HY16F18X

HYCON IP 使用說明書

目　錄

01.文件說明 3

02.晶片說明 3

03.數位 IP(Timer A and Timer B) 7

04.數位 IP(Timer C) 15

05.數位 IP(WDT) 18

06.數位 IP(PWM) 21

07.類比 IP(DAC) 24

08.類比 IP(OPAMP) 27

09.類比 IP(ADC) 30

10.類比 IP(CMP) 35

11.通訊 IP(SPI) 41

12.通訊 IP(UART) 49

13.通訊 IP(I2C) 54

14.其他 IP(GPIO) 62

15.其他 IP(Power) 65

16.數位 IP(RTC) 69

17.程式附件 73

18.修訂紀錄 73

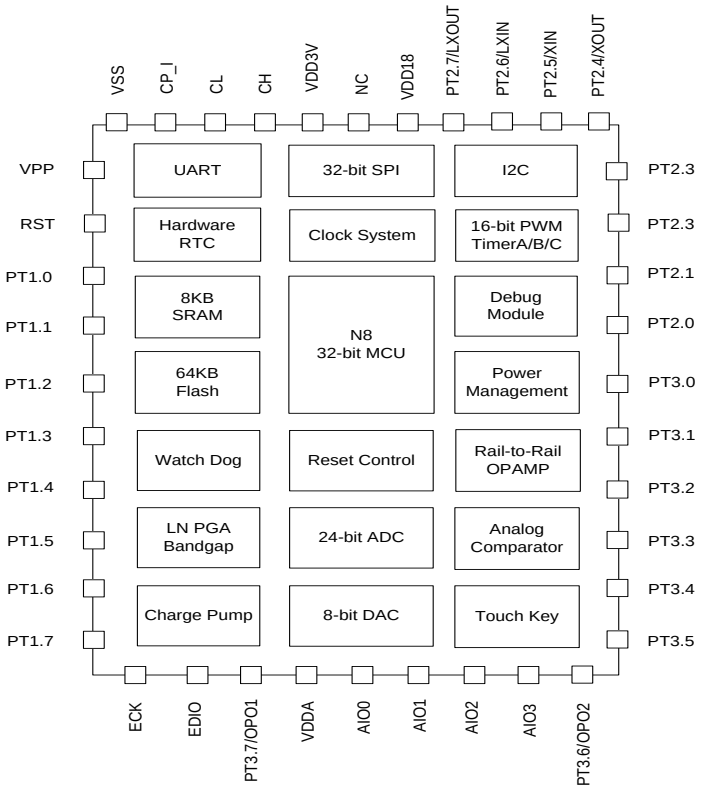
01.文件說明

HYCON IP(IP 矽智財 Intellectual Property 縮寫)，代表紘康 32 位元 MCU 內部各元件的(矽智財)半導體矽晶片智慧財產權。本文件針對 HY16F18X 系列，SOC 晶片內數位、類比及通訊等周邊 IP，做使用說明。主要包含以下 4 大分類:

(1)數位 IP：TimerA/TimerB/TimerC/WDT/PWM/Hardware RTC
(2)類比 IP：DAC/ADC/OPAMP/Analog CMP
(3)通訊 IP：Hardware 32-bit SPI/ Hardware UART/ Hardware I2C
(4)其他 IP：GPIO/Sleep/Idle

02.晶片說明

HY16F188 各功能 IP 基礎使用說明。



(01)採用晶心科技 Andes 最新 32 位元 CPU 核心 N801 處理器。

(02)電壓操作範圍 2.0~3.6V，以及-40℃~85℃工作溫度範圍。

(03)支援外部 20MHz 石英震盪器或內部 20MHz 高精緻 RC 震盪器，
擁有多種 CPU 工作時脈切換選擇，可讓使用者達到最佳省電規劃。

(3.1)運行模式 350uA@2MHz/2

(3.2)待機模式 10uA@35KHz/2

(3.3)休眠模式 2.5uA

(04)程式記憶體 64K-Byte Flash ROM

(05)資料記憶體 08K-Byte SRAM。

(06)擁有 BOR and WDT 功能，可防止 CPU 死機。

(07)24-bit 高精準度 ΣΔADC 類比數位轉換器

(7.1)內置 PGA (Programmable Gain Amplifier)最高可達 128 倍放大。

(7.2)內置溫度感測器 TPS。

(08)內建 1 組 OPA 運算放大器與 1 組類比比較器(具 16 段可程式化電阻)。

(09)內建硬體 8-bit DAC。

(10)16-bit Timer A

(11)16-bit Timer B 模組具 PWM 波形產生功能

(12)16-bit Timer C 模組具數位 Capture/Compare 功能

(13)硬體串列通訊 32-bit SPI/I2C/UART 模組

(14)硬體 RTC 時鐘功能模組

(15)硬體 Touch KEY 功能模組

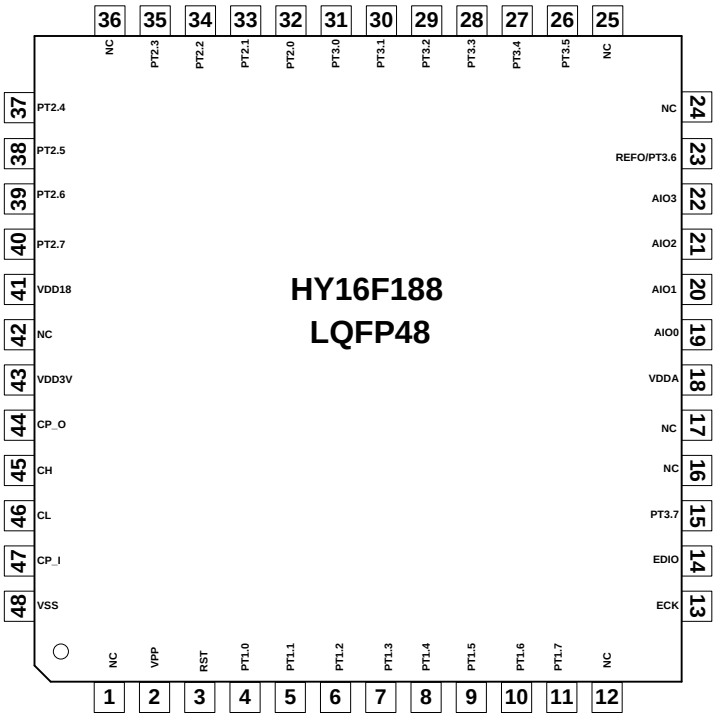
※中斷副程式型態

編號	中斷副程式宣告	說明	備註
HW0	void HW0_ISR(void)	通訊 IP	UART/I2C/32-bit SPI
HW1	void HW1_ISR(void)	數位 IP	Timer A/B/C & WDT/RTC
HW2	void HW2_ISR(void)	ADC IP	SD 24-bit ADC
HW3	void HW3_ISR(void)	CMP IP	Analog CMP/OPAMP
HW4	void HW4_ISR(void)	PT1 IP	PT1.0~PT1.7 可獨立或一起使用
HW5	void HW5_ISR(void)	PT2 IP	PT2.0~PT2.7 可獨立或一起使用

	數位 IP	類比 IP	通訊 IP	其他 IP
01	TimerA	8-bit DAC	Hardware 32-bit SPI	GPIO
02	TimerB	R2R OPAMP	Hardware UART	Sleep Mode
03	TimerC	24-bit SD ADC	Hardware I2C	Idle Mode

04	WDT	Analog CMP	-	-
05	PWM	-	-	-
06	Hardware RTC	-	-	-

GPIO Port	OSC	Interrupt	Timer C	SPI	IIC	UART	CMP	Analog	PWM
Priority	0	0	0	1	2	3	4	5	6
PT1.0		INT1.0	TCI1_1	CS_1	SCL_1	TX_1	CH1		PWM0_1
PT1.1		INT1.1	TCI2_1	CK_1	SDA_1	RX_1	CH2		PWM1_1
PT1.2		INT1.2	TCI1_2	MISO_1	SCL_2	TX_2	CH3		PWM0_2
PT1.3		INT1.3	TCI2_2	MOSI_1	SDA_2	RX_2	CL1		PWM1_2
PT1.4		INT1.4	TCI1_3	CS_2	SCL_3	TX_3	CL2		PWM0_3
PT1.5		INT1.5	TCI2_3	CK_2	SDA_3	RX_3	CL3		PWM1_3
PT1.6		INT1.6	TCI1_4	MISO_2	SCL_4	TX_4	CL4		PWM0_4
PT1.7		INT1.7	TCI2_4	MOSI_2	SDA_4	RX_4	CMPO1		PWM1_4
PT2.0		INT2.0	TCI1_5	CS_3	SCL_5	TX_5			PWM0_5
PT2.1		INT2.1	TCI2_5	CK_3	SDA_5	RX_5			PWM1_5
PT2.2		INT2.2	TCI1_6	MISO_3	SCL_6	TX_6			PWM0_6
PT2.3		INT2.3	TCI2_6	MOSI_3	SDA_6	RX_6			PWM1_6
PT2.4	LSXT1	INT2.4	TCI1_7	CS_4	SCL_7	TX_7			PWM0_7
PT2.5	LSXT2	INT2.5	TCI2_7	CK_4	SDA_7	RX_7			PWM1_7
PT2.6	HSXT1	INT2.6	TCI1_8	MISO_4	SCL_8	TX_8			PWM0_8
PT2.7	HSXT2	INT2.7	TCI2_8	MOSI_4	SDA_8	RX_8			PWM1_8
PT3.0							OPO1		
PT3.1							OPO2	DAO	
PT3.2								AIO4	
PT3.3								AIO5	
PT3.4								AIO6	
PT3.5								AIO7	
PT3.6								REFO	
PT3.7								OPO	
AIO0								AIO0	
AIO1								AIO1	
AIO2								AIO2	
AIO3								AIO3	



03.數位 IP(Timer A and Timer B)

