IIC 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 SM59R04A2/SM59R04A1/SM59R03A1/SM59R02A1

2 IIC 使用概述：

2.1 使用硬件 SCL (clock)及 SDA (data) 兩條線，必須接拉升電阻。

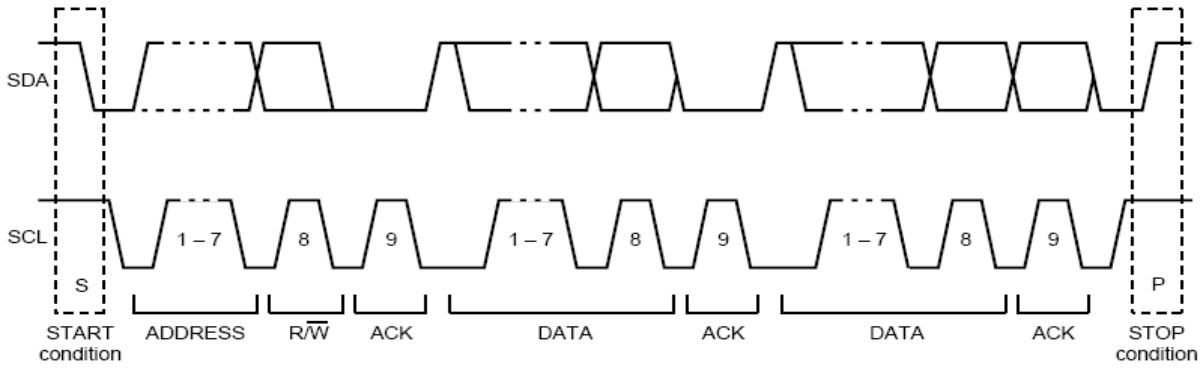
2.2 速度：SCL 最高可達 400Kbps(由軟件設定 SFR IICBR[2:0])。

2.3 IIC 可選擇 master 或 slave 兩種模式。

2.4 提供中斷(RXIF, TXIF)及兩組控制位址使用。

2.5 由軟件設定，Master 硬件可自動產生 IIC 的啓動(START)、重新啓動(Re-START)及停止(STOP)信號；Slave 硬件可偵測 IIC 的 START、Re-START 及 STOP 信號。

2.6 Slave 最多可接 127 devices，線材電容量最高限制爲 400pF。



3 P4IIC (IIC function pin assignment)說明：

	P4IIC (SPI function pin assignment)
SM59R16A5	○
SM59R09A5	○
SM59R05A5	○
SM59R16A3	○
SM59R09A3	○
SM59R05A3	○
SM59R16A2	×
SM59R08A2	×
SM59R04A2	○

SM59R04A1	○
SM59R03A1	○
SM59R02A1	○

○：表示該型號可以使用 P4IIC 功能
x：表示該型號不可使用 P4IIC 功能
Notice：所有系列 Package type DIP-40 皆沒有 P4，所以沒有用此功能

P4IIC (IIC function pin assign)說明

內建IIC，可直接經由**軟件**設定將其腳位指定至P1或P4，避免和其它特殊功能腳位重複，並提高**硬件**規劃的相容性。

Mnemonic	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
Serial interface 0 and 1											
AUX	Auxiliary register	91h	BRS	P4CC	P4SPI	P4UR1	***P4IIC***	P0KBI	-	DPS	00H

P4IIC: P4IIC = 0 – IIC function on P1.
P4IIC = 1 – IIC function on P4.

P4IIC	IIC_SCL	IIC_SDA
0	P1.6	P1.7
1	P4.0	P4.1

4 IIC 相關的特殊暫存器 SPI Special Function Register (SFR)

Mnemonic	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
IIC function											
IICCTL	IIC control register	F9h	IICEN	***BF***	MSS	MAS	RStart	IICBR[2:0]			04H
IICS	IIC status register	F8h	MStart	RXIF	TXIF	RDR	TDR	RXAK	TXAK	RW	00H
IICA1	IIC Address 1 register	FAh	IICA1[7:1]							MATCH1 or RW1	A0H
IICA2	IIC Address 2 register	FBh	IICA2[7:1]							MATCH2 or RW2	60H
IICRWD	IIC Read/Write register	FCh	IICRWD[7:0]								00H
IICS2	***IIC status2 register***	***FDh***	-	-	-	-	***AB_EN***	***BF_EN***	***AB_F***	***BF***	***00H***

Mnemonic: IICCTL **Address: F9h**

7	6	5	4	3	2	1	0	Reset
IICEN	*BF*	MSS	MAS	RStart	IICBR[2:0]			04h

IICEN: **IC 致能位元**(Enable IIC module)
IICEN = 1，IIC 致能。
IICEN = 0，IIC 禁能。

BF: ***(only SM59R16A2/SM59R08A2 used)***

IC 傳送失敗旗標 Bus failed flag (used in master mode only)：

當 Master 傳送資料"1"至 SDA，但 Master 偵測 SDA 爲"0"時，此旗標將被**硬件**設置爲"1" 。此旗標可由**軟件**清除。

MSS: **主從模式選擇位元 Master or slave mode select：**

MSS = 1，設定爲 master mode。

MSS = 0，設定爲 slave mode。

***要使用 IIC，在設定其它 IIC SFR 時，程序必須最先致能此位元。**

MAS: **控制位址位元 Master address select (master mode only)：**

MAS = 0 ，選擇控制位址(control byte)由 SFR IICA1 送出。

MAS = 1 ，選擇控制位址(control byte)由 SFR IICA2 送出。

RStart: **重新啓動位元 Re-start control bit (master mode only)：**

RStart = 0，在送出控制位址後旗標由硬件清除爲"0" 。

RStart = 1，由軟件設定，在收到 ACK 後，送出**啓動條件(Start condition)**及控制位址(控制位址位元由 MAS)。

IICBR[2:0]: **IIC 鮑率選擇元位 Baud rate selection (master mode only)：**

系統頻率(Fosc)依據 MCU 外部晶振(或內部晶振)而定，預設值爲 Fosc/512 。

IICBR[2:0]	Baud rate
000	Fosc/32
001	Fosc/64
010	Fosc/128
011	Fosc/256
100	Fosc/512
101	Fosc/1024
110	Fosc/2048
111	Fosc/4096

