ISP & EEPROM 功能使用方法

1 適用產品：

1.1 SM59R16A2/ SM59R08A2

1.2 SM59R16A5/ SM59R09A5/ SM59R05A5/ SM59R16A3/ SM59R09A3/ SM59R05A3

1.3 SM59R16G6/ SM59R09G6/ SM59R05G6

1.4 SM59R04A2/ SM59R04A1/ SM59R03A1/ SM59R02A1

2 功能概述：

2.1 SM59R 系列皆有提供 ISP code 燒錄檔及 ISP source code，利於客戶生產及開發。

2.2 ISP(In System Program)與 IAP(In Application Program)之差異說明：

- **IAP**：SM59R系列使用code flash模擬爲Internal EEPROM，在客戶應用程序執行中，即可變更code flash，指令包括(1)byte program (2)chip erase (3)page erase (4)chip protect。
- **ISP**：MCU 線上燒寫編程常用的方式之一，屬於 IAP+UART(或其它通訊方式…)的應用，新茂標準的 ISP code 都是使用 UART0。另除了 IAP 的指令可使用外，還可以執行功能選擇位元區域做讀寫的變更，指令包括 (1)Write option (2)Read option (3)Erase option (4)Finish flag。

2.3 SM59R 系列 MCU 各型號差異比較列表(一)：

ISP addr. / Device	addr. start	addr. end	ISP code size 最大限制(KB)	Page size (Byte)	ISP code supported	ISP H/W Entry Mechanism
SM59R16A5	F000h	FFFFh	4K byte	256	有	有
SM59R09A5	F000h	FFFFh	4K byte	256	有	有
SM59R05A5	F000h	FFFFh	4K byte	256	有	有
SM59R16A3	F000h	FFFFh	4K byte	256	有	有
SM59R09A3	F000h	FFFFh	4K byte	256	有	有
SM59R05A3	F000h	FFFFh	4K byte	256	有	有
SM59R16G6	F000h	FFFFh	4K byte	256	有	有
SM59R09G6	F000h	FFFFh	4K byte	256	有	有
SM59R05G6	F000h	FFFFh	4K byte	256	有	有

SM59R16A2	—	—	無限制	512	有	無
SM59R08A2	—	—	無限制	512	有	無
SM59R04A2	3000h	3FFFh	4K byte	256	有	有
SM59R04A1	3000h	3FFFh	4K byte	256	有	有
SM59R03A1	3000h	3FFFh	4K byte	256	有	有
SM59R02A1	3000h	3FFFh	4K byte	256	有	有

Notice: 上述只有 **SM59R16A2** 及 **SM59R08A2** 出廠沒有帶 **ISP code**，其它出廠皆有。

(如有須要請聯繫代理商或新茂原廠提供即可)

2.4 SM59R 系列 MCU 各型號差異比較列表(二)：

		Device			**Device**		
寫入位置	燒寫指令	**SM59R16A5/ SM59R09A5/ SM59R05A5/ SM59R16A3/ SM59R09A3/ SM59R05A3/ SM59R16G6/ SM59R09G6/ SM59R05G6/ SM59R04A2/ SM59R04A1/ SM59R03A1/ SM59R02A1/**			**SM59R16A2/ SM59R08A2**		
		IAP	**ISP**	**ICP**	**IAP**	**ISP**	**ICP**
Code flash	**byte program**	O	O	O	O	O	O
	chip erase	O	O	O	O	O	O
	page erase	O	O	O	O	O	O
	chip protect	O	O	O	X	X	O
Function Option (功能選擇位元)	**Read option**	O	O	O	X	X	O
	Write option	X	O	O	X	X	O
	Erase option	X	O	O	X	X	O
	Finish flag*(註一)	X	O	O	X	X	X

註一：只有SM59R16G6/ SM59R09G6/ SM59R05G6有此功能。

3 以下說明適用：SM59R16A5/ SM59R09A5/ SM59R05A5/ SM59R16A3 SM59R09A3/ SM59R05A3/ SM59R16G6/ SM59R09G6/ SM59R05G6/ SM59R04A2/ SM59R04A1/ SM59R03A1/ SM59R02A1
(關於 SM59R16A2/ SM59R08A2 請參考第 4 章節說明)。

3.1 ISP service program 最大可設計爲 4K byte，共可區分爲 16 個 page，每一 page 爲 256 byte；未使用到的部分仍可當作 code flash 使用(ISP 定址位址請參考：各型號 ISP 差異列表)。