3.1 範例名稱

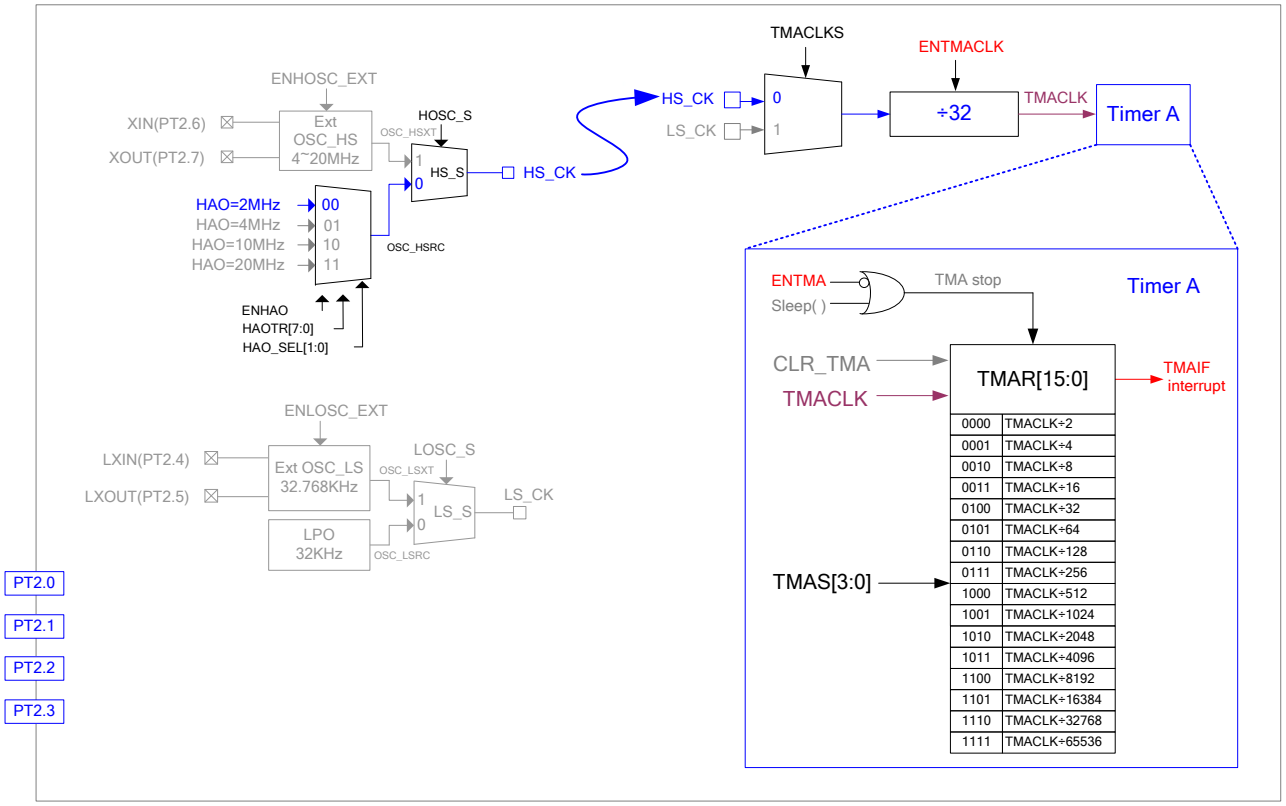
Timer A (or Timer B)使用方式與說明

3.2 範例說明

(1) 程式中用#define　TMATEST 代表使用 TimerA 中斷。
　　程式中用#define　TMBTEST 代表使用 TimerB 中斷。
(2)每進一次 Timer 中斷，會將变量 TimerCount 数值加 1。
　LCD 显示变量 TimerCount 数值。

3.3 系統設定

Timer A：



(1)時脈操頻設定，預設為 4MHz(高速內部震盪器)，最高可達 20MHz@2.6V。
(2)採用紘康 C 函式庫，DrvTMA_Open(X,Y)可將 TimerA IP 與 Timer A Clock 啟動。
　其中 X 代表 TMAR 除頻，選擇 15 即除以 65536，

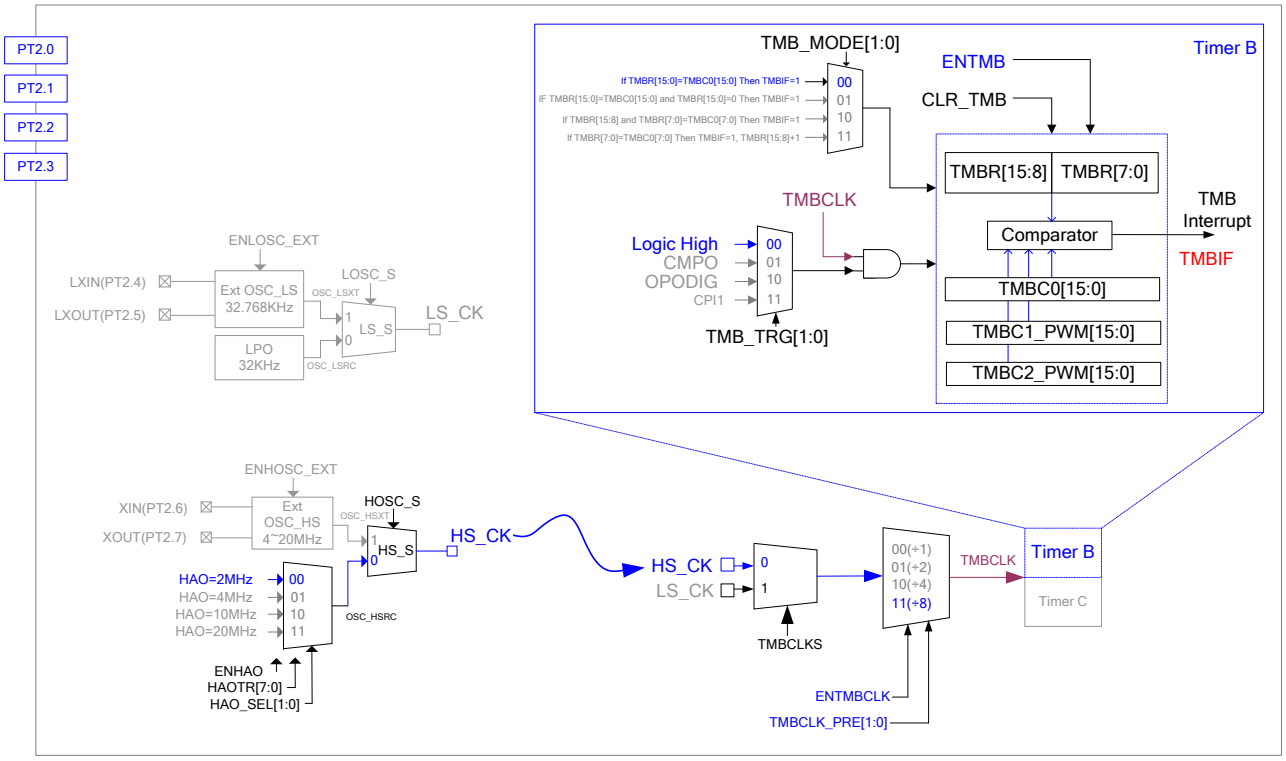
Y 為選擇高速或低速 CPU Clock 進入 TimerA IP。

(3)紘康 C 函式庫 DrvTIMER_EnableInt(E_TMA)，表示將 Timer A 中斷致能。

(4)紘康 C 函式庫 DrvTIMER_ClearIntFlag(E_TMA)，表示將 Timer A 中斷旗標清除。

(5)Timer A/B/C 與 WDT 隸屬於 HW1 中斷，使用格式為 void HW1_ISR(void)。

Timer B：



(1)採用紘康 C 函式庫，DrvTMBC_Clk_Source(X,Y) 可選澤 Timer B 時脈來源，X 為高低速選擇。X=0 為 HS_CK，Y 為除頻選擇，若 Y=3 代表 Timer B IP 時脈 除頻 8。

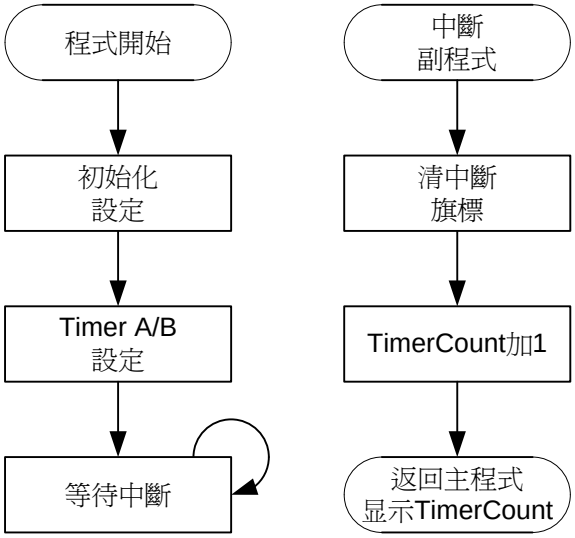
(2)函式 DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0XFFFF)，代表 TimerB IP 設定，包含 Mode 選擇，致能 Timer B 以及 Timer B counter 計數設定。