Mnemonic: IICS **Address: F8h**

7	6	5	4	3	2	1	0	Reset
MStart	RxIF	TxIF	RDR	TDR	RxAK	TxAK	RW	00h

MStart: **啓動位元 Master start control bit (master mode only)**

MStart = 1，由**軟件**設置爲"1"，送出**啓動條件(Start condition)**及控制位址位元(控制位址位元由 MAS)。

MStart = 0，由**軟件**清除爲"0"，送出**停止條件(Stop condition)**。

RxIF: **資料接收中斷旗標 Data receive interrupt flag**

Slave 由此旗標可判斷資料是否已接收

RxIF = 1，當資料讀寫暫存器(IICRWD)載入資料完成時，由**硬件**設爲"1"。

RxIF = 0，接收完成後必須由**軟件**清除。

TxIF: **資料傳送中斷旗標 Data transmit interrupt flag**
Master 由此旗標可判斷資料是否已傳出
TxIF = 1，當資料已由資料讀寫暫存器(IICRWD)載至位移暫存器，並已由位移暫存器傳送時，TxIF 爲"1"
TxIF = 0，傳送資料完成後必須由**軟件**清除。

RDR: **資料接收完成位元 Read data ready**
RDR = 1，IIC module 接收資料至資料讀寫暫存器(IICRWD)時，由**硬件**自動設定爲"1" 。
RDR = 0，當資料讀寫暫存器(IICRWD)完成接收後，必須由**軟件**清除爲"0"；當 RDR = 0 時，IIC module 才可再次寫入新的資料至資料讀寫暫存器(IICRWD)。

TDR: **資料傳送完成位元 Transmit data ready**
TDR = 1，由程序將資料寫入資料讀寫暫存器(IICRWD)後，由**軟件**設定此旗標爲"1" ，告知硬件 IIC module 將資料傳出。
TDR = 0，當IIC module由資料讀寫暫存器(IICRWD)讀取資料並完成傳送時，此位元則由**硬件**自動清除。

RxAK: **接收應答位元 Receive acknowledgement**
當 IIC module 傳送資料時此位元爲唯讀位元，等待接收端回覆應答(ACK/NACK)，不可由程序編寫。
當 IIC module 爲 master mode：傳出資料(8-bit)後，由 slave 回覆應答位元(RxAK)。
當 IIC module 爲 slave mode：傳出資料(8-bit)後，由 master 回覆應答位元(RxAK)。

TxAK: **傳送應答位元 Transmit acknowledgement**
IIC module 當接收端時，接收資料完成後，回覆傳送端應答位元(ACK/NACK)。

RW: **Slave mode read or write(read only)**
1. 當 IIC module 爲 slave mode 時，此位元爲唯讀，不可由程序編寫。
2. 此位元由 master 的位址 IICA1(或 IICA2) 的 8th-bit 所控制：
   = 0 : master 要求 slave 的 IIC module 爲接收模式(即 master write, slave read)。
   = 1 : master 要求 slave 的 IIC module 爲傳送模式(即 master read, slave write)。

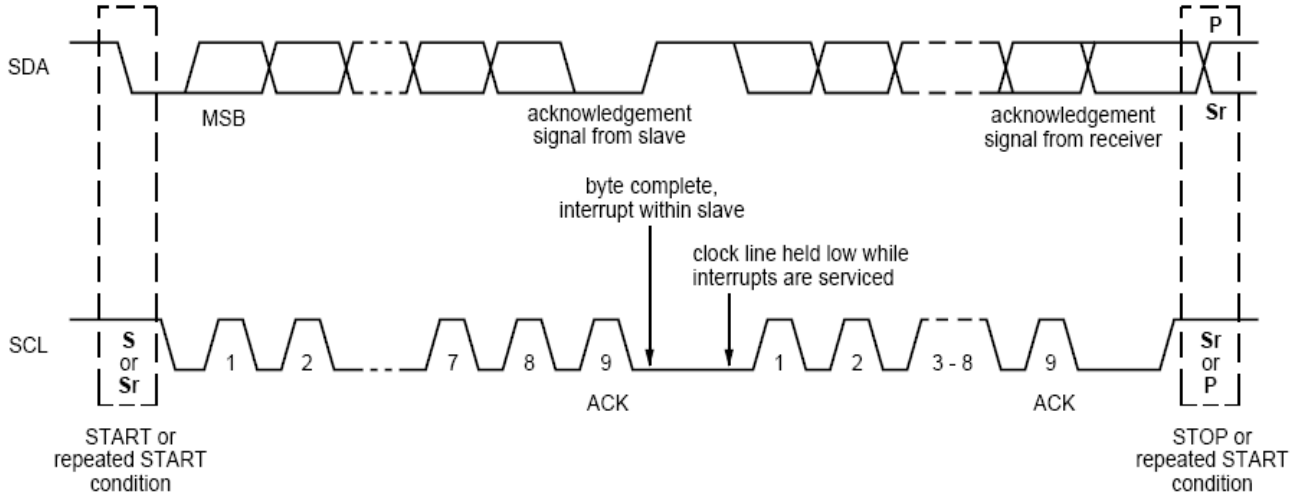


Fig. Acknowledgement bit in the 9th bit of a byte transmission

Mnemonic: IICA1 **Address: FAH**

7	6	5	4	3	2	1	0	Reset
IICA1[7:1]							Match1 or RW1	A0H
R/W							R or R/W	

Slave mode:

IICA1[7:1]: **IIC Address registers**

第一組控制位址 IICA1 共 7-bit，由軟件設定，當 slave 接收到 master 的**啓動條件(Start condition)**，及控制位址(IICA1)7-bit，兩者會相互比對，比較結果可參考 Match1。

Match1: = 1，當 slave 硬件偵測到**啓動條件(Start condition)**，且控制位址(IICA1)7-bit 相同時，slave 的 8th-bit(Match1)會由硬件設置爲"1" 。

= 0，當 slave 硬件(1) RXIF 發生但未偵測到**啓動條件(Start condition)**，或(2)TXIF 發生時，Match1 由硬件清除爲"0"。

Master mode:

IICA1[7:1]: **IIC Address registers**

1. 第一組控制位址 IICA1 共 7-bit，由軟件設定。
2. 當 MAS = 0，選擇控制位址(IICA1)。
   (當 MAS = 1，選擇控制位址(IICA2)。)
3. 當 MStart 由**軟件**設置爲"1"時，會送出**啓動條件(Start condition)**及控制位址位元(IICA1)。