3.2 chip protect，當 chip protect 之前，code flash (包括 ISP service program)仍可被 writer 或 ICE 讀出，chip protect 之後，code flash (包括 ISP service program)則不可讀出。

3.3 ISP service program 進入方式：

3.3.1 可由使用者選擇致能或禁能，但系統設計時必須注意各種條件，以避免誤入ISP service program。

3.3.2 ISP service program的進入方式，條件如下(以下復位僅限制爲POR或Pad reset)：

- ☐ 硬件：code flash addr. $0000=#FFH。
- ☐ 硬件：P2.6, P2.7爲低電位。
- ☐ 硬件：P4.3爲低電位。
- ☐ 硬件：P3.0 輸入兩個時脈。
- ☐ 軟件：可應用程序"LJMP"跳躍至ISP service program相對的起啓位置。

3.4 爲避免誤入 ISP，MCU 復位必須是 POR 或 PAD reset 才可以進入，若是 LVR 或 software reset 則不會進入 ISP。

3.5 爲保護功能選擇位元，只限定 MCU 是以硬件進入 ISP 才可以執行變更功能選擇位元，若以軟件進入 ISP(例如，LJMP 3E00h)則不可以變更功能選擇位元。

3.6 爲保護功能選擇位元，ISP 執行完成後，離開 ISP servicer program 時，建議使用 software reset(SFR 會 Reset)，不建議使用 LJMP 0000h 的方式(SFR 不會 Reset)。

3.7 ISP Command Table：

寫入位址	燒寫指令	IAP	ISP
code flash	byte program	O	O
	chip erase	O	O
	page erase	O	O
	chip protect	O	O
功能選擇位元	Write option	X	O
	Read option	O	O
	Erase option	X	O
	Finish flag*(註一)	X	O

註一：只有SM59R16G6/ SM59R09G6/ SM59R05G6有此功能。

3.8 共有六組重要的功能選擇位元提供使用，ISP 及 ICP/Writer 存取能力的比較，參考如下：59R

ISPFAL	功能選擇位元說明	ISP Read	ISP Write	ICP/Writer Read	ICP/Writer Write
0x03H	WDTEN.	O	O	O	O
0x04H	System clock select, FCLK [3:0]	O	O	O	O
0x05H	pin function select	O	O	O	O
0x06H	Internal reset time	O	O	O	O
0x07H	Lock bit N	O	X	O	O
0x08H	ISP entry mechanism select	O	O	O	O

3.8.1 看門狗功能致能(ISPFAL=0x03H)說明：

於程式中執行看門狗功能前需先將此特殊功能暫存器第7位元置低(0x03H，Bit 7=0)，此位元可使用ISP或ICP方式設定；於程式執行中亦可使用IAP之字節讀命令讀取此位元之狀態。

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTEN	-	-	-	-	-	-	-

WDTEN 0 is enable WDT
1 is disable WDT

3.8.2 時鐘來源選擇(ISPFAL=0x04H) 說明：

可使用ISP或ICP等方式設定時鐘來源選擇；於程式執行中亦可使用IAP之字節讀命令讀取時鐘來源之設定值。

System clock Select：

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	FCLK3	FCLK2	FCLK1	FCLK0

System clock select FCLK[3:0]	System clock
XXXX-0XXX	External crystal
XXXX-1000	24MHz from internal OSC
XXXX-1001	20MHz from internal OSC
XXXX-1010	16MHz from internal OSC
XXXX-1011	12MHz from internal OSC
XXXX-1100	8MHz from internal OSC

XXXX-1101	4MHz from internal OSC
XXXX-1110	2MHz from internal OSC
XXXX-1111	1MHz from internal OSC

3.8.3 P4[4:7] I/O功能選擇(ISPFAL=0x05H) 說明：

可使用ICP或ISP方式設定，將OCI_SCL、ALE、OCI_SDA and RESET 等I/O 定義成P4.4、P4.5、P4.6 and P4.7；於程式執行中亦可使用IAP之字節讀命令讀取I/O 之設定值。

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	RESET	OCI_SDA	ALE	OCI_SCL

RESET: 0 is bi-direction I/O pin (P4.7).
1 is reset input pin with 1000ns deglitch.
OCI_SDA: 0 is bi-direction I/O pin (P4.6).
1 is OCI_SDA pin.
ALE: 0 is bi-direction I/O pin (P4.5).
1 is ALE pin.
OCI_SCL: 0 is bi-direction I/O pin (P4.4).
1 is OCI_SCL pin.