(3)紘康 C 函式庫 DrvTIMER_EnableInt(E_TMB)，表示將 Timer B 中斷致能。

(4)紘康 C 函式庫 DrvTIMER_ClearIntFlag(E_TMB)，表示將 TimerB 中斷旗標清除。

(5)Timer A/B/C 與 WDT 隸屬於 HW1 中斷，使用格式為 void HW1_ISR(void)。

4.4 軟體流程



4.5 程式說明

```
0
1  #include "HY16F188.h"
2  #include "System.h"
3  #include "DrvGPIO.h"
4  #include "DrvTimer.h"
5  #include "DrvI2C.h"
6  #include "DrvHWI2C.h"
7  #include "DrvCLOCK.h"
8  #include "HY2613.h"
9  #include "my define.h"
10
11 volatile typedef union _MCUSTATUS
12 {
13     char    _byte;
14     Struct
15     {
16         unsigned b_ADCdone:1;
17         unsigned b_TMAdone:1;
18         unsigned b_TMBdone:1;
19         unsigned b_TMC0done:1;
20         unsigned b_TMC1done:1;
21         unsigned b_RTCdone:1;
22         unsigned b_UART_TxDone:1;
23         unsigned b_UART_RxDone:1;
24     };
25 } MCUSTATUS;
26
27 #define    TMATEST                    //For TimerA Test
```

```
28  //#define    TMBTEST                                  //For TimerB Test
29
30  MCUSTATUS    MCUSTATUSbits;
31  volatile    unsigned int TimerCount;       //
32  extern unsigned char seg[16];
33
34  void InitalTimerA(void);
35  void InitalTimerB(void);
36  void Delay(unsigned int num);
37
38  int main(void)
39  {
40      DrvCLOCK_SelectIHOSC (1);                        // Select HAO 4MHz
41      DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
42  #if defined(TMATEST)
43      InitalTimerA();
44  #elif defined(TMBTEST)
45      InitalTimerB();
46  #endif
47      InitalI2C();
48      SYS_EnableGIE(7,0x3F);                           // Enable GIE(Global Interrupt)
49
50      Ini_Display();
51      ClearLCDframe();
52      Delay(10000);
53      DisplayHYcon();
54      Delay(50000);
55      MCUSTATUSbits._byte = 0;
56      TimerCount=0;
57
58      while(1)
59      {
60          if(MCUSTATUSbits.b_TMAdone==1)
61          {
62              LCD_DATA_DISPLAY(TimerCount);            //LCD display the data of TimerCount
63              MCUSTATUSbits.b_TMAdone=0;
64          }
65          if(MCUSTATUSbits.b_TMBdone==1)
66          {
67              LCD_DATA_DISPLAY(TimerCount);            //LCD display the data of TimerCount
68              MCUSTATUSbits.b_TMBdone=0;
69          }
70      }
71      return 0;
72  }
73
74  /*---------------------------------------------------------------------------*/
```

```
75  /* Hardware Communication Interrupt                    */
76  /*I2C Interrupt Service Routines                       */
77  /*---------------------------------------------------------------------------*/
78  void HW0_ISR(void)
79  {
80      unsigned char I2C_Status;
81
82      if(DrvI2C_ReadIntFlag()==E_DRVI2C_INT)          // Get I2C Interrupt Flag
83      {
84          I2C_Status=DrvI2C_GetStatusFlag();          // Get I2C Status Flag
85          switch(I2C_Status)
86          {
87              case 0x90: //MACTFlag+RWFlag
88                  { /* START has been transmitted */
89                      DrvI2C_WriteData(I2C_TARGET);   // Send Slave Address & R/W Bit
90                      DrvI2C_Ctrl(0,0,0,0);           // Clear all I2C flag
91                      break;
92                  };
93              case 0x84: //MACTFlag+ACKFlag
94                  { /* Slave A + W has been transmitted. ACK has been
    received. */
95                      DrvI2C_WriteData(Sendbuf[DataTxIndex++]);   // Send Data to Slave
96                      DrvI2C_Ctrl(0,0,0,0);           // Clear all I2C flag
97                      break;
98                  };
99              case 0x80: //MACTFlag
100                 { /* Slave A + W has been transmitted. NACK has
    been received. */
101                     DrvI2C_WriteData(Sendbuf[DataTxIndex++]);   // Send Data to Slave
102                     DrvI2C_Ctrl(0,0,0,0);    // Clear all I2C flag
103                     break;
104                 };
105             case 0x30:
106                 {/*A STOP has been transmitted.*/
107                     DrvI2C_Ctrl(0,0,0,0);    // Clear all I2C flag
108                     EndFlag=1;
109                     break;
110                 };
111             case 0x8C: // MACTFlag+DFFlag+ACKFlag
112                 { /* DATA has been transmitted and ACK has been
    received */
113                     if(DataTxIndex<DataTxLen)
114                     {
115                         DrvI2C_WriteData(Sendbuf[DataTxIndex++]);   //Send Data to Slave
116                         DrvI2C_Ctrl(0,0,0,0);    // Clear all I2C flag
117                     }
118                     else
119                     {
```

```
120                     if(I2C_RW == WRITE)
121                     {
122                        DrvI2C_Ctrl(0,1,0,0);          // I2C as master sends STOP signal
123                        EndFlag=1;
124                     }
125                     else if(I2C_RW == READ)
126                        DrvI2C_Ctrl(1,0,0,0);          // I2C as master sends START signal
127                     DataTxIndex=0;
128                  }
129                  break;
130               };
131            case 0x88:                                 //MACTFlag+DFFlag
132               { /* DATA has been transmitted and NACK has been
                  received */
133                  DrvI2C_Ctrl(0,1,0,0);               // I2C as master sends STOP signal
134                  DataTxIndex=0;
135                  EndFlag=1;
136                  break;
137               };
138            case 0xB0:
139               { /* A repeated START has been transmitted. */
140                  DrvI2C_WriteData(I2C_TARGET | READ);  //Send Slave Address & R/W Bit
141                  DrvI2C_Ctrl(0,0,0,0);               // Clear all I2C flag
142                  break;
143               }
144            case 0x94:                                 //MACTFlag+RWFlag
145               { /* Slave A + R has been transmitted. ACK has been
                  received. */
146                  DrvI2C_Ctrl(0,0,0,1);               // Set ACK bit
147                  break;
148               };
149            case 0x9C:                                 //MACTFlag+RWFlag+DFFlag+ACKFlag
150               { /* Data byte has been received. ACK has been
                  transmitted. */
151                  if(DataRxLen>DataRxIndex)
152                  {
153                     Recbuf[DataRxIndex++]=DrvI2C_ReadData();
154                     DrvI2C_Ctrl(0,0,0,0);            // Clear all I2C flag
155                  }
156                  else
157                  {
158                     Recbuf[DataRxIndex++]=DrvI2C_ReadData();
159                     DrvI2C_Ctrl(0,1,0,0);            // I2C as master sends STOP signal
160                     EndFlag=1;
161                  }
162                  break;
163               };
164            case 0x98:                                 //MACTFlag+RWFlag+DFFlag
```

```
165                 { /* Data byte has been received. NACK has been transmitted. */
166                     DrvI2C_Ctrl(0,1,0,0);                   // I2C as master sends STOP signal
167                     EndFlag=1;
168                     break;
169                 };
170             default:
171                 {
172                     DrvI2C_Ctrl(0,0,0,0);                   // Clear all I2C flag
173                     EndFlag=1;
174                     break;
175                 };
176         }
177         DrvI2C_ClearEIRQ();
178         DrvI2C_ClearIntFlag(2);                             // Clear I2C Interrupt Flag(I2CIF)
179         SYS_EnableGIE(7,0X3F);                              // Enable GIE(Global Interrupt)
180     }
181     if(DrvI2C_ReadIntFlag()==E_DRVI2C_ERROR_INT)            //Get I2C Error Interrupt Flag
182     {
183         EndFlag=1;
184         DrvI2C_ClearIRQ();
185         DrvI2C_ClearIntFlag(2);                             // Clear I2C Interrupt Flag(I2CEIF)
186         DrvI2C_Ctrl(0,1,0,0);                               // I2C as master sends STOP signal
187         SYS_EnableGIE(7,0X3F);                              // Enable GIE(Global Interrupt)
188     }
189 }
190
191 /*---------------------------------------------------------------------------*/
192 /*Timer A/B/C Interrupt Service Routines                    */
193 /*---------------------------------------------------------------------------*/
194 void HW1_ISR(void)
195 {
196     if(DrvTIMER_GetIntFlag (E_TMA))                         // Get the interrupt flag of TMA
197     {
198         DrvTIMER_ClearIntFlag(E_TMA);                       //Clear TMA interrupt flag
199         TimerCount++;
200         MCUSTATUSbits.b_TMAdone=1;
201     }
202     else if(DrvTIMER_GetIntFlag (E_TMB))                    // Get the interrupt flag of TMB
203     {
204         DrvTIMER_ClearIntFlag(E_TMB);                       //Clear TMB interrupt flag
205         TimerCount++;
206         MCUSTATUSbits.b_TMBdone=1;
207     }
208 }
210 }
211
212 void InitalTimerA(void)
```

```
213 {
214     DrvTMA_Open(15,0);                                  //TimerA overflow
215                                                         //15 : taclk/65536/32;TMRDV=∫32
216                                                           // 0 : HS_CK
217     DrvTIMER_ClearIntFlag(E_TMA);                       //Clear TMA interrupt flag
218     DrvTIMER_EnableInt(E_TMA);                          //TimerA interrupt enable
219 }
220
221 void InitalTimerB(void)
222 {
223     DrvTMBC_Clk_Source(0,3);                            //TimerB Prescaler 1
224                                                         //0: HS_CK,clock source.
225                                                           //3: clock divider.∫8
226     DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0xFFFF);         //TimerB overflow 0xFFFF
227     DrvTIMER_ClearIntFlag(E_TMB);                       //Clear TMB interrupt flag
228     DrvTIMER_EnableInt(E_TMB);                          //TimerB interrupt enable
229 }
230
231 void Delay( unsigned int num)
232 {
233     for(;num>0;num--)
234         asm("NOP");
235 }
236
```

04.數位 IP(Timer C)