RW1: 當 master 與 slave 的控制位址 7-bit 相同時，master 送出 8th-bit(R／W-bit)，告知 slave 讀寫的狀態

RW1= 1：爲 master IIC module 接收模式(即 master read, slave write)。

RW1= 0：爲 master IIC module 傳送模式(即 master write, slave read)。

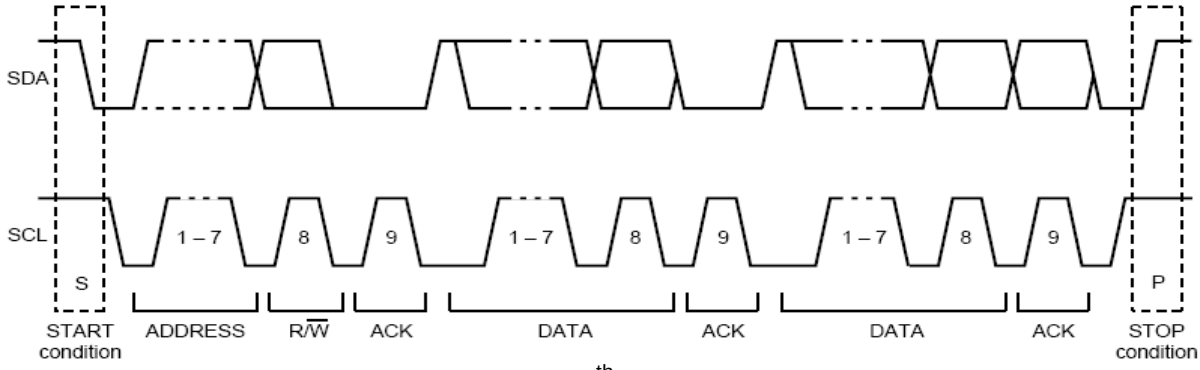


Fig. : RW bit in the 8th bit after IIC address

Mnemonic: IICA2 **Address: FBh**

7	6	5	4	3	2	1	0	Reset
IICA2[7:1]							Match2 or RW2	60h
R/W							R or R/W	

Slave mode:

IICA2[7:1]: **IIC Address registers**

第二組控制位址 IICA2 共 7-bit，由軟件設定，當 slave 接收到 master 的**啓動條件(Start condition)**，及控制位址(IICA2)7-bit，兩者會相互比對，比較結果可參考 Match2。

Match2: = 1，當 slave 硬件偵測到**啓動條件(Start condition)**，且控制位址(IICA2)7-bit 相同時，slave 的

8th-bit(Match2)會由硬件設置爲"1" 。
= 0，當 slave 硬件(1) RXIF 發生但未偵測到**啓動條件(Start condition)**，或(2)TXIF 發生時，Match2 由硬件清除爲"0"。

Master mode:

IICA2[7:1]: **IIC Address registers**

1. 第二組控制位址 IICA2 共 7-bit，由軟件設定。
2. 當 MAS = 1，選擇控制位址位元(IICA2)。
   (當 MAS = 0，選擇控制位址位元(IICA1)。)
3. 當 MStart 由**軟件**設置爲"1"時，會送出**啓動條件(Start condition)**及控制位址位元(IICA2)。

RW2: 當 master 與 slave 的控制位址 7-bit 相同時，master 送出 8th-bit(R／W-bit)，告知 slave 讀寫的狀態
RW1= 1：爲 master IIC module 接收模式(即 master read, slave write)。
RW1= 0：爲 master IIC module 傳送模式(即 master write, slave read)。

Mnemonic: IICRWD **Address: FCh**

7	6	5	4	3	2	1	0	Reset
IICRWD[7:0]								00h

IICRWD[7:0]: IIC 資料讀寫暫存器(8-bit) IIC read write data buffer：
IIC module 爲接收模式(讀取)時，爲接收資料的暫存區。
IIC module 爲傳送模式(寫入)時，爲傳送資料的暫存區。

The IICS2 register only SM59R16A5/SM59R09A5/SM59R05A5/SM59R16A3/SM59R09A3/SM59R05A3 and SM59R04A2/SM59R04A1/SM59R03A1/SM59R02A1 used.

Mnemonic: IICS2 ***Address: FDH***

7	6	5	4	3	2	1	0	Reset
-	-	-	-	AB_EN	BF_EN	AB_F	BF	00H

AB_EN: **仲裁失去了致能位元. (僅主機模式)Arbitration lost enable bit. (Master mode only)**
=1，致能，master硬件將檢查仲裁丟失位元AB_F，一旦發生丟失仲裁，AB_F將被設置爲"1"，即返回到閒置狀態。
=0，禁能，master硬件不會理會仲裁丟失情況。
當系統爲多主從(multi-master)應用時，請致能該偵測位元(AB_EN)。
當系統爲單主從(single-master)應用時，可禁能該偵測位元(AB_EN)。

BF_EN: **匯流排忙碌偵測致能位元 Bus busy enable bit. (Master mode only)**
=1，致能，master硬件無法送出啓動條件(Start condition)，直到BF=0。
=0，禁能，master硬件不理會BF=0或1。
當系統爲多主從(multi-master)應用時，請致能該偵測位元(BF_EN)。
當系統爲單主從(single-master)應用時，可禁能該偵測位元(BF_EN)。

AB_F: **仲裁丟失位元(僅主機模式) Arbitration lost bit. (Master mode only)**
=1，當master SDA送出"1"卻偵測回應爲"0"時，即發生丟失仲裁，AB_F將被設置爲"1"。 系統爲多主從(multi-master)應用，通訊時請偵測該位元。
=0，在重送資料前**軟件**需清除此位元，且需檢查BF直到已清除爲"0"。

BF: **匯流排忙碌偵測位元 Bus busy bit. (Master mode only)**
=1，當Master偵測到BUS上有SCL=0或SDA＝0或啓動條件(Start condition)時，該BF由**硬件**設爲"1"。系統爲多主從(multi-master)應用，通訊時請偵測該位元。
=0，當Master偵測到**停止條件(Stop condition)**，大約4.7us後，BF由**硬件**清除爲"0"。此位元也可由軟件清除爲"0"，使匯流排回到初始狀態。

5 IIC 中斷

5.1 向量表(Interrupt vectors table)

Interrupt Request Flags	Interrupt Vector Address	Interrupt Number *(use Keil C Tool)
IE0 – External interrupt 0	0003h	0
TF0 – Timer 0 interrupt	000Bh	1
IE1 – External interrupt 1	0013h	2
TF1 – Timer 1 interrupt	001Bh	3
RI0/TI0 – Serial channel 0 interrupt	0023h	4
TF2/EXF2 – Timer 2 interrupt	002Bh	5
PWMIF – PWM interrupt (The SM59R16A2/SM59R08A2 haven't)	0043h	8
SPIIF – SPI interrupt	004Bh	9
ADCIF – A/D converter interrupt	0053h	10
KBIIF – keyboard Interface interrupt	005Bh	11
LVIIF – Low Voltage Interrupt (The SM59R16A2/SM59R08A2 haven't)	0063h	12
IICIF – IIC interrupt	***006Bh***	***13***
RI1/TI1 – Serial channel 1 interrupt	0083h	16
RTC/ALARM interrupt (Only SM59R16A5/SM59R09A5/SM59R05A5 have)	008Bh	17
Comparator interrupt (Only SM59R16A5/SM59R09A5/SM59R05A5 have)	0093h	18

***See Keil C about C51 User's Guide about Interrupt Function description**

5.2 中斷相關暫存器(Interrupt SFR)