各種封裝對應之PIN腳如下表：

	OCI_SCL/P4.4	ALE/P4.5	OCI_SDA/P4.6	RESET/P4.7
40-PIN PDIP	29	30	31	9
44-PIN PLCC	32	33	35	10
44-PIN PQFP	26	27	29	4
48-PIN LQFP	29	30	32	5

3.8.4 內建復位電路重置時間選擇(ISPFAL=0x06H) 說明：

可使用ICP或ISP方式選擇復位電路重置時間；於程式執行中亦可使用IAP之字節讀命令讀取復位電路重置時間之設定值。

Internal reset timer select：

Internal reset time select[7:0]	Reset time
XXXX-X111	25ms (default)
0000-0110	200ms
0000-0101	100ms
0000-0100	50ms

0000-0011	16ms
0000-0010	8ms
0000-0001	4ms
0000-0000	2ms

3.8.5 ISP服務程式區進入方式之設定(ISPFAL=0x08H) 說明：

可使用ICP或ISP方式選擇ISP服務程式區進入方式，於程式執行中亦可使用IAP之字節讀命令讀取ISP服務程式區進入方式之設定值，ISP服務程式區進入方式有四種，分別由旗標EMF1~4所記錄。

ISP entry mechanism select：

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Trigger source / ISP entry mechanism	晶片復位源 Internal Reset	晶片復位源 PAD reset	EMF (Entry mechanism flag)
(1) First Address Blank. I.e. $0000 = FFh	Bit 7	Bit 6	EMF1
(2) P2.6 = 0 & P2.7 = 0	Bit 5	Bit 4	EMF2
(3) P4.3 = 0	Bit 3	Bit 2	EMF3
(4) P3.0 input 2 clocks	Bit 1	Bit 0	EMF4

Bit 7: = 1, 禁能觸發(1)，禁能 ISP 由內建復位電路觸發(包括 POR 及 WDT)。
= 0, 致能觸發(1)，致能 ISP 由內建復位電路觸發(包括 POR 及 WDT)。
Bit 6: = 1, 禁能觸發(1)，禁能 ISP 由 PAD reset 觸發。
= 0, 致能觸發(1)，致能 ISP 由 PAD reset 觸發。
Bit 5: = 1, 禁能觸發(2)，禁能 ISP 由內建復位電路觸發(包括 POR 及 WDT)。
= 0, 致能觸發(2)，致能 ISP 由內建復位電路觸發(包括 POR 及 WDT)。
Bit 4: = 1, 禁能觸發(2)，禁能 ISP 由 PAD reset 觸發。
= 0, 致能觸發(2)，致能 ISP 由 PAD reset 觸發。
Bit 3: = 1, 禁能觸發(3)，禁能 ISP 由內建復位電路觸發(包括 POR 及 WDT)。
= 0, 致能觸發(3)，致能 ISP 由內建復位電路觸發(包括 POR 及 WDT)。
Bit 2: = 1, 禁能觸發(3)，禁能 ISP 由 PAD reset 觸發。
= 0, 致能觸發(3)，致能 ISP 由 PAD reset 觸發。
Bit 1: = 1, 禁能觸發(4)，禁能 ISP 由內建復位電路觸發(包括 POR 及 WDT)。
= 0, 致能觸發(4)，致能 ISP 由內建復位電路觸發(包括 POR 及 WDT)。
Bit 0: = 1, 禁能觸發(4)，禁能 ISP 由 PAD reset 觸發。
= 0, 致能觸發(4)，致能 ISP 由 PAD reset 觸發。

3.8.6 ISP Lock bit N(ISPFAL=0x07H) 說明：

只可由燒錄器或ICP才可以設定及抹除，ISP service program並無此指令。該位元是用來設定ISP service program的容量大小，且可保護ISP service program不被ISP or IAP所執行的”chip erase”抹除。lock bit ”N”設定ISP相對的起啓位置請參考如下：

ISP code area Table：

SM59R04A2/ SM59R04A1/ SM59R03A1/ SM59R02A1

N	ISP service program address
0	No ISP service program
1	256 bytes ($3F00h ~ $3FFFh)
2	512 bytes ($3E00h ~ $3FFFh)
3	768 bytes ($3D00h ~ $3FFFh)
4	1.0 K bytes ($3C00h ~ $3FFFh)
5	1.25 K bytes ($3B00h ~ $3FFFh)
6	1.5 K bytes ($3A00h ~ $3FFFh)
7	1.75 K bytes ($3900h ~ $3FFFh)
8	2.0 K bytes ($3800h ~ $3FFFh)
9	2.25 K bytes ($3700h ~ $3FFFh)
10	2.5 K bytes ($3600h ~ $3FFFh)
11	2.75 K bytes ($3500h ~ $3FFFh)
12	3.0 K bytes ($3400h ~ $3FFFh)
13	3.25 K bytes ($3300h ~ $3FFFh)
14	3.5 K bytes ($3200h ~ $3FFFh)
15	3.75 K bytes ($3100h ~ $3FFFh)
16	4.0 K bytes ($3000h ~ $3FFFh)