4.1 範例名稱

Timer C 使用方式與說明

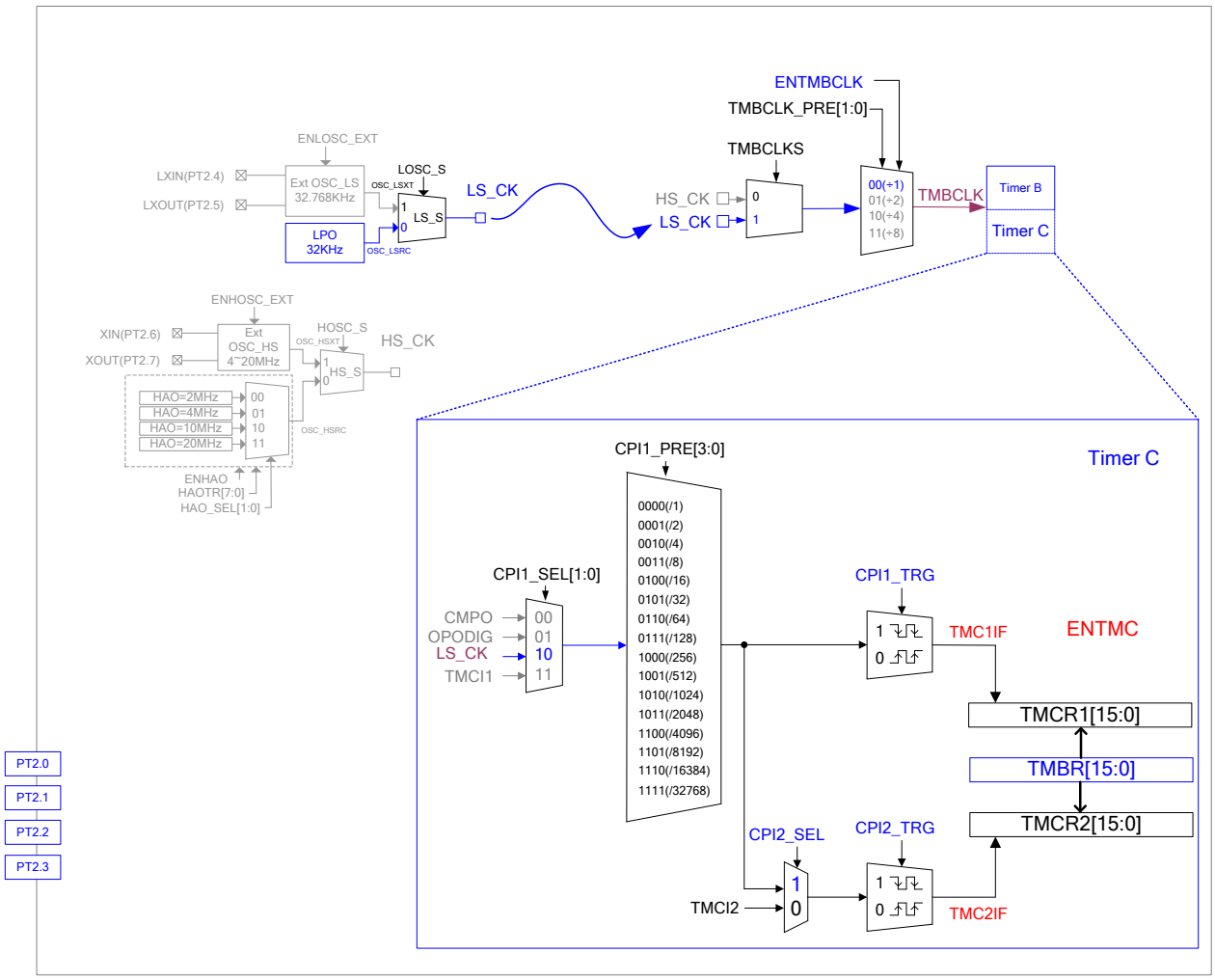
4.2 範例說明

(1)使用 TimerC 中斷。

(2)每進一次 TimerC 中斷，會將 IO 反向輸出。

例如進中斷前 0X5，進中斷後變 0XA。

4.3 系統設定

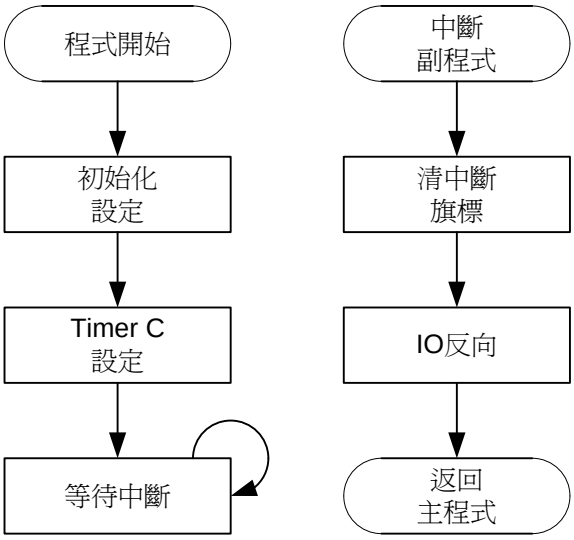


(1)DrvTMB_Open( )會將 Timer B 致能。

(2)DrvCapture1_Open( );設定 Timer C0。

(3)DrvCapture2_Open( );設定 Timer C1。

4.4 軟體流程



4.5 程式說明

```
00
01 #include "HY16F188.h"
02
03 unsigned int i;
04
05 int main(void)
06 {
07         DrvCLOCK_SelectIHOSC (1);                          // Select HAO 4MHz
08         DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
09         i=0X05;
10         DrvGPIO_Open(E_PT2,0X0F,E_IO_OUTPUT);              //PT2_0~3 Set Output
11         DrvGPIO_SetPortBits(E_PT2,i);                      //PT2 Output i=0X05
12
13         DrvTMBC_Clk_Source(0,0);                           //Timer B Prescaler 1
14                                                            //0: HS_CK,clock source.
15                                                            //0: clock divider.÷1
16
17         DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0XFFFF);      //Timer B overflow 0XFFFF
18
19         DrvCapture1_Open(2,14,1);                          //TimerC0 use as Capture 1
20                                                            //input source selection
21                                                            //2:LS_CK
22                                                            //14:÷16384 1:Positive-edge trigger
23
24         DrvCapture2_Open(1,1);                             //TimerC1 use as Capture 2
25                                                            // input source selection
26
27         DrvTIMER_ClearIntFlag(E_TMC0);                     //Clear TMC0 interrupt flag
28         DrvTIMER_ClearIntFlag(E_TMC1);                     //Clear TMC1 interrupt flag
29         DrvTIMER_EnableInt(E_TMC0);                        //TimerC0 interrupt enable
30         DrvTIMER_EnableInt(E_TMC1);                        //TimerC1 interrupt enable
31         SYS_EnableGIE(7,0x3F);                             //Enable GIE
32
33         while(1);                                          //Wait for Interrupt
34 }
```

37		
38	void HW1_ISR(void)	
39	{	
40	DrvTIMER_ClearIntFlag(E_TMC0);	//Clear TMC0 interrupt flag
41	DrvTIMER_ClearIntFlag(E_TMC1);	//Clear TMC1 interrupt flag
42	i=i^0XF;	//i XOR 0XF
43	DrvGPIO_SetPortBits(E_PT2,i);	//PT2 Output i=0X0A~0X05
44	}	
45		

05.數位 IP(WDT)

5.1 範例名稱

WDT 使用方式與說明

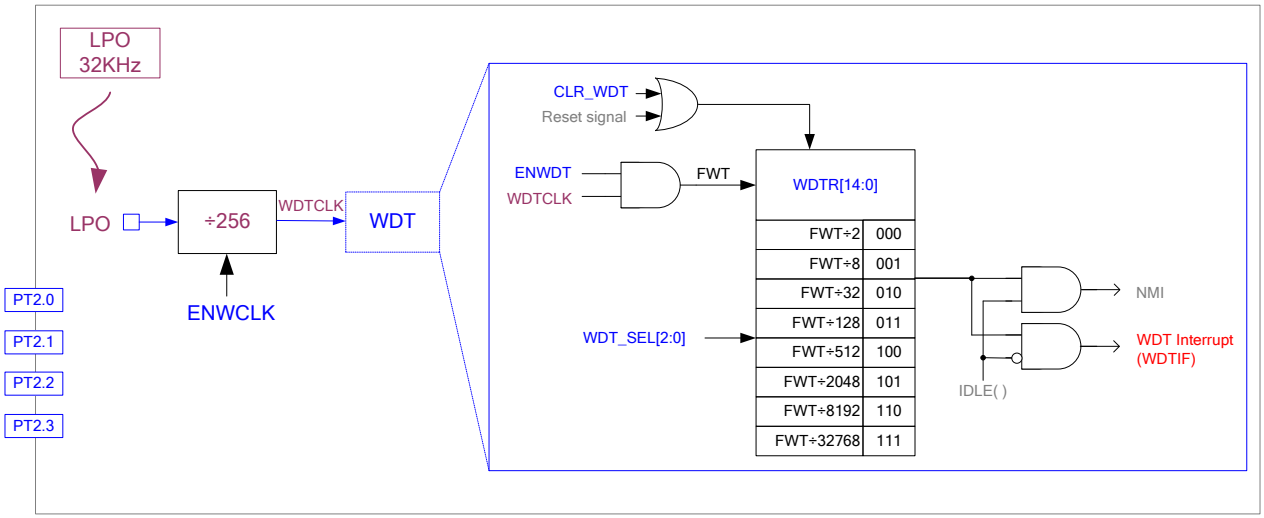
5.2 範例說明

(1)使用 WDT 中斷。

(2)每進一次 WDT 中斷，會將变量 TimerCount 数值加 1。

LCD 显示变量 TimerCount 数值。

5.3 系統設定

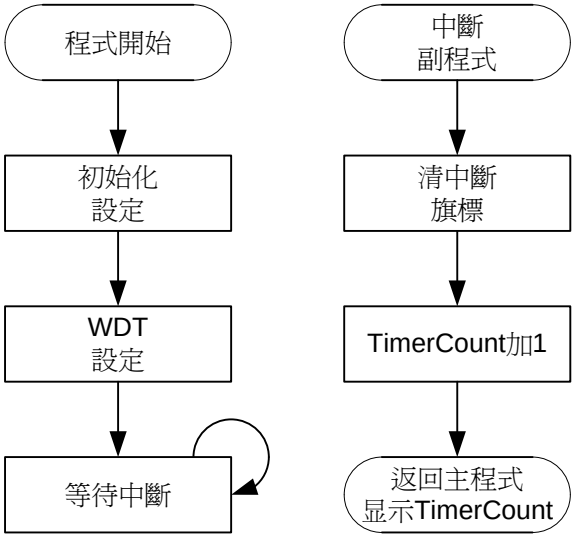


(1)DrvWDT_Open( )會將 WDT 致能。

(2)DrvTIMER_EnableInt(E_WDT)，表示將 WDT 中斷致能。

(3)DrvTIMER_ClearIntFlag(E_WDT)，表示將 WDT 中斷旗標清除。

5.4 軟體流程



6.5 程式說明

```
0
1  #include "HY16F188.h"
2  #include "System.h"
3  #include "DrvGPIO.h"
4  #include "DrvTimer.h"
5  #include "DrvI2C.h"
6  #include "DrvHWI2C.h"
7  #include "DrvCLOCK.h"
8  #include "HY2613.h"
9  #include "my define.h"
10
11 volatile typedef union _MCUSTATUS
12 {
13     char    _byte;
14     Struct
15     {
16         unsigned b_ADCdone:1;
17         unsigned b_TMAdone:1;
18         unsigned b_TMBdone:1;
19         unsigned b_TMC0done:1;
20         unsigned b_TMC1done:1;
21         unsigned b_RTCdone:1;
22         unsigned b_UART_TxDone:1;
23         unsigned b_UART_RxDone:1;
24     };
25 } MCUSTATUS;
26
```

```
27  MCUSTATUS    MCUSTATUSbits;
28  volatile    unsigned int TimerCount;       //
29  extern unsigned char seg[16];
30
31  void Delay(unsigned int num);
32
33  int main(void)
34  {
35      DrvCLOCK_SelectIHOSC (1);                   // Select HAO 4MHz
36      DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
37      InitalI2C();
38      SYS_EnableGIE(7,0x3F);                      // Enable GIE(Global Interrupt)
39      Ini_Display();
40      ClearLCDframe();
41      Delay(10000);
42      DisplayHYcon();
43      Delay(50000);
44      DrvWDT_Open(E_IRQ,E_PRE_SCALER_D32);        //WDT IRQ open pre scaler 32
45      DrvWDT_ClearWDT();                           //Clear WDT interrupt flag
46      DrvTIMER_EnableInt(E_WDT);                  //WDT interrupt enable
47      MCUSTATUSbits._byte = 0;
48      TimerCount=0;
49
50      while(1) //Wait for Interrupt
51      {
52          if(MCUSTATUSbits.b_WDTdone==1)
53          {
54              LCD_DATA_DISPLAY(TimerCount);       //LCD display the data of TimerCount
55              MCUSTATUSbits.b_WDTdone=0;
56          }
57      }
58      return 0;}
59
60  /*WDT&Timer A/B/C Interrupt Service Routines                */
61  void HW1_ISR(void)
62  {
63      if(DrvTIMER_GetIntFlag (E_WDT))             // Get the interrupt flag of WDT
64      {
65          TimerCount++;
66          MCUSTATUSbits.b_WDTdone=1;
67          DrvTIMER_ClearIntFlag(E_WDT);             //Clear WDT interrupt flag
68      }
69  }
70  void Delay(unsigned int num)
71    { for(;num>0;num--)
72          asm("NOP");
73  }
```