Mnemonic	Description	Direct	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET
Interrupt											
IEN0	Interrupt Enable 0 register	A8h	***EA***	-	ET2	ES0	ET1	EX1	ET0	EX0	00h
IEN1	Interrupt Enable 1 register	B8h	EXEN2	-	***IEIIC***	-	IEEEI	IEADC	IESPI	-	00h
IEN2	Interrupt Enable 2 register	9Ah	-	-	-	-	-	-	-	ES1	00h
IP0	Interrupt priority level 0	A9h	-	-	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0	00h
IP1	Interrupt priority level 1	B9h	-	-	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0	00h

IIC 中斷應用可參考以下設定：

(1) IIC 中斷致能設定：

```
    IEN0 |= 0x80;                    //Enable interrupt All
    IEN1 |= 0x20;                    //Enable interrupt IIC
```

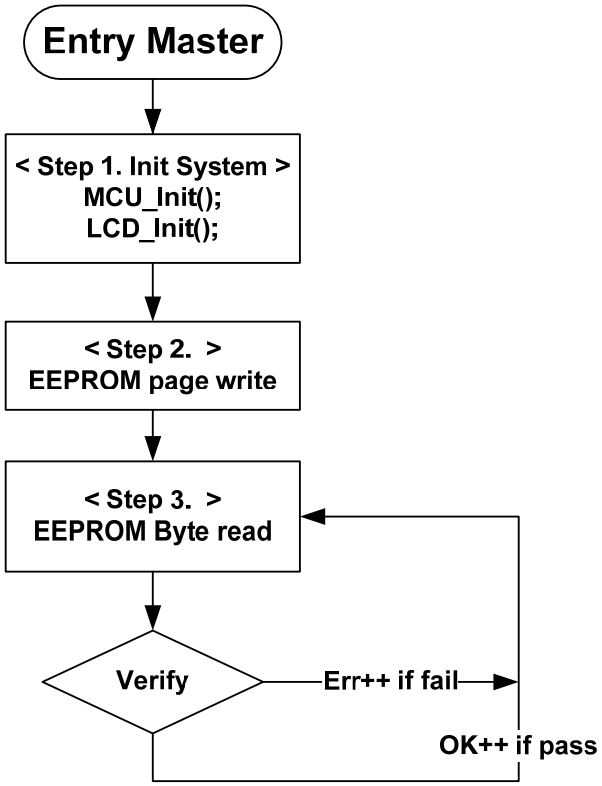
(2) IIC 中斷程序表示：

```
void IIC_interrupt(void) interrupt 13
{
    if(IICS_TXIF)
    {
        TxInt = 1;
        IICS_TXIF = 0;//Clear interrupt flag
    }
    if(IICS_RXIF)
    {
        RxInt = 1;
        IICS_RXIF = 0;//Clear interrupt flag
    }
}
```

Ps. IIC 中斷發生，必須清除 IICS^TXIF 或 IICS^RXIF，而非 IRCON 的 IICIF！（中斷應用可參考範例程式二）

6 MCU 爲 master，EEPROM 爲 slave 通訊應用的流程圖(二)：

6.1 Demo Board(SDB0001B)上的 EEPROM 爲 24LC04 (4KEEPROM，512x8bits，32pages，16bytes/page，詳細規格書可上網下載)。
6.2 MCU 對 EEPROM 做 page write (256bytes = 16bytes x 16page)，再用 random read 做 data verify 確認



7 MCU 對 EEPROM 24LC04 系列應用的範例程式(二)

Description	MCU master w/r IIC-EEPROM:

Main program	//==========
	//INCLUDE FILES
	//==========
	#include "..\h\SM59R16A2.h"
	#include "..\h\SM59R16A2_ExtraDef.h"
	
	#include "..\LCD\LCD16x2.h"
	#include "..\MISC\Delay.h"
	
	//==========
	//DEFINITIONs
	//==========
	#define IICSendStop() IICS_MStart = false
	#define NACK 1
	#define ACK 0
	#define Address_S 0x00 //for ICCEE verify
	#define Address_E 0xFF //for ICCEE verify
	#define NC 0 //Don't care
	//==========

```
//GLOBAL VARIABLES
//==============================
bit     IICEE_Error = false ;   //This bit is used to indicate if the read/write
is successful.
                                    // True/1: Error, False/0: No Error
bit TxInt = 0;
bit RxInt = 0;
unsigned int Count_OK=0, Count_Err=0;

unsigned char code  EDID_Data[]=
{
    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F,
    0X10, 0X11, 0X12, 0X13, 0X14, 0X15, 0X16, 0X17, 0X18, 0X19, 0X1A, 0X1B,
0X1C, 0X1D, 0X1E, 0X1F,
    0X20, 0X21, 0X22, 0X23, 0X24, 0X25, 0X26, 0X27, 0X28, 0X29, 0X2A, 0X2B,
0X2C, 0X2D, 0X2E, 0X2F,
    0X30, 0X31, 0X32, 0X33, 0X34, 0X35, 0X36, 0X37, 0X38, 0X39, 0X3A, 0X3B,
0X3C, 0X3D, 0X3E, 0X3F,
    0X40, 0X41, 0X42, 0X43, 0X44, 0X45, 0X46, 0X47, 0X48, 0X49, 0X4A, 0X4B,
0X4C, 0X4D, 0X4E, 0X4F,
    0X50, 0X51, 0X52, 0X53, 0X54, 0X55, 0X56, 0X57, 0X58, 0X59, 0X5A, 0X5B,
0X5C, 0X5D, 0X5E, 0X5F,
    0X60, 0X61, 0X62, 0X63, 0X64, 0X65, 0X66, 0X67, 0X68, 0X69, 0X6A, 0X6B,
0X6C, 0X6D, 0X6E, 0X6F,
    0X70, 0X71, 0X72, 0X73, 0X74, 0X75, 0X76, 0X77, 0X78, 0X79, 0X7A, 0X7B,
0X7C, 0X7D, 0X7E, 0X7F,
    0X80, 0X81, 0X82, 0X83, 0X84, 0X85, 0X86, 0X87, 0X88, 0X89, 0X8A, 0X8B,
0X8C, 0X8D, 0X8E, 0X8F,
    0X90, 0X91, 0X92, 0X93, 0X94, 0X95, 0X96, 0X97, 0X98, 0X99, 0X9A, 0X9B,
0X9C, 0X9D, 0X9E, 0X9F,
    0XA0, 0XA1, 0XA2, 0XA3, 0XA4, 0XA5, 0XA6, 0XA7, 0XA8, 0XA9, 0XAA, 0XAB,
0XAC, 0XAD, 0XAE, 0XAF,
    0XB0, 0XB1, 0XB2, 0XB3, 0XB4, 0XB5, 0XB6, 0XB7, 0XB8, 0XB9, 0XBA, 0XBB,
0XBC, 0XBD, 0XBE, 0XBF,
    0XC0, 0XC1, 0XC2, 0XC3, 0XC4, 0XC5, 0XC6, 0XC7, 0XC8, 0XC9, 0XCA, 0XCB,
0XCC, 0XCD, 0XCE, 0XCF,
    0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB,
0xDC, 0xDD, 0xDE, 0xDF,
    0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7, 0xE8, 0xE9, 0xEA, 0xEB,
0xEC, 0xED, 0xEE, 0xEF,
    0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7, 0xF8, 0xF9, 0xFA, 0xFB,
0xFC, 0xFD, 0xFE, 0xFF
};
//=============================================================
//OWNED SUBROUTINES
//=============================================================
void OSD(unsigned char SW, unsigned int D0, unsigned int D1)
{
    switch( SW)
    {
        case 0:
            PrintLcdStrLX(1, 0, "IIC demo:24xx EE");
            break;
        case 1:
            PrintLcdStrLX(2, 0, "Writing...     ");
            SetCursorAddr(2, 13);
```

```
            PrintLcdDec(D0);
            break;
        case 2:
            PrintLcdStrLX(2, 0, "Done.         ");
            Delay10mSec(50);
            break;
        case 3:
            PrintLcdStrLX(1, 0, "Verifying...   ");
            PrintLcdStrLX(1, 12, "   ");
            SetCursorAddr(1, 13);
            PrintLcdDec(D0);
            break;
        case 4:
            PrintLcdStrLX(2, 0, "OK:    Er:    ");
            SetCursorAddr(2, 3);
            PrintLcdDec(D0);
            SetCursorAddr(2, 11);
            PrintLcdDec(D1);
            Delay10mSec(50);
            break;
        case 5:
            break;
        case 6:
            break;
        case 7:
            break;
        case 8:
            break;
        case 9:
            break;
        case 10:
            break;
        default:
            break;
    }
}

void IIC_Init(unsigned char IICSetting)
{
    IICCTL  = IICSetting ;
}

void MCU_Init()
{
//  IIC_Init(IIC_EN|IIC_MASTER|IIC_ADR_CA1|IIC_BR4096); // Init BR
    IIC_Init(IIC_EN|IIC_MASTER|IIC_ADR_CA1|IIC_BR32);   // Init BR
    IEN0 = 0x80;        //Enable interrupt All
    IEN1 = 0x20;        //Enable interrupt IIC
}

void IICDisable(void)
{
    IICCTL  = 0x04 ;            // 0x04 is MCU reset value
}

//=============================================================
//This function will send out the START pulse and CONTROL byte.
```