SM59R16A5/ SM59R16A3/ SM59R09A5/ SM59R09A3/ SM59R05A5/ SM59R05A3/ SM59R16G6/ SM59R09G6/ SM59R05G6

N	ISP service program address
0	No ISP service program
1	256 bytes ($FF00h ~ $FFFFh)
2	512 bytes ($FE00h ~ $FFFFh)
3	768 bytes ($FD00h ~ $FFFFh)
4	1.0 K bytes ($FC00h ~ $FFFFh)
5	1.25 K bytes ($FB00h ~ $FFFFh)
6	1.5 K bytes ($FA00h ~ $FFFFh)
7	1.75 K bytes ($F900h ~ $FFFFh)
8	2.0 K bytes ($F800h ~ $FFFFh)
9	2.25 K bytes ($F700h ~ $FFFFh)
10	2.5 K bytes ($F600h ~ $FFFFh)
11	2.75 K bytes ($F500h ~ $FFFFh)
12	3.0 K bytes ($F400h ~ $FFFFh)
13	3.25 K bytes ($F300h ~ $FFFFh)
14	3.5 K bytes ($F200h ~ $FFFFh)
15	3.75 K bytes ($F100h ~ $FFFFh)
16	4.0 K bytes ($F000h ~ $FFFFh)

3.9 ISP 相關的特殊暫存器 Special Function Register (SFR)

3.9.1 ISP 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	ITS	CDPR	F12K	F8K	ALEC.1	ALEC.0	EMEN	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SM59R16G6/ SM59R09G6/ SM59R05G6有此功能。當功能選擇位元(option)經由Erase及Write之後，最後必須下一次Finish flag指令以示完成，否則寫入值皆視爲無效。

3.10 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。 ***範例適用：*** ***SM59R16A5/ SM59R16A3/ SM59R09A5/ SM59R09A3/ SM59R05A5/ SM59R05A3/ SM59R04A2/ SM59R04A1/ SM59R03A1/ SM59R02A1/ SM59R16G6/ SM59R09G6/ SM59R05G6***

Main program	`//=====================================================================`
	`//`
	`//  S Y N C M O S  T E C H N O L O G Y`
	`//`
	`//=====================================================================`
	`#include "SM59R04A2.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     $
    jmp     Start

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
```

3.11 EEPROM C 語言範例程式：

Description	SyncMOS 原廠已有標準的 ISP code 及 AP 提供客戶使用；此範例展示有 chip erase, page erase, chip protect, byte write, byte read 等基本功能，提供使用者自行開發 ISP code 或 EEPROM 應用參考。 ***範例適用：*** ***SM59R16A5/ SM59R16A3/ SM59R09A5/ SM59R09A3/ SM59R05A5/ SM59R05A3/ SM59R04A2/ SM59R04A1/ SM59R03A1/ SM59R02A1/ SM59R16G6/ SM59R09G6/ SM59R05G6***

Main program	`//=====================================================================`
	`//`
	`//  S Y N C M O S  T E C H N O L O G Y`
	`//`
	`//=====================================================================`
	`#include "..\h\SM59R04A2.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`
	`#define d_Command_Write_Option  4`
	`#define d_Command_Read_Option   5`

```
#define d_Command_Erase_Option  6   //all

//=====================================================================
/*
ISPFAL must entry ISP by hardware;Lock bit as read only;
ISPFAL: Function:
0x03H   WDTEN.
0x04H   System clock select, FCLK [3:0]
0x05H   pin function select
0x06H   Internal reset time
0x07H   Lock bit N
0x08H   ISP entry mechanism select
*/
//=====================================================================

void Delay1mSec(unsigned int NTime)
{
  unsigned int i, j;

  for(i=0; i<NTime; i++)
    for(j=0; j<1300; j++);
}

void EEPROM_Enable(void)
{
    unsigned char ucRetry=255;
    while((IFCON&0x01)!=0x01)   // V105; go on if ISPE != 1
    {
        TAKEY  = 0x55;
        TAKEY  = 0xAA;
        TAKEY  = 0x5A;
        IFCON |= 0x01;          // ISPE=1, Enable ISP function