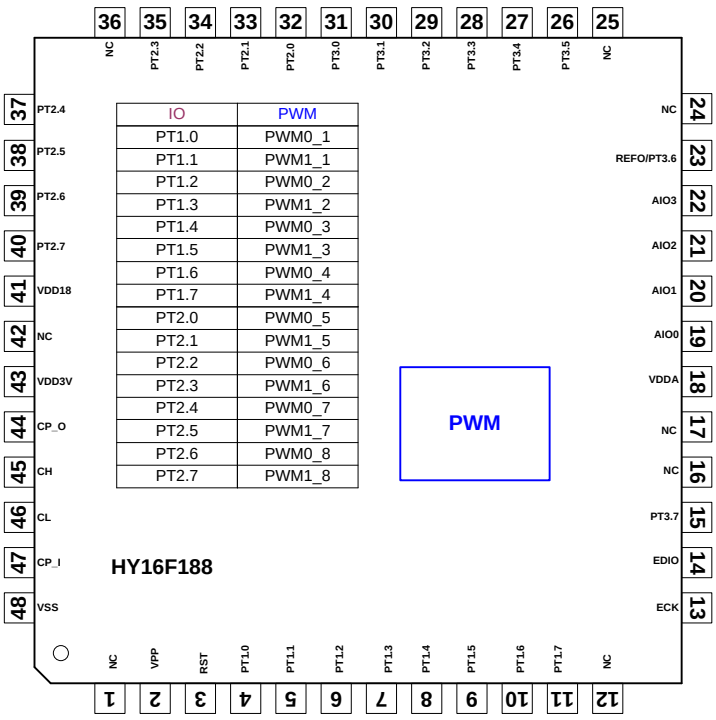
06.數位 IP(PWM)

6.1 範例名稱

PWM 輸出

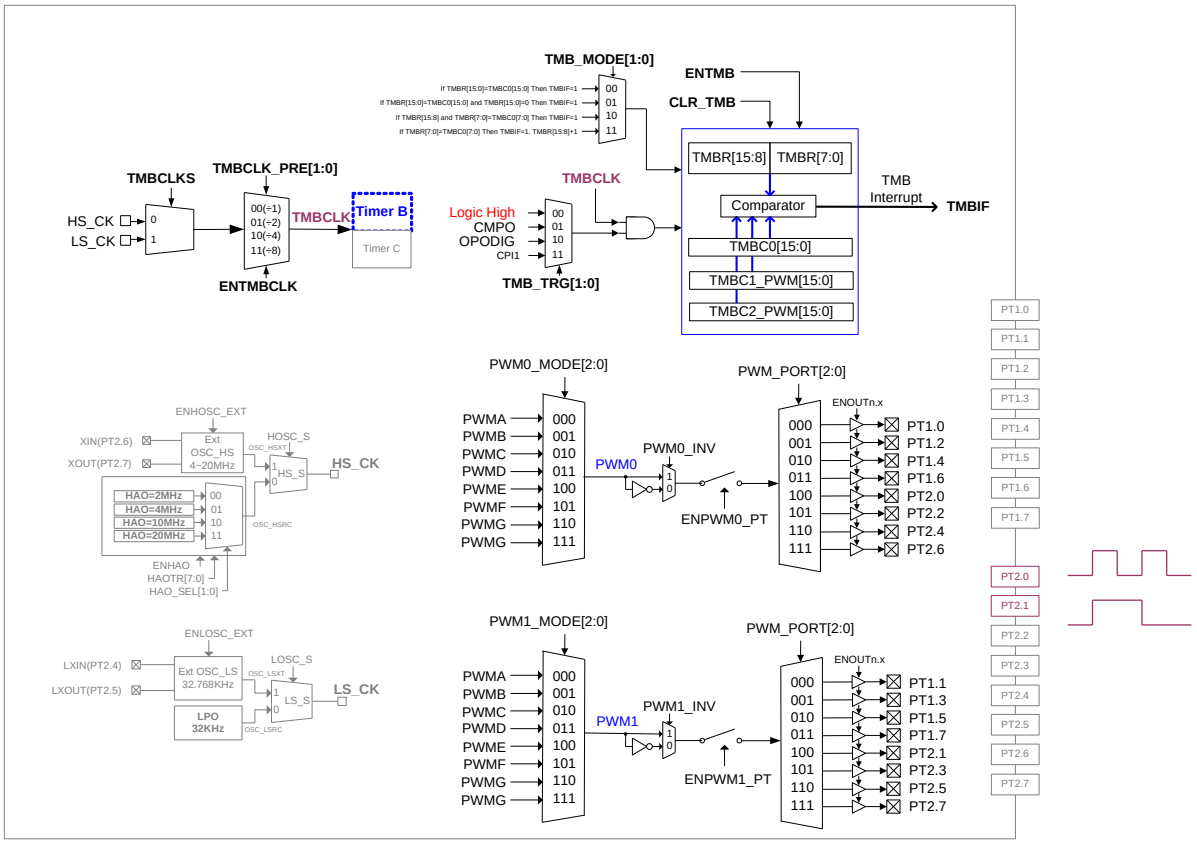
6.2 範例說明

(1)透過適當的 PWM 暫存器設定，與 IO 選擇。
(2)可在 PT2.0 與 PT2.1 觀察到 PWM 波型。

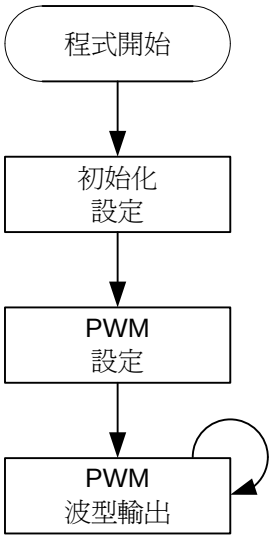


IO	PWM
PT1.0	PWM0_1
PT1.1	PWM1_1
PT1.2	PWM0_2
PT1.3	PWM1_2
PT1.4	PWM0_3
PT1.5	PWM1_3
PT1.6	PWM0_4
PT1.7	PWM1_4
PT2.0	PWM0_5
PT2.1	PWM1_5
PT2.2	PWM0_6
PT2.3	PWM1_6
PT2.4	PWM0_7
PT2.5	PWM1_7
PT2.6	PWM0_8
PT2.7	PWM1_8

6.3 系統設定



6.4 軟體流程



6.5 程式說明

```
00
01  #include "HY16F188.h"
02
03
04
05  int main(void)
06  {
07          DrvCLOCK_SelectIHOSC (1);                       // Select HAO 4MHz
08          DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
09
10          SYS_DisableGIE();
11
12          DrvGPIO_Open(E_PT2,0X03,E_IO_OUTPUT);           //PT2_0~1 Set Output
13          DrvTMBC_Clk_Source(0,0);                        //TimerB Clock enable prescaler 1
14
15          DrvTMB_Open(E_TMB_MODE0,E_TMB_NORMAL,0XFFFF);   //TimerB overflow 0XFFFF
16                                                          //PWM period 0XFFFF
17
18          DrvPWM0_Open(0,1,4);                            //0:PWM mode
19                                                          //1:Normal    0:inverse
20                                                          //4:Select PT2.0 PT2.1 as PWM
21
22          DrvPWM1_Open(1,1,4);                            //1:PWM mode
23                                                          //1:Normal    0:inverse
24                                                          //4:Select PT2.0 PT2.1 as PWM
25
26          DrvPWM_CountCondition(0X7FFF,0X3FFF);           //pwm0 duty 0X7FFF
27                                                          //pwm1 duty 0X3FFF
28
29          while(1);                                       //While Loop
30  }
31
```

07.類比 IP(DAC)

7.1 範例名稱

DAC 使用方式與說明

7.2 範例說明

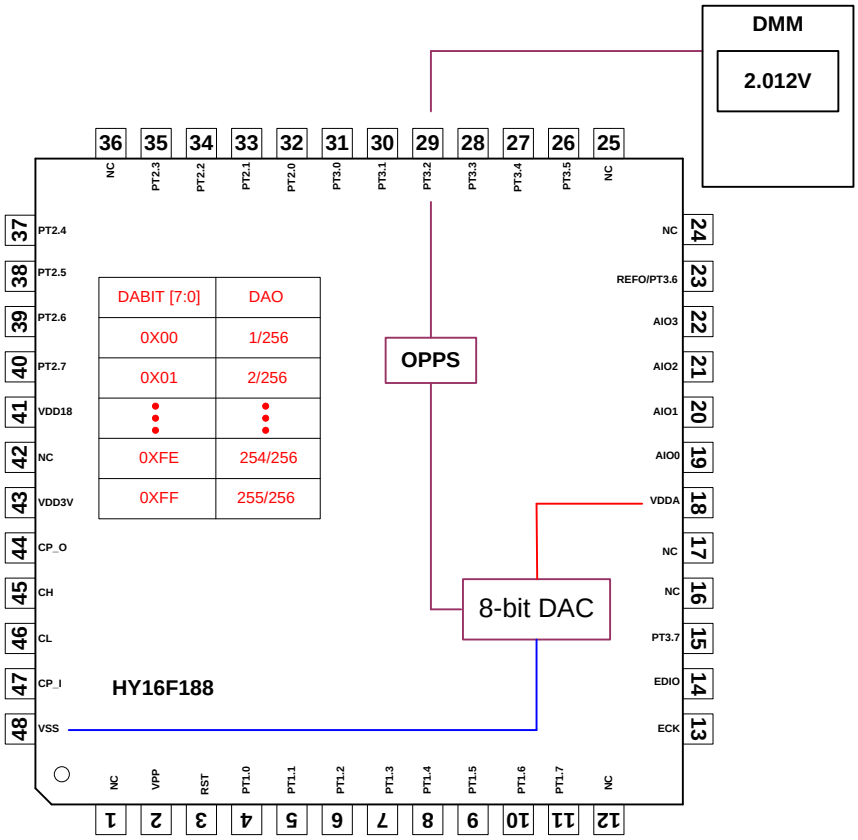
(1)將類比電壓與數位電壓做適當設定。

(2)將 REFO 接到 DAC 的正端。

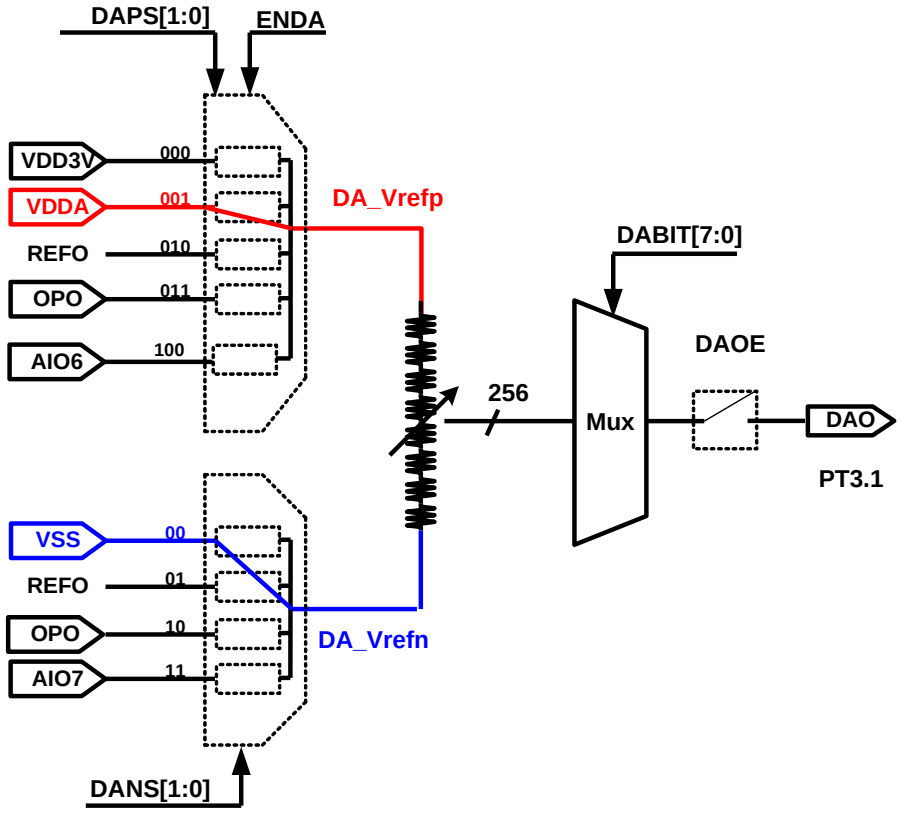
(3)DAC 的負端選擇 VSS，此時會透過 OP 的 AIO4。(AIO4 和 DAC 要同時選擇)