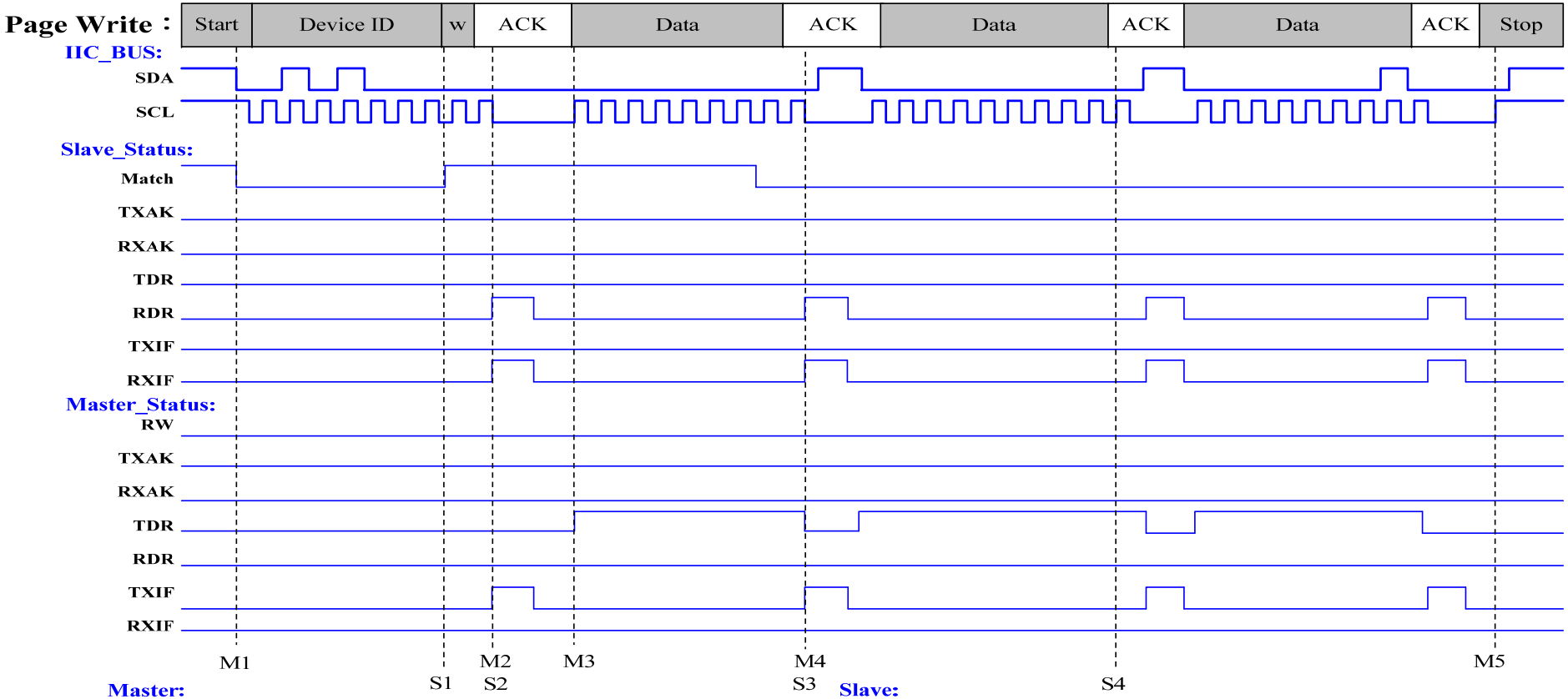
```
//=============================================================
void IIC_SendStart(unsigned char ControlByte)
{
    if( IICCTL & MSK_IICCTL_MAS )
        IICA2   = ControlByte;          // MAS = 1
    else
        IICA1   = ControlByte;          // MAS = 0

    IICS_MStart = true;

    while(~TxInt);
    TxInt = 0;
}

//=============================================================
//This function will send out the RESTART pulse and CONTROL byte.
//=============================================================
void IIC_SendRestart(unsigned char ControlByte)
{
  if( IICCTL & MSK_IICCTL_MAS )
    IICA2   = ControlByte;          // MAS = 1
  else
    IICA1   = ControlByte;          // MAS = 0

    IICCTL |= IIC_RSTART;

    while(~TxInt);
    TxInt = 0;
}

//=============================================================
//This function will wait and receive one byte, then feedback ACK or NACK.
//=============================================================
unsigned char IIC_ReceiveByte(bit ACKStatus)
{
    unsigned char   temp;
    unsigned int    Counter = 0;
    // Feedback ACK/NACK to slave for continue/stopping the transaction soon.
    IICS_TXAK   = ACKStatus;

    while(~RxInt);
    RxInt = 0;
    temp        = IICRWD;
    IICS_RDR    = 0;            // Clear Read Data Ready bit

  return(temp);
}

void IIC_TransmitByte(unsigned char TxData)
{
    IICRWD  = TxData;
    IICS_TDR    = 1;
    while(~TxInt);
    TxInt = 0;
}

unsigned char IICEE_RandomRead(unsigned char Block, unsigned char Address)
```

```
{
    unsigned char temp;
    // -- Send CONTROL byte --
    IIC_SendStart( 0xA0|Block );            // Set CONTROL byte to be write
operation

    // -- Send ADDRESS --
    IIC_TransmitByte(Address);

    // -- Send CONTROL byte --
    IIC_SendRestart( 0xA1|Block );      // Set CONTROL byte to be read
operation.

    // -- Read one byte --
    temp    = IIC_ReceiveByte(NACK);        // Wait for one byte

    // -- Send STOP --
    IICSendStop();

    return(temp);
}

void IICEE_ByteWrite(unsigned char Block, unsigned char Address, unsigned char
WriteData)
{
    // -- Send CONTROL byte --
    IIC_SendStart( 0xA0|Block );        // Set CONTROL byte to be write
operation

    // -- Send ADDRESS --
    IIC_TransmitByte(Address);

    // -- Write one byte --
    IIC_TransmitByte(WriteData);

    // -- Send STOP --
    IICSendStop();
}

void IICEE_Page_Write(unsigned char Block, unsigned char Page_sel )
{
    unsigned int i=0;
    unsigned int Address_Page_S=0;
    Address_Page_S= Page_sel<<4;

    // -- Send CONTROL byte --
    IIC_SendStart( 0xA0|Block );    // Set CONTROL byte to be write operation

    // -- Send ADDRESS --
    IIC_TransmitByte( Address_Page_S ); Delay10mSec(1);

    // -- Write N bytes --
    for (i=Address_Page_S; i<(Address_Page_S+16); i++)  //16byte per 1page
    {
        OSD(1, EDID_Data[i], NC);
        IIC_TransmitByte( EDID_Data[i]);    // Transmit one byte
        Delay10mSec(1);
```

```
    }
    // -- Send STOP --
    IICSendStop();
}

void IICEE_Byte_Read()
{
    unsigned int i, temp ;
    for (i=Address_S; i<=Address_E; i++)
    {
        //PrintLcdDec(i);
        temp = IICEE_RandomRead(0, i);  //read
        OSD(3, temp, NC);
        P2 = temp;

        if( temp == EDID_Data[i] )      //verify
            Count_OK++;
        else
        {
            Count_Err++;
        }
        Delay10mSec(1);         // For display stable.
    }
    OSD(4, Count_OK, Count_Err);
}

void IIC_interrupt(void) interrupt 13 using 1
{
    if(IICS_TXIF)
    {
        TxInt = 1;
        IICS_TXIF = 0;//Clear interrupt flag
    }
    if(IICS_RXIF)
    {
        RxInt = 1;
        IICS_RXIF = 0;//Clear interrupt flag
    }
}

void main()
{
    unsigned char i=0;
    P2 = 0x00;          // P2 is 7 Segment. for test only
    MCU_Init();         //Init MCU
    LCD_Init();         //Init LCD
    Delay10mSec(5);
    OSD(0, NC, NC);     //On screen display

    for (i=0; i<=15; i++)
    {
        IICEE_Page_Write(0, i); //Block=0, page=0~15
    }
    OSD(2, NC, NC);

    while(1)
    {
```

```
    IICEE_Byte_Read();
    }
}
```

8 通訊協定以標準的 **IIC EEPROM 24Cxx** 爲範例，**Master** 和 **Slave** 皆使用 **SM59R16A5**，示範 **page write** 及 **random ead** 之間如何溝通及控制，以下是 **Master** 及 **Slave** 溝通時的暫存器狀態及說明。



Master:

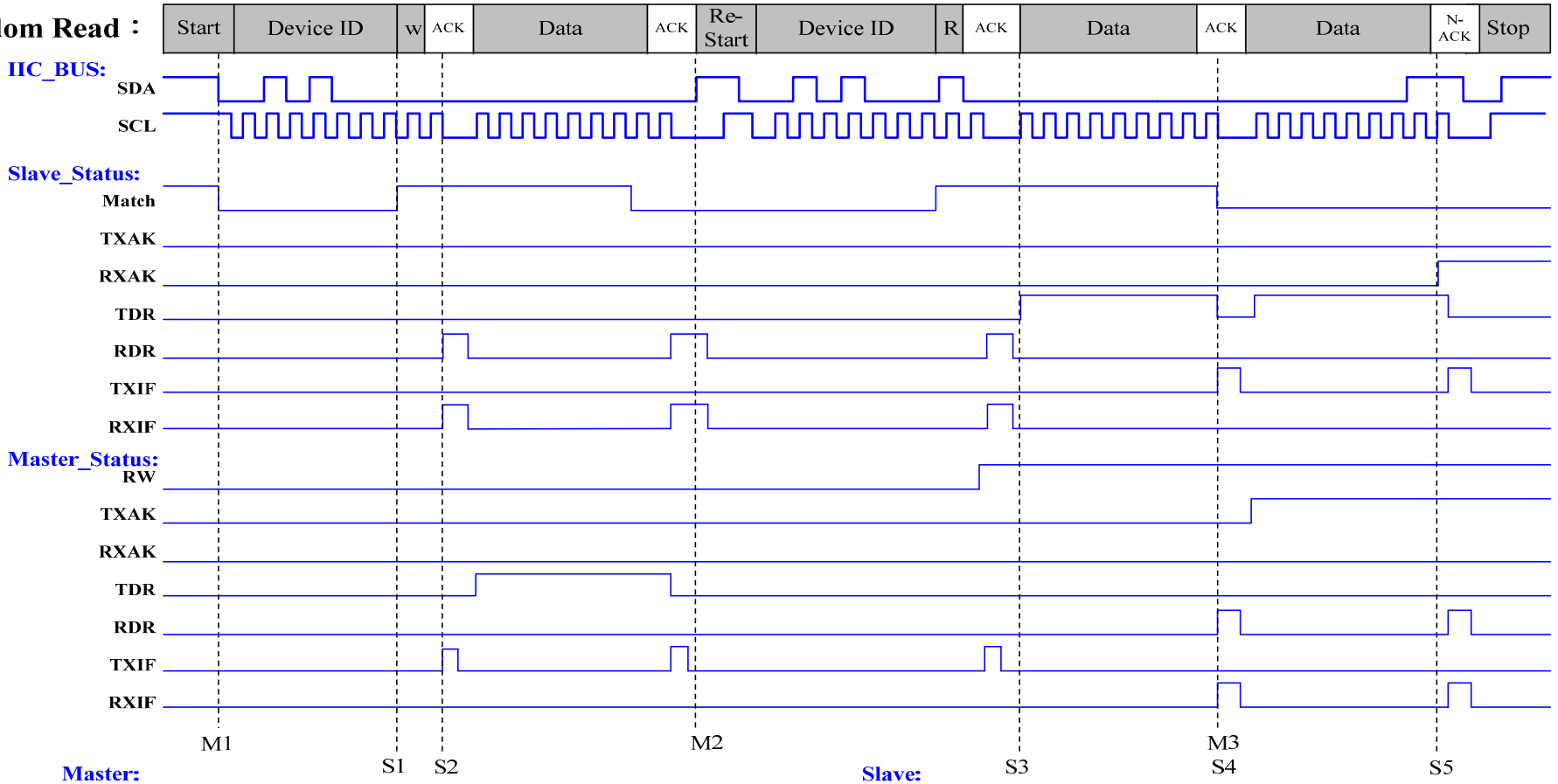
M1. 當IICS^Mstart由軟件設定爲1，Master會送出Start、Device ID及RW訊號。
M2. Start及Device ID送出完成時，TXIF由硬體設置爲1，由軟體清除。
M3. 當Data寫入IICRWD後，TDR由軟件設爲1，即會將Data送出。
M4. Data完成送出後，TDR由硬件清除爲0，TXIF由軟件設置爲1。
M5. 當IICS^Mstart=1由軟件清除爲0，即送出Stop訊號。