        ucRetry--;
        if(ucRetry<=0)  break;
    }
}

void EEPROM_Disable(void)
{
    unsigned char ucRetry   =255;
    while((IFCON&0x01)!=0x00)   // V105; go on if ISPE != 0
    {
        TAKEY  = 0x55;
        TAKEY  = 0xAA;
        TAKEY  = 0x5A;
        IFCON &= 0xFE;          // ISPE=0, Disable  ISP function

        ucRetry--;
        if(ucRetry<=0)  break;
    }
```

```
}

//void EEPROM_Byte_Program(unsigned char Addr_H, unsigned char Addr_L,
unsigned char Data )
void EEPROM_Byte_Program(unsigned int Addr, unsigned char Data)
{
    //unsigned char Temp =ISPFC;
    EEPROM_Enable();
    ISPFD   = Data;
    ISPFAH  = (Addr >>8);
    ISPFAL  = (Addr & 0x00FF);
    ISPFC   = d_Command_Byte_Program;
    EEPROM_Disable();
}

void EEPROM_Chip_Protect(void)
{
    EEPROM_Enable();
    ISPFC   = d_Command_Chip_Protect;
    EEPROM_Disable();
}

//void EEPROM_Page_Erase(unsigned char Addr_H, unsigned char Addr_L)
void EEPROM_Page_Erase(unsigned int Addr)       //256-Byte
{
    EEPROM_Enable();
    ISPFAH  = (unsigned char)(Addr >>8);
//  ISPFAL  = (unsigned char)(Addr & 0x00FF);
    ISPFC   = d_Command_Page_Erase;
    EEPROM_Disable();
}

void EEPROM_Chip_Erase(void)
{
    EEPROM_Enable();
    ISPFC   = d_Command_Chip_Erase;
    EEPROM_Disable();
}

void EEPROM Sector Program(unsigned int Addr start, unsigned int Addr end,
unsigned char Data)
{
    unsigned int i;
    for(i=0; i<(Addr_end - Addr_start); i++)
    {
    EEPROM_Byte_Program(Addr_start+i, Data);
    }
}

//unsigned char EEPROM_Byte_Read(unsigned char Addr_H, unsigned char Addr_L)
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;
}

void EEPROM_Byte_Modify(unsigned int Addr, unsigned char Data)
{
    unsigned int i;
    unsigned char xdata buf[256];
    for(i=0; i<256; i++)                    //page read
        buf[i] = EEPROM_Byte_Read((Addr&0xFF00)+i);

    EEPROM_Page_Erase(Addr&0xFF00);         //page erase

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

    for(i=0; i<256; i++)                    //page program
        EEPROM_Byte_Program((Addr&0xFF00)+i, buf[i]);
}

void main(void)
{
    unsigned char buffer=0;
    unsigned int Err=0, i=0;
    P0 = 0x55;
    Delay1mSec(100);    //wait for system stable
    P0 = 0xAA;

    EEPROM_Page_Erase(0x1000);
    EEPROM_Page_Erase(0x2000);

    for(i=0; i<256; i++)
    {
        EEPROM_Byte_Program( 0x1000+i, i );
        buffer = EEPROM_Byte_Read( 0x1000+i);
        if(buffer != i)
            Err++;

    }
    P0 = (unsigned char)(Err>>8);
    P1 = (unsigned char)(Err);

    P2 = ~P2;       Delay1mSec(500);
    P2 = ~P2;       Delay1mSec(500);

    EEPROM_Byte_Modify(0x2000, 0x66);
    buffer = EEPROM_Byte_Read(0x2000);
    P0 = buffer;
```

```
    EEPROM_Byte_Modify(0x2001, 0x88);
    buffer = EEPROM_Byte_Read(0x2001);
    P1 = buffer;