(4)透過適當的 DAC 網路設定，可在 AIO4 輸出端量到 256 階的 DAC 輸出。

(5)或採用 DAC 專用類比電壓輸出腳位 PT3.1

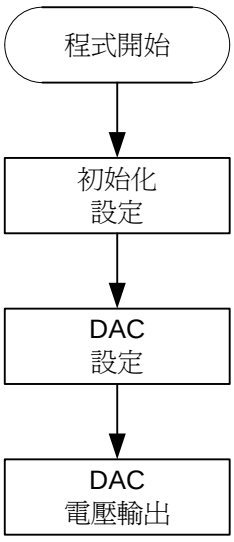


7.3 系統設定



DABIT[7:0]	DABIT
0X00	1/256
0X01	2/256
⋮	⋮
0XFE	254/256
0XFF	255/256

7.4 軟體流程



7.5 程式說明

```
00
01 #include "HY16F188.h"
02
03 int main(void)
04 {
05     DrvCLOCK_SelectIHOSC (1);                          // Select HAO 4MHz
06     DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
07     DrvPMU_VDDA_LDO_Ctrl(E_LDO);                       //LDO ON
08     DrvPMU_VDDA_Voltage(E_VDDA3_0);                    //VDDA=3.0
09     //DrvOP_PInput(0X06);                              //DAO output with AIO4(PIN29)
10     DrvDAC_Enable();                                   //DAC IP enable
11     DrvDAC_EnableOutput();                             //DAC output enable
12     DrvDAC_Open(E_DAC_VDDA,E_DAC_VSSA,0X80);           //DA_Vrefp= VDDA
13                                                        //DA_Vrefn= VSS
14                                                        //DAC BIT 0X80
15     DrvDAC_SetoutputIO(1);                             //enable DAO output with PT3.1
16
17     while(1);                                          //while loop
18 }
19
```

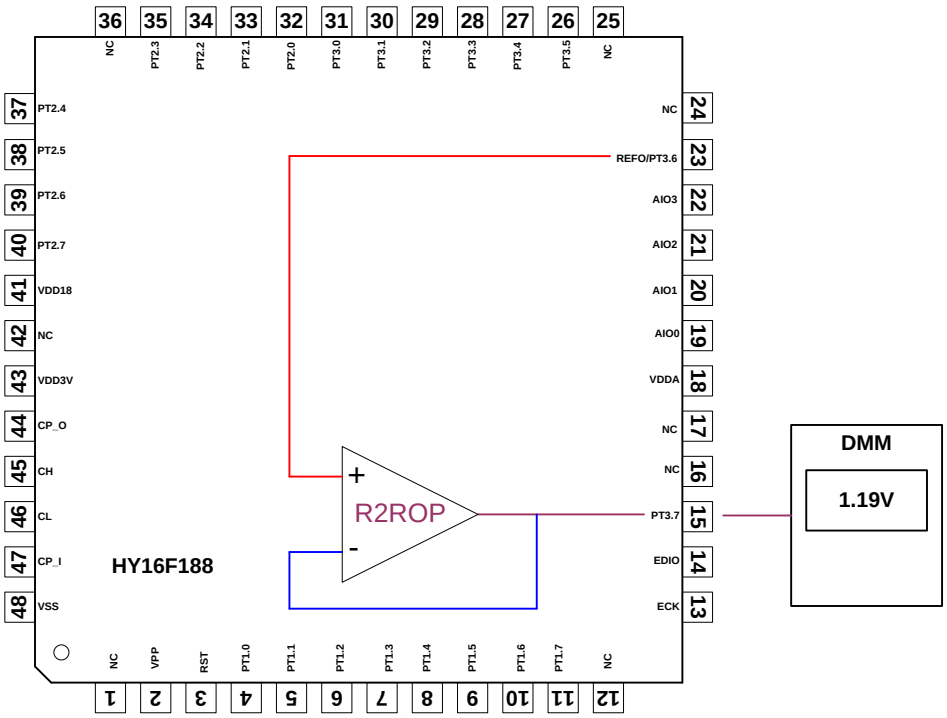
08.類比 IP(OPAMP)

8.1 範例名稱

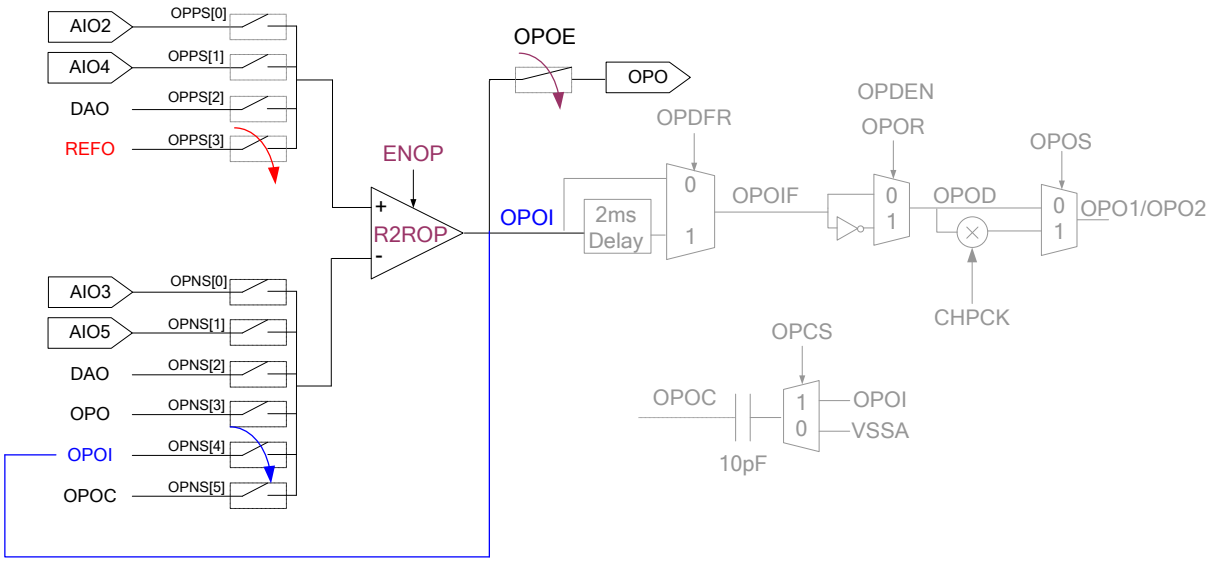
OPAMP 使用與設定方式

8.2 範例說明

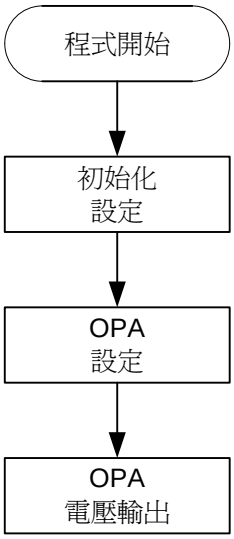
(1)將類比電壓 REFO=1.2V 打開
(2)將 REFO 接到 OPAMP 的正端 V+
(3)OPAMP 的負端選擇 OPOI，此時會接成 OPA Unit Gain Buffer
(4)透過適當的 OPAMP 網路設定，可在 OPO 輸出端量到 REFO=1.2V



8.3 系統設定



8.4 軟體流程



.5 程式說明

```
00
01  #include "HY16F188.h"
02
03  int main(void)
04  {
05      DrvCLOCK_SelectIHOSC (1);                   //Select HAO 4MHz
06      DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
07      DrvPMU_VDDA_LDO_Ctrl(E_LDO);                //LDO ON
08      DrvPMU_VDDA_Voltage(E_VDDA3_0);             //VDDA=3.0
09      DrvPMU_REFO_Enable();                       //REFO ON
10      DrvOP_Open();
11      DrvOP_PInput(0X08);                         //OPA positive reference input selection REFO
12      DrvOP_NInput(0X10);                         //OPA negative reference input selection OPOI
13
14      DrvOP_OPOoutEnable();                       //OP OoutEnable , analog signal
15
16      while(1);                                   //while loop
17  }
18
```

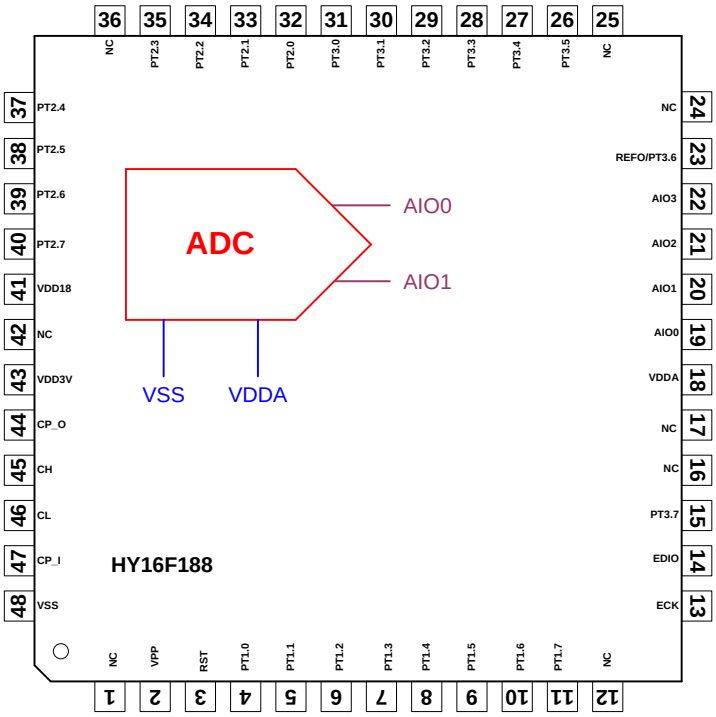
09.類比 IP(ADC)

9.1 範例名稱

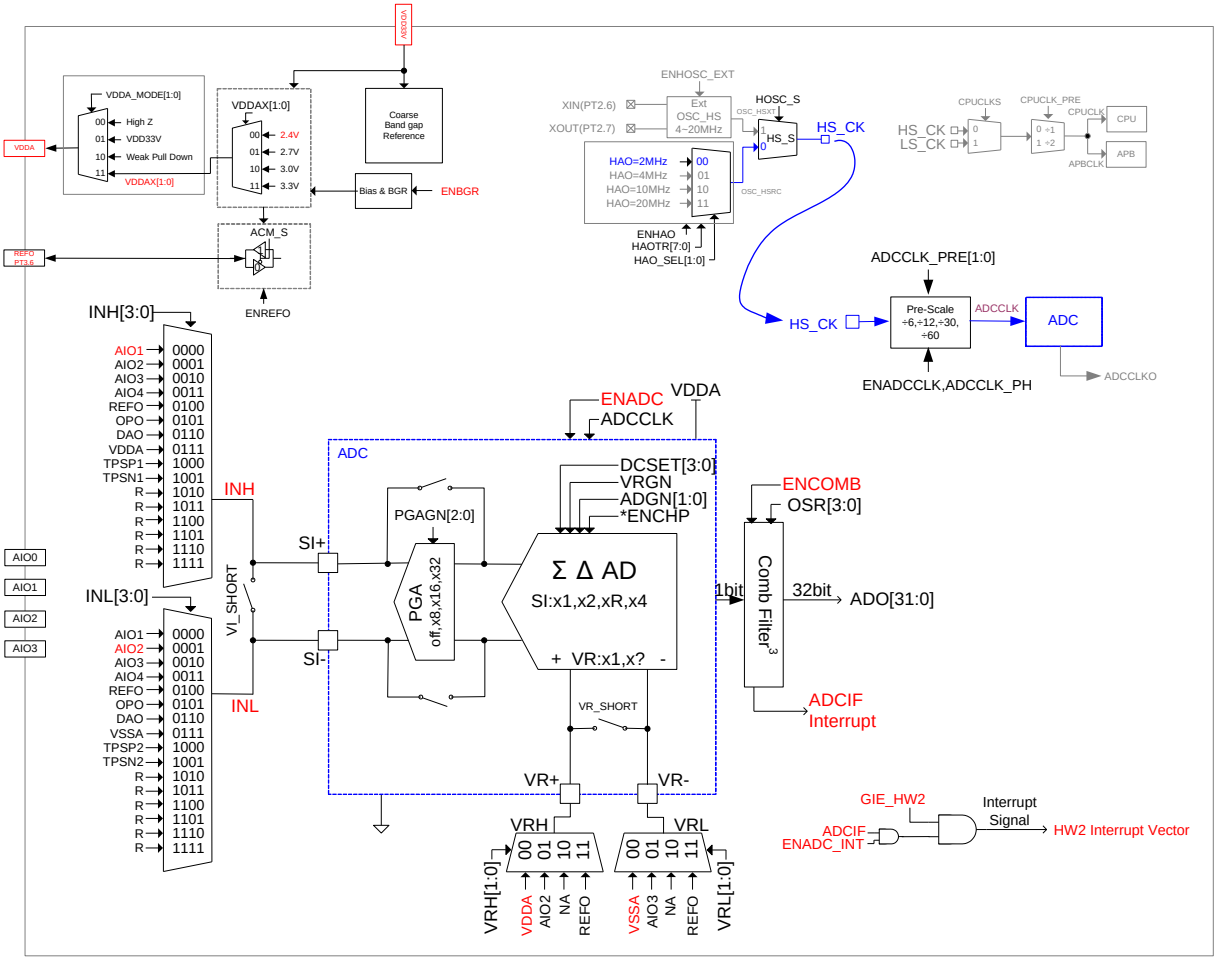
ADC 中斷使用方式

9.2 範例說明

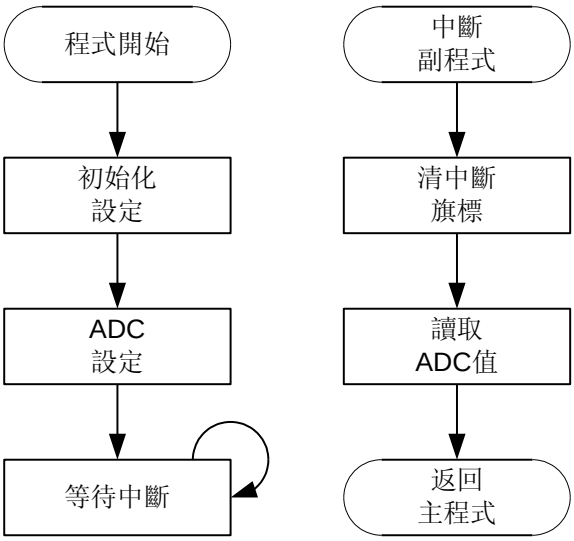
(1)透過 ADC 相關設定，可以實現 ADC 中斷功能。
(2)設定 ADC 的電源 VDDA。
(3)設定 ADC 的時脈 ADCCLK。
(2)ADC 的 Vin 為 AIO0-AIO1
(3)ADC 的 Vref 為 VDDA 對 VSS



9.3 系統設定


VDDA_MODE[1:0]
High Z
VDD33V
Weak Pull Down
VDDAX[1:0]
VDDAX[1:0]
2.4V
2.7V
3.0V
3.3V
Coarse Band gap Reference
Bias & BGR
ENBGR
ACM_S
ENREFO
ENHOSC_EXT
XIN(PT2.6)
XOUT(PT2.7)
Ext OSC_HS 4~20MHz
HOSC_S
HS_S
HS_CK
HAO=2MHz
HAO=4MHz
HAO=10MHz
HAO=20MHz
ENHAO
HAOTR[7:0]
HAO_SEL[1:0]
CPUCLKS
CPUCLK_PRE
CPUCLK
CPU
APB
APBCLK
HS_CK
LS_CK
ADCCLK_PRE[1:0]
Pre-Scale ÷6,÷12,÷30, ÷60
ADCCLK
ADC
ADCCLKO
ENADCCLK,ADCCLK_PH
INH[3:0]
INL[3:0]
AIO1 0000
AIO2 0001
AIO3 0010
AIO4 0011
REFO 0100
OPO 0101
DAO 0110
VDDA 0111
TPSP1 1000
TPSN1 1001
R 1010
R 1011
R 1100
R 1101
R 1110
R 1111
VSSA 0111
TPSP2 1000
TPSN2 1001
ENADC
ADCCLK
VDDA
ADC
DCSET[3:0]
VRGN
ADGN[1:0]
*ENCHP
PGAGN[2:0]
PGA off,x8,x16,x32
Σ Δ AD
SI:x1,x2,xR,x4
+ VR:x1,x? -
SI+
SI-
VI_SHORT
INH
INL
VR_SHORT
VR+
VR-
VRH
VRL
VRH[1:0]
VRL[1:0]
VDDA
AIO2
NA
REFO
VSSA
AIO3
NA
REFO
ENCOMB
OSR[3:0]
Comb Filter3
1bit
32bit
ADO[31:0]
ADCIF Interrupt
GIE_HW2
ADCIF
ENADC_INT
Interrupt Signal
HW2 Interrupt Vector
AIO0
AIO1
AIO2
AIO3

9.4 軟體流程



9.5 程式說明

```
0
1  #include "HY16F188.h"
2  #include "System.h"
3  #include "DrvGPIO.h"
4  #include "DrvTimer.h"
5  #include "DrvI2C.h"
6  #include "DrvHWI2C.h"
7  #include "DrvCLOCK.h"
8  #include "HY2613.h"
9  #include "my define.h"
10
11 volatile typedef union _MCUSTATUS                    //define mcu status bit
12 {
13     char    _byte;
14     Struct
15     {
16         unsigned b_ADCdone:1;
17         unsigned b_TMAdone:1;
18         unsigned b_TMBdone:1;
19         unsigned b_TMC0done:1;
20         unsigned b_TMC1done:1;
21         unsigned b_RTCdone:1;
22         unsigned b_UART_TxDone:1;
23         unsigned b_UART_RxDone:1;
24     };
```

```
25  } MCUSTATUS;
26
27  MCUSTATUS    MCUSTATUSbits;
28  int ADCData;
29  extern unsigned char seg[16];
30
31  void InitalADC(void);
32  void Delay(unsigned int num);
33
34  int main(void)
35  {
36      DrvCLOCK_SelectIHOSC (0);                  // Select HAO 2MHz
37      DrvCLOCK_EnableHighOSC(E_INTERNAL,50);
38      InitalI2C();
39      InitalADC();
40      SYS_EnableGIE(7,0x3F);                     // Enable GIE(Global Interrupt)
41      Ini_Display();
42      ClearLCDframe();
43      Delay(10000);
44      DisplayHYcon();
45      Delay(50000);
46      MCUSTATUSbits._byte = 0;
47
48      while(1)
49      {
50          if(MCUSTATUSbits.b_ADCdone)
51          {
52              LCD_DATA_DISPLAY(ADCData>>16);     //give up 16bits.
53              MCUSTATUSbits.b_ADCdone=0;
54          }
55      }
56      return 0;
57  }
58
59
60  /*---------------------------------------------------------------------------*/
61  /*HW2 ADC Interrupt Subroutines                                              */
62  /*---------------------------------------------------------------------------*/
63  void HW2_ISR(void)
64  {
65      if(DrvADC_ReadIntFlag())                   // Get the interrupt flag of ADC
66      {
67          DrvADC_ClearIntFlag();                 //Clear ADC interrupt flag
68          ADCData=DrvADC_GetConversionData();    //Get ADC data
69          MCUSTATUSbits.b_ADCdone=1;
70      }
71  }
```

72	/* ADC Initialization Subroutines */	
73	void InitalADC(void)	
74	{	
75	//Set ADC input pin	
76	#if defined(TPSCH)	
77	DrvADC_SetADCInputChannel(TPS1,TPS0);	
78	#else	
79	DrvADC_SetADCInputChannel(ADC_Input_AIO0,ADC_Input_AIO1);	//Set the ADC positive/negative input voltage source.
80	#endif	
81	//Set VDDA voltage	
82	DrvPMU_BandgapEnable();	
83	DrvPMU_VDDA_Voltage(E_VDDA2_4);	
84	DrvPMU_VDDA_LDO_Ctrl(E_LDO);	
85	Delay(0x1000);	
86	DrvPMU_AnalogGround(ENABLE);	//ADC analog ground source selection.
87		//1 : Enable buffer and use internal source(need to work with ADC)
88	DrvADC_InputSwitch(OPEN);	//ADC signal input (positive and negative) short(VISHR) control.
89	DrvADC_RefInputShort(OPEN);	//Set the ADC reference input (positive and negative) short(VRSHR) control.
90		
91	DrvADC_Gain(ADC_PGA_Disable,ADC_PGA_Disable);	//Input signal gain for modulator.
92	DrvADC_DCoffset(0);	//DC offset input voltage selection (VREF=REFP-REFN)
93	DrvADC_RefVoltage(0,0);	//Set the ADC reference voltage. VDDA-VSSA
94	DrvADC_FullRefRange(1);	//Set the ADC full reference range select.
95		//0: Full reference range input
96		//1: 1/2 reference range input
97	DrvADC_OSR(0);	//0 : OSR=32768
98		//Enable OSR
99	DrvADC_ClkEnable(0,1);	//Setting ADC CLOCK ADCK=HS_CK/6 & Rising edge is high
100	//Set ADC interrupt	
101	DrvADC_ClearIntFlag();	
102	DrvADC_EnableInt();	
103	DrvADC_Enable();	
104	DrvADC_CombFilter(ENABLE);	
105	DrvADC_CombFilter(DISABLE);	
106	DrvADC_CombFilter(ENABLE); }	
107		
108	void Delay(unsigned int num)	
109	{	
110	for(;num>0;num--)	
111	asm("NOP");	
112	}	

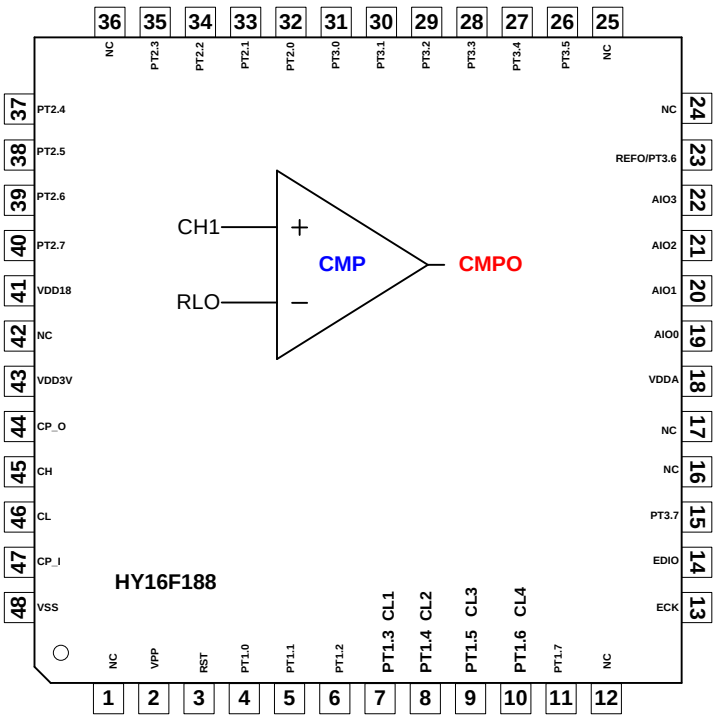
10.類比 IP(CMP)

10.1 範例名稱

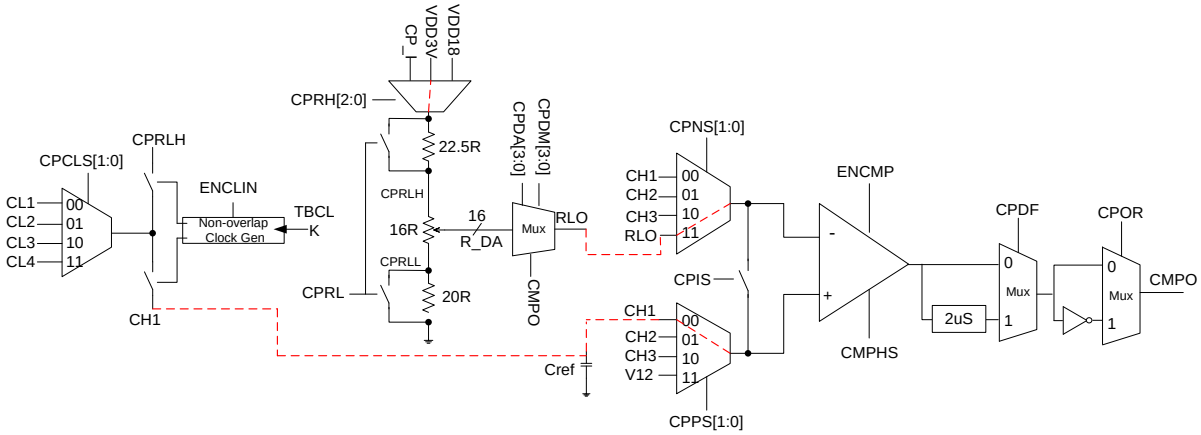
CMP 使用方式說明

10.2 範例說明

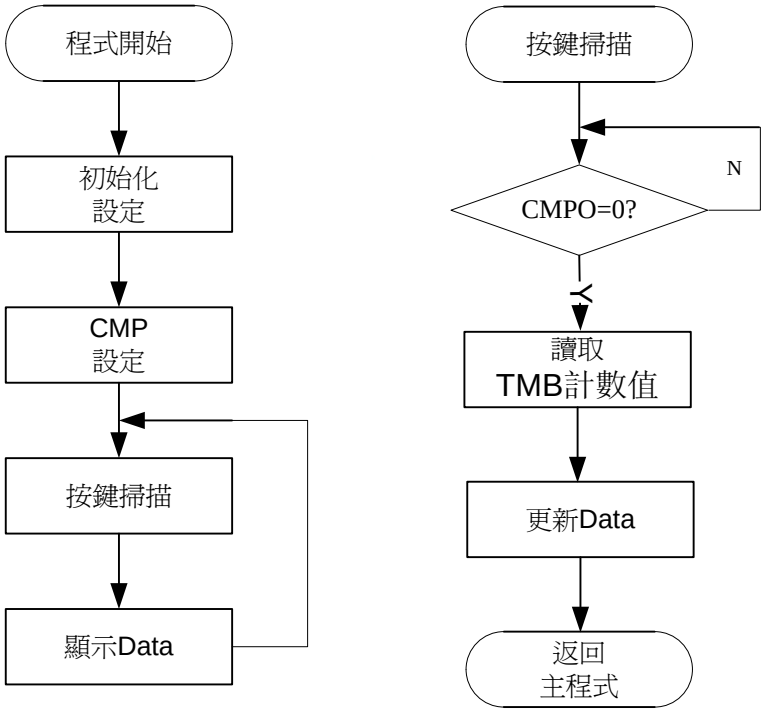
觸控按鍵的測量，主要原理是通過多節點電阻器設置比較電壓值輸入至 RLO，利用多節點電阻器提供電壓對觸控按鍵充電，再用觸控按鍵的電荷對負向輸入通道 CH1 的外接參考電容充電，TMB 計數 CH1 的電壓值大於 RLO 端電壓的充電時間，根據時間來區分觸控按鍵被觸摸與無觸摸的狀態。



10.3 系統設定



10.4 軟體流程



10.5 程式說明

0		
1	#include "HY16F188.h"	
2	#include "System.h"	
3	#include "DrvGPIO.h"	
4	#include "DrvCLOCK.h"	
5	#include "DrvTimer.h"	
6	#include "DrvADC.h"	
7	#include "DrvPMU.h"	
8	#include "DrvREG32.h"	
9	#include "DrvSWI2C.h"	
10	#include "HY2613.h"	
11	#include "seg7.h"	
12	#include "DrvCMP.h"	
13	#include "my define.h"	
14		
15	unsigned int DisplayBuffer[18];	
16	#define Disable 0	
17	#define Enable 1	
18	unsigned int dot=4;	
19	unsigned int k1,k2,k3;	
20	unsigned int LED;	
21	unsigned int neg_flag;	
22	unsigned int LCD1,LCD2,LCD3,LCD4;	
23	unsigned int TCH;	
24	int cmpresult,tmrbcnt;	
25	int threh0=2500;	
26	int threh1=2500;	
27	int threh2=2500;	
28	int threh3=2500;	
29	int CL=0;	
30		
31	void ClearLCDframe(void);	
32	void LCD_DATA_DISPLAY(unsigned int LcdBuffer);	
33	void Delay(unsigned int num);	
34	void Ini_Touch(void);	
35	int ScanKey4(unsigned int TCH);	
36	void LCD_DISPLAY_HYCON(void);	
37		
38		
39	int main(void)	
40	{	
41	DrvCLOCK_SelectIHOSC (1);	// Select HAO 4MHz
42	DrvCLOCK_EnableHighOSC(E_INTERNAL,10);	
43	Ini_Touch();	

```
44    Ini_Display();
45    On_Display();
46    ClearLCDframe();
47    LCD_DATA_DISPLAY(0);
48    threh0=2500;                                    //CL0
49    threh1=2500;                                    //CL1
50    threh2=2500;                                    //CL2
51    threh3=2200;                                    //CL3
52    while(1)
53    {
54      for(TCH=0;TCH<4;TCH++)
55        {ScanKey4(TCH);}
56      LCD_DATA_DISPLAY(LED);                        //LCD Display Data
57    }
58  }
59
60  /* Touch Key Scan   */
61  int ScanKey4(unsigned int TCH)
62  {
63    CL=TCH;pio_00=0x0100;
64    DrvTMB_ClearTMB();                              //clear TMB
65    DrvCMP_EnableNonOverlap(CL);
66    cmpresult=DrvCMP_ReadData();
67    while(cmpresult==1){cmpresult=DrvCMP_ReadData();}
68    tmrbcnt=DrvTMB_CounterRead();                   //TMB
69    switch(CL)
70    {
71      case 0 :
72     if(tmrbcnt<=threh0){if(LED<999999){LED++;}}break;
73      case 1 :
74    if(tmrbcnt<=threh1){if(LED<999999){LED=LED-1;}}break;
75      case 2 :
76    if(tmrbcnt<=threh2){if(LED<999999){LED=LED+10000;}}break;
77      case 3 :
78      if(tmrbcnt<=threh3){if(LED<999999){LED=0;}}break;
79      default: LED=0;
80    }
81    pio_00=0x0101;
82    pio_04=0x0100;
83    if(LED>999900){LED=0;}
84    return LED;
85  }
86
87  /* Clear LCD RAM      */
88  void ClearLCDframe(void)
89  {
90    int i=0;
91    for(i=0;i<11;i++){DisplayