
     call     EEP_Enable
     mov      ISPFAH,  addh
     mov      ISPFC,   #d_COMMAND_PAGE_ERASE
     call     EEP_Disable
     ret
EEP_Byte_Read:
     clr      A
     mov      dph,     addh
     mov      dpl,     addl
     movc     A,       @a+dptr
     mov      dat,     A
     ret
EEP_Enable:
     MOV      A,       IFCON
     JB       0xE0.0,  EEP_Enable_ret
     mov      TAKEY,   #0x55
     mov      TAKEY,   #0xAA
     mov      TAKEY,   #0x5A
     orl      IFCON,   #0x01         // ISPE=1, Enable ISP function
     MOV      A,       IFCON
     JNB      0xE0.0,  EEP_Enable
EEP_Enable_ret:
     ret
EEP_Disable:
     MOV      A,       IFCON
     JNB      0xE0.0,  EEP_Disable_ret
     mov      TAKEY,   #0x55
     mov      TAKEY,   #0xAA
```

```
     mov       TAKEY,   #0x5A
     anl       IFCON,   #0xFE         // ISPE=1, Enable ISP function
     MOV       A,       IFCON
     JB        0xE0.0,  EEP_Disable
EEP_Disable_ret:
     ret
     end
```

2.8 EEPROM C 語言範例程式：

Description

此範例展示有 EEP_Byte_Program、EEP_Chip_Protect、EEP_Page_Erase、EEP_Byte_Read、EEP_Enable 及 EEP_Disable 基本功能，提供使用者自行開發 ISP code 或 EEPROM 應用參考。

EEPROM program

```
//=====================================================================
//
//               S Y N C M O S   T E C H N O L O G Y
//
//=====================================================================
#include <source\SM39R04G1.h>
#include <absacc.h>    /* Include Macro Definitions */

/*ISPFC.ISPF[2:0]*/
#define d_Command_Byte_Program 0
#define d_Command_Chip_Protect 1
#define d_Command_Page_Erase   2
#define d_Command_Chip_Erase   3

//=====================================================================
#define d_DATALEN 128               // idata limit:1~128
unsigned char idata buf[d_DATALEN]; // idata limit:1~128

void EEPROM_Enable(void)
{
     TAKEY  = 0x55;
     TAKEY  = 0xAA;
     TAKEY  = 0x5A;
     IFCON |= 0x01;         // ISPE=1, Enable ISP function
}

void EEPROM_Disable(void)
{
     TAKEY  = 0x55;
     TAKEY  = 0xAA;
     TAKEY  = 0x5A;
     IFCON &= 0xFE;         // ISPE=0, Disable  ISP function
}

void EEPROM_Byte_Program(unsigned int Addr, unsigned char Data)
{
     EEPROM_Enable();
     ISPFD    = Data;
     ISPFAH   = (Addr / 256);
     ISPFAL   = (Addr % 256);
     ISPFC    = d_Command_Byte_Program;
     EEPROM_Disable();
}
```

```
void EEPROM_Page_Erase(unsigned int Addr)        // erase 256-Byte
{
     EEPROM_Enable();
     ISPFAH   = (unsigned char)(Addr >>8);
     ISPFC    = d_Command_Page_Erase;
     EEPROM_Disable();
}
/*
void EEPROM_Page_Erase(unsigned int Addr)        // erase while chip
{
     EEPROM_Enable();
     ISPFAH   = (unsigned char)(Addr >>8);
     ISPFC    = d_Command_Chip_Erase;
     EEPROM_Disable();
}
*/

/*
void EEPROM_Chip_Protect(void)
{
     EEPROM_Enable();
     ISPFC    = d_Command_Chip_Protect;
     EEPROM_Disable();
}
*/
void EEPROM Sector Program(unsigned int Addr start, unsigned int Addr end, unsigned
char Data)
{
     unsigned int i;
     for(i=Addr_start; i<=Addr_end; i++)
     {
          EEPROM_Byte_Program(i, Data);
     }
/*   for(i=0; i<(Addr_end - Addr_start)+1; i++)
     {
          EEPROM_Byte_Program(Addr_start+i, Data);
     }
*/
}

unsigned char EEPROM_Byte_Read(unsigned int Addr)
{
/*   unsigned char temp;
     unsigned char code *pAddr;
     pAddr = Addr;                    // Set address
     temp = (*pAddr);           // Read data
     return temp;
*/
     return CBYTE[Addr];             //The range of valid index values for this macro
is 0-65535.
}
/*
unsigned int EEPROM_Word_Read(unsigned int Addr)
{
     return CWORD[Addr];             //The range of valid index values for this macro
is 0-32767.
```

```
}
*/
void EEPROM_Byte_Modify(unsigned int Addr, unsigned char Data)
{
     unsigned char i;

     for(i=0; i<d_DATALEN; i++)
          buf[i] = EEPROM_Byte_Read((Addr&0xFF00)+i);  // page read

     EEPROM_Page_Erase(Addr);                          // erase page

     buf[(unsigned char)Addr] = Data;                  // byte modify

     for(i=0; i<d_DATALEN; i++)
          EEPROM_Byte_Program((Addr&0xFF00)+i, buf[i]);// page program
}
```

Main.c

```
#include <source\SM39R04G1.h>
#include <source\EEPROM.h>

#define test_len      100
#define test_start_addr    0x1000
void main(void)
{
     unsigned int i;
     unsigned char buf[test_len];

     EEPROM_Page_Erase(test_start_addr);

     EEPROM_Sector_Program(test_start_addr, test_start_addr+test_len-1, 0x55);
     for(i=0; i<test_len; i++)
     {
          buf[i]= EEPROM_Byte_Read(test_start_addr+i);
     }

     for(i=0; i<test_len; i++)
     {
          EEPROM_Byte_Modify(test_start_addr+i, i);
     }

     for(i=0; i<test_len; i++)
     {
          buf[i]= EEPROM_Byte_Read(test_start_addr+i);
     }

     while(1)
     {

     }
}
```

新茂國際科技 希望能爲客戶減少開發的時間及辛勞，針對所有特殊功能應在C語言程序開發，皆提供 “Codzard 範例程式產生器” 可於 新茂網站首頁>下載專區> 軟體下載 內下載此軟體，如有任何建議，請來信告知，謝謝!

銷售客服
電子信箱： sales@mail.syncmos.com.tw

技術支援
電子信箱： support@mail.syncmos.com.tw