    while(1)
    {
        P2 = ~P2;       Delay1mSec(500);
        P2 = ~P2;       Delay1mSec(500);
    };
}
```

4 適用產品：SM59R16A2 / SM59R08A2

4.1 EEPROM 使用概述：

4.1.1 使用program flash模擬爲Internal EEPROM。

4.1.2 Page Erase及Program Byte的動作說明：

Page Erase 是將 flash 的電位全部提升至”1”，Program Byte 是將 flash 的電位”high”拉爲”low”。當程序執行一次 Program Byte，IC 內部硬體完整的步驟爲：(1)read page(2)modify Byte(3)erase page(4)write page；寫 N 個 byte，就必須經過 N 次(1~4)的步驟。

若少量次數的資料變動，可直接使用 EEPROM Program Byte 的方式。若大量的資料變動，爲加快 Program Byte 完成的速度，可先將 (1)資料 page read 至 SRAM 中做修改 (2)將該 program flash 做 page erase (3)將 SRAM 資料寫回至 program flash。

A. 爲什麼第一次會較快？

因爲當第一次使用時，flash 在 program 前已先被 erase 爲 0xFF，當 IC 內部硬體判斷該位址資料爲 0xFF，在 program 時就會省略(3)Erase 的步驟。

B. 爲什麼第二次以後會變慢？

因爲使用第二次以後，當 IC 內部硬體判斷不爲 0xFF,在 program 時就會做(3)Erase 的步驟。

Erase／Program／Read／Modify 測試速度：

System clock (MHz)	Page Erase (ms)	Program Byte (us)	Read Byte (us)	Modify Byte (ms)

25	13.22	112.2	3.12	68.44
24	13.77	116.8	3.25	71.3
22.1184	14.94	126.8	3.53	77.36
12	27.54	233.8	6.5	142.6
11.1592	29.88	253.6	7.05	156.6
Note: (1)Program Byte 即該位置原資料等於 0xFF (2)Modify Byte 即該位置原資料非 0xFF				

應用說明：

如果使用 External ISP，燒錄後 code flash 中的 data 如何保留？例如 Product ID、Key word 或 Menu setting …

應用方式如下：

1. 執行程序：read flash data，再將 data write to Xdata(SRAM)
2. 執行燒錄：External ISP program (使用 OCD or MSM9055)
3. 執行重置：Reset MCU (不可斷電)
4. 執行程序：read Xdata，再將 data write to flash，即可保存 flash 中的資料

	256	512	(Byte)
SRAM-->code flash	3.4	6.8	ms
code flash-->SRAM	3	6	ms
test with external crystal 22.1184MHz			

* 爲保持 SRAM 中的資料正確，執行過程請勿中斷電源。

* 此方法限制於內建 SRAM size，最大爲 2KB。

4.2 EEPROM 相關的暫存器：

Mnemonic	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
EEPROM Function											
PCON	Power Control	87h	SMOD	MDUF	-	PMW	-	-	STOP	IDLE	00h
PES	Program Memory Page Erase Control Register	A1h	EPE	-	-	-	-	-	-	-	00h

Mnemonic: PCON **Address: 87h**

7	6	5	4	3	2	1	0	Reset
SMOD	MDUF	-	PMW	-	-	STOP	IDLE	00h

PMW = 1: R / W Flash Memory enable
PMW = 0: disable

Mnemonic: PES **Address: A1h**

7	6	5	4	3	2	1	0	Reset
EPE	-	-	-	-	-	-	-	00h

EPE=1: 執行抹除 Page Erase (=512 bytes)，完成後由硬體自動清

4.3 執行 EEPROM page erase 抹除程序，請參考如下：