Slave:

S1. 當Slave硬件偵測到起始訊號(Start condition)，且控制位址IICA1(or IICA2)相同時，Match1(or Match2)由硬件設爲1，但收到的Device ID並不會載入IICRWD。
S2. 每次當收完Device ID(及RW)或Data時，都必須回傳ACK；RXIF及RDR由硬件設置爲"1"，必須由軟件清除。
S3. 當Data載入IICRWD時，RXIF及RDR由硬件設置爲"1"，必須由軟件清除。

IIC 功能使用方法

Random Read：



Master:

M1. 當IICS^Mstart由軟件設定爲1，Master會送出Start、Device ID及RW訊號。

M2. 當IICCTL^RStartt由軟件設定爲1，Master會送出Re-Start、Device ID及RW訊號。

M3. 每次Data載入IICRWD時，RXIF及RDR由硬件設置爲"1"，必須由軟件清除。

Slave:

S1. 當Slave硬件偵測到起始訊號(Start condition)，且控制位址IICA1(or IICA2)相同時，Match1(or Match2)由硬件設爲1，但收到的Device ID並不會載入IICRWD。

S2. 每次當收完Device ID(及RW)或Data時，都必須回傳ACK；RXIF及RDR由硬件設置爲"1"，必須由軟件清除。

S3. 每次Data寫入IICRWD後，TDR由軟件設爲1，即會將Data送出。

S4. 每次Data完成送出後，TDR由硬件清除爲0，TXIF由硬體設置爲1，TXIF須由軟體清除爲0。

S5. 當收到N-ACK時，Slave即停止Data回傳。

9 通訊協定以標準的 IIC EEPROM 24Cxx，使用 SM59R16A5 當 Master 和 Slave 的範例程序：

Description

-- MCU IIC master --

1. 以下定義有註明"user modify"，是 user 可搭配 slave 自行變更修改，其它未註明的不建議修改。
2. 一般模式中，Device ID 應盡量避免使用 0xA0(EEPROM 常用)和 0x00(general call address 使用)，或其它 IIC 規範中特定保留的部分。
3. 範例使用 IIC 資料存取最常使用的通訊協定，page write 及 random read，可直接與 slave 溝通。

Main program

```
//=====================================================================
//
//              S Y N C M O S   T E C H N O L O G Y
//
//=====================================================================
//IIC Master sample code, "user modify" as fallow, that is allowed for
//modify. Others are not suggest or illegal.
//==============================
//INCLUDE FILES
//==============================
#include "SM59R16A5.h"
#include "IIC.h"

//==============================
//IIC DEFINITIONs
//==============================
#define d_IIC_EN            0x80
#define d_DEVICE_ID         0x50            //user modify
//#define   d_DEVICE_ID     0xA0            //user modify

#define d_IIC_MASTER        0x20
#define d_IIC_SLAVE         0x00
#define d_IIC_MODE_SEL      d_IIC_MASTER    //user modify

#define d_IIC_ADR_CA1       0x00
#define d_IIC_ADR_CA2       0x10
#define d_MSK_IICCTL_MAS    0x10
#define d_IIC_ADR_SEL       d_IIC_ADR_CA2   //user modify

#define d_BR32              0x00
#define d_BR64              0x01
#define d_BR128             0x02
#define d_BR256             0x03
#define d_BR512             0x04
#define d_BR1024            0x05
#define d_BR2048            0x06
#define d_BR4096            0x07
#define d_IIC_BR            d_BR32          //user modify

#define d_NACK              1
#define d_ACK               0
#define d_WRITE             0
#define d_READ              1
#define IIC_Start()         MStart   = 1
#define IIC_Restart()       IICCTL  |= 0x08
#define IIC_SendStop()      MStart   = 0
//==============================
//Code Data
//==============================
unsigned char code  EDID_Data[]=
{
    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F
};

//=============================================================
```

```
//OWNED SUBROUTINES
//============================================================
void Delay10uSec(unsigned int NTime)
{
  unsigned int       i, j;
  for(i=0; i<NTime; i++)
        for(j=0; j<18; j++);
}

void IIC_interrupt(void) interrupt d_IIC_Vector
{
    if(TXIF)
    {
        TXIF = 0;                    //Clear interrupt flag
    }

    if(RXIF)
    {
    }
}

void IIC_Init(unsigned char IICSetting)
{
    IICA1   = 0x00;
    IICA2   = 0x50;
    IICCTL  = IICSetting ;
    IEN1    = 0x20;              // Enable interrupt IIC
    IICS2   = 0x0C;                  // Enable IIC AB_EN & BF_EN
}

void MCU_Init()
{
    IFCON   |= 0x80;                 // MCU 1T
    IEN0     = 0x80;                 // Enable interrupt All
    IIC_Init(d_IIC_EN | d_IIC_MODE_SEL | d_IIC_ADR_SEL | d_IIC_BR); // Init
BR
}

void IIC_Disable(void)
{
    IEN1    &= 0xDF;                 // Disable interrupt IIC
    IICCTL   = 0x00;             // Disable IIC all function
    IICS     = 0x00;             // Clear IIC all status
    IICS2    = 0x00;             // Clear IIC all status
}

//============================================================
//This function will send out the START pulse and CONTROL byte.
//============================================================
void IIC_SendStart(unsigned char ControlByte)
{
    if( IICCTL & d_MSK_IICCTL_MAS )
        IICA2   = ControlByte;       // MAS = 1
    else
        IICA1   = ControlByte;       // MAS = 0

    while(TDR);                      // wait last data trans. finish
    IIC_Start();                     // MStart    = 1
}

//============================================================
//This function will send out the RESTART pulse and CONTROL byte.
//============================================================
void IIC SendRestart(unsigned char ControlByte)
```

```
{
    while(TDR);                         // wait last data trans. finish
    if( IICCTL & d_MSK_IICCTL_MAS )
        IICA2   = ControlByte;          // MAS = 1
    else
        IICA1   = ControlByte;          // MAS = 0

    IIC_Restart();                      // IICCTL   |= 0x08;
}

//=============================================================
//This function will wait and receive one byte, then feedback ACK or NACK.
//=============================================================
unsigned char IIC_ReceiveByte(bit ACKStatus)
{
    unsigned char   temp;
    TXAK    = ACKStatus;                // Feedback ACK/d_NACK to slave

    while(~RXIF){};
    RXIF=0;

    temp    = IICRWD;
    RDR     = 0;                        // Clear Read Data Ready bit

    return(temp);
}

void IIC_TransmitByte(unsigned char TxData)
{
    while(TDR);                         // wait last data trans. finish
    IICRWD  = TxData;
    TDR = 1;
}

void IIC_Page_Write( char Crtl_byte, char Addr, char LEN)
{
    unsigned char counter=0;

    IIC_SendStart( Crtl_byte );             // -- Send CONTROL byte --
    IIC_TransmitByte( Addr );               // -- Send ADDRESS --

    for (counter=0; counter<LEN; counter++)         // -- Write N bytes --
    {
        IIC_TransmitByte( EDID_Data[Addr + counter]);// -- Transmit one byte
--

    }
    while(TDR);                 // must wait last data trans. finish
    IIC_SendStop();             // -- Send STOP -- MStart= 0
}

void IIC_Random_Read( char Crtl_byte, char Addr, char LEN)
{
    unsigned char Temp[16];
    unsigned char counter=0;

    IIC_SendStart( Crtl_byte | d_WRITE );   // -- Send CONTROL byte --
    IIC_TransmitByte(Addr + counter);       // -- Send ADDRESS --

    IIC_SendRestart(Crtl_byte | d_READ);    // -- Send Restart --
    for(counter=0; counter<LEN; counter++)  // -- Read N byte --
    {
        if(counter<(LEN-1))
        {
```

```
            Temp[counter] = IIC_ReceiveByte(d_ACK); // Receive one byte
        }
        else
        {
            Temp[counter] = IIC_ReceiveByte(d_NACK);// Receive last byte
        }
    }
    IIC_SendStop();             // -- Send STOP -- MStart= 0
}

void Check_Arbitration(void)    // multi-master use
{
    while(IICS2&0x02)
    {
        IICS2 &= 0xFD;
        Delay10uSec(1);
    }
}

void Check_Busy(void)           // multi-master use
{
    while(IICS2&0x01)
    {
        IICS2 &= 0xFE;
        Delay10uSec(1);
    }
}

void main()
{
    MCU_Init();

    Check_Arbitration();                    // multi-master use
    Check_Busy();                           // multi-master use

    IIC_Page_Write(d_DEVICE_ID, 0x00, 16);  Delay10uSec(10);
    IIC_Random_Read(d_DEVICE_ID, 0x00, 8);  Delay10uSec(10);
    IIC_Random_Read(d_DEVICE_ID, 0x08, 8);  Delay10uSec(10);

    IIC_Disable();
    while(1)
    {
    }
}
```

Description	-- MCU IIC Slave --
Main program	

```
//====================================================================
//
//              S Y N C M O S   T E C H N O L O G Y
//
//====================================================================
#include "SM59R16A5.h"

//====================================================================
#define d_IIC_Adress_1  0x50    //user modify
#define d_IIC_Adress_2  0x00    //user modify

#define d_ACK           0
#define d_NACK          1
#define d_Write         0
#define d_Read          1
#define d_null          0x00
```

```
//=====================================================================
unsigned char n_RW      = d_null;
unsigned char n_Addr    = d_null;
unsigned char n_IIC_Step= d_null;
unsigned char n_DAT[16];

//=====================================================================
void IIC_interrupt(void) interrupt d_IIC_Vector
{

    if((IICA1&0x01)|(IICA2&0x01))
    {
        n_IIC_Step = d_null;        // Device ID match, step clear
    }

    if(RXIF)
    {
        RXIF = 0;
        RDR =0;
        n_IIC_Step++;

        switch (n_IIC_Step)
        {
            case 0:
                break;
            case 1:                 // match
                n_RW = RW;
                break;
            case 2:                 // write addr
                n_Addr = IICRWD;
                break;
            default:                // write data
                n_DAT[n_Addr++] = IICRWD;
                break;
        }
    }
    else if(TXIF)
    {
        TXIF    = 0;
    }

}

void IIC_init_slave(void)
{
    IICCTL  = 0x80;             //Enable IIC module
    IICS    = 0x00;             //init IICS
    IRCON   = 0x00;             //init IRCON
    IICA1   = d_IIC_Adress_1;   //Control Byte 1
    IICA2   = d_IIC_Adress_2;   //Control Byte 2
    IEN0    |= 0x80;            //Enable interrupt All
    IEN1    |= 0x20;            //Enable interrupt IIC

}

void main(void)
{
    IIC_init_slave();

    while(1)
    {

//=====================================================================
        if((n_RW==d_Read)&(n_IIC_Step>=1))  // slave TX start
```

```
        {
            if(RXAK == d_ACK)               // trans. data if recive ACK
            {
                IICRWD = n_DAT[n_Addr++];
                TDR=1;
                while(TDR){}
            }
            else if(RXAK == d_NACK)         // End if recive NACK
            {
                n_IIC_Step  = d_null;
                n_RW        = d_null;
            }

        }

//=====================================================================
    }//end of while(1)
}//end of main
```