```
//=====================================================================
//This function will erase one page(512-Byte).
//SM59R16A2 64KB = 0~127 pages.
//SM59R08A2 32KB = 0~ 63 pages.
//=====================================================================
void PMEE_PageErase(unsigned int Address)
{
  unsigned char xdata *pAddr;
  PCON |=0x10; // R / W Flash Memory enable
  pAddr =Address;
  PES =0x80;   // Page erase enable, after finish auto clear by hard
  *pAddr=0xff; // Page erase command
  PCON = PCON & 0xEF; // R / W Flash Memory disable
}
```

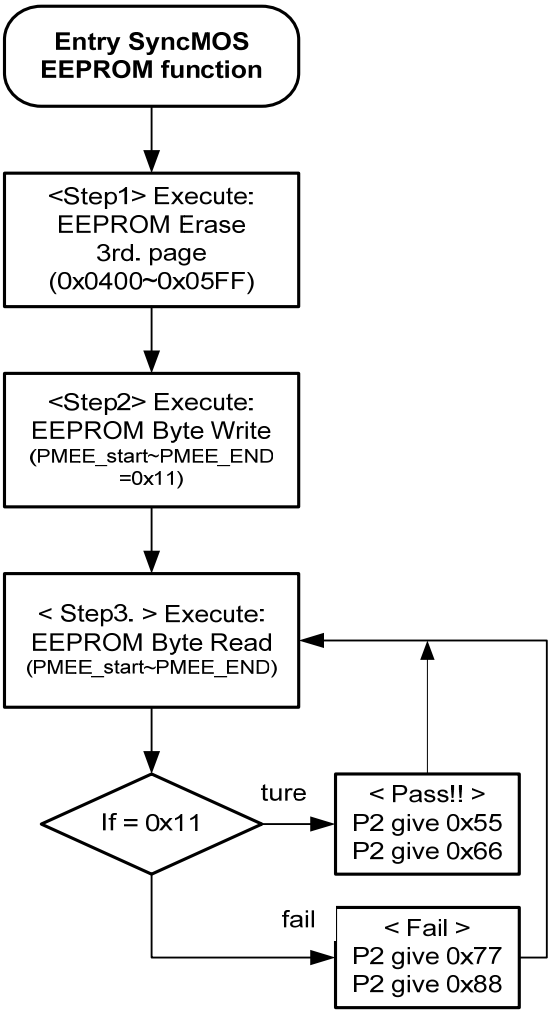
＊ EEPROM Erase 重要說明：

- **＊ page erase爲一硬體設定動作，執行程序順序請依照以上範例**
- **＊ page erase enable與command之間，絕不可插入其它程序**
- **＊ page erase enable後，必須在要抹除的flash page的第一個位置寫入”0xFFh”，當作page erase command，即可抹除。**

＊ ***Command address***如下例表：

SM59R16/08 page Erase		
Page	**Memory address**	**First byte address of page *(Command address)***
0	0x0000~0x01FF	0x0000
1	0x0200~0x03FF	0x0200
2	0x0400~0x05FF	0x0400
. **.** **.** **.**	. . . .	. . . .
126	0xFC00~0xFDFF	0xFC00
127	0xFE00~0xFFFF	0xFE00

4.4 範例程式流程圖（一）：EEPROM 的應用流程圖:



4.5 範例程式(一)：EEPROM 範例程式

Description	*範例適用：SM59R16A2/ SM59R08A2*
Main program	//========== // // S Y N C M O S T E C H N O L O G Y // //========== #include "..\h\SM59R16A2.h" #include "..\MISC\delay.h" #define PMEE_start 0X0400 #define PMEE_end 0x0600 //==========

```
//PES：Program Memory Page Erase Control Register
//When enable the EPE bit，the Page Erase（Each page include 512 bytes
//function can be executed by below Instructions?
//=====================================================================
sfr PES = 0xA1;
//sbit EPE  = PES^7;

//=====================================================================
//This function will erase one page(512-Byte).
//SM59R16A2 64KB = 0~127 pages.
//SM59R08A2 32KB = 0~ 63 pages.
//=====================================================================
void PMEE_PageErase(unsigned int Address)
{
    unsigned char xdata *pAddr;
    PCON |=0x10; // R / W Flash Memory enable
    pAddr =Address;
    PES =0x80;   // Page erase enable, after finish auto clear by hard
    *pAddr=0xff; // Page erase command
    PCON = PCON & 0xEF; // R / W Flash Memory disable
}

//=====================================================================
//This function will write one byte to EEPROM.
//Adderss is from 0x0000~ 0xFFFF.
//=====================================================================
void PMEE_ByteWrite(unsigned int Address, unsigned char Data)
{
    unsigned char xdata *pAddr;
    PCON    |= 0x10;        // R / W Flash Memory enable
    pAddr   = Address;      // Set address
    *pAddr  = Data;         // Write data
    PCON    = PCON & 0xEF;  // R / W Flash Memory disable
}

//=====================================================================
//This function will read one byte from EEPROM.
//Adderss is from 0x0000~ 0xFFFF.
//=====================================================================
unsigned char PMEE_ByteRead(unsigned int Address)
{
    unsigned char xdata *pAddr;
    unsigned char temp;
    PCON    |= 0x10;        // R / W Flash Memory enable
    pAddr   = Address;      // Set address
```

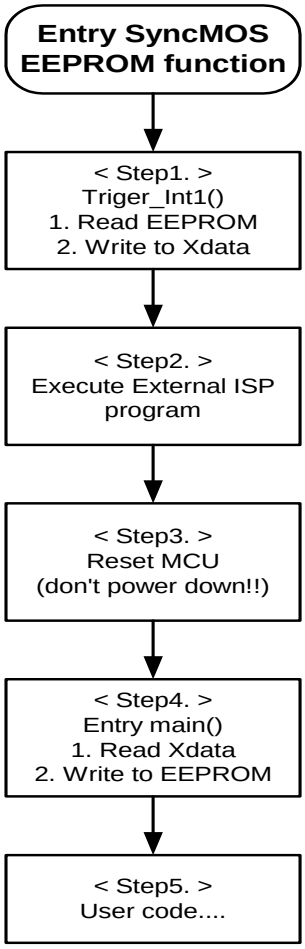
```
    temp    = (*pAddr);     // Read data
   PCON     = PCON & 0xEF;  // R / W Flash Memory disable
    return(temp);
}

void main(void)
{
    unsigned int i;

    PMEE_PageErase(0x0400);         //Erase 3rd. page (0x0400~0x05FF)
    for(i=PMEE_start; i<PMEE_end; i++)  //EEPROM Write
    {
        PMEE_ByteWrite(i, 0x11);
    }

    while(1)
    {
        for(i=PMEE_start;i<PMEE_end;i++)    //EEPROM Read
        {
            if(PMEE_ByteRead(i) == 0x11)    //EEPROM Verify
            {
                P2 = 0x55;  Delay1mSec(100);//Pass!!
                P2 = 0x66;  Delay1mSec(100);//Pass!!
            }
            else
            {
                P2 = 0x77;  Delay1mSec(200);//Fail Warning!!
                P2 = 0x88;  Delay1mSec(200);//Fail Warning!!
            }
        }
    }
}
```

4.6 範例程式流程圖（二）：EEPROM + SRAM 的應用流程圖:



4.7 範例程式(二)：EEPROM + SRAM 使用的範例程式

Description	
Main program	//=== // // S Y N C M O S T E C H N O L O G Y // //=== #include "..\h\SM59R16A2.h" #include "..\h\SM59R16A2_extradef.h" //#include "..\InternalEE\InternalEE.h" #include "..\Interrupt\Interrupt.h" #include "..\MISC\delay.h" #define Expanded_start 0X0600 #define Expanded_end 0x0700

```
unsigned char xdata *Expanded_RAM;
unsigned char xdata *PMEE_RWpt;

//====================================================================
//This function will write one byte to eeprom.
//Adderss is from 0x0000~ 0xFFFF.
//====================================================================
void PMEE_ByteWrite(unsigned int Address, unsigned char Data)
{
  //IEN0_EA = 0;                    // Disable all interrupt
  PCON      |= PCON_PROGMEM_WR;     // Enable eeprom write
  PMEE_RWpt = Address;              // Set address
  *PMEE_RWpt= Data;             // Write data
  PCON      &= (~PCON_PROGMEM_WR);  // Disable eeprom write
}

//====================================================================
//This function will read one byte from eeprom.
//Adderss is from 0x0000~ 0xFFFF.
//====================================================================
unsigned char PMEE_ByteRead(unsigned int Address)
{
  unsigned char     temp;
  //IEN0_EA = 0;                // Disable all interrupt
  PCON      |= PCON_PROGMEM_WR; // Enable eeprom write
  PMEE_RWpt = Address;          // Set address
  temp      = (*PMEE_RWpt);         // Read data
  PCON      &= (~PCON_PROGMEM_WR);  // Disable eeprom write
  return(temp);
}

void Expanded_RAM_Byte_Write(unsigned int addr, unsigned char dat)
{
    Expanded_RAM = addr;        //set addr
    *Expanded_RAM = dat;        //write Data
}

unsigned char Expanded_RAM_Byte_Read(unsigned int addr)
{
    unsigned char dat;
    Expanded_RAM = addr;        //set addr
    dat = *Expanded_RAM;        //read Data at
    return dat;
}
```

```
//====================================================================
// Read from eeprom, Write to Xdata. We want to keep Xdata
// so don't power down!!
//====================================================================
void Int1ISR(void) interrupt 2 using 0
{
    unsigned int i;
    unsigned char buff=0;

    for(i=Expanded_start; i<Expanded_end; i++)
    {
        buff = PMEE_ByteRead(i);            //Read from eeprom
        Expanded_RAM_Byte_Write( i , buff );//Write to Xdata
    }

    while(INT0IF)// Clear again to debounce.
    {
    INT1IF = false;
    Delay10mSec(1);
    }
}
void mcu_init(void)
{
    //Initialize External INT
    //Int0Init( EXINT0_EN, INT0_FALLEDGE );
    Int1Init( EXINT1_EN, INT1_FALLEDGE );
}

void main(void)
{
    unsigned int i;
    unsigned char buff=0;
    mcu_init();
    // Read from Xdata, Write to eeprom
    for(i=Expanded_start; i<Expanded_end; i++)
    {
        buff = Expanded_RAM_Byte_Read(i);   //Read from Xdata
        PMEE_ByteWrite( i , buff );         //Write to eeprom
    }
    while(1)
    {
    //user code...
    }
}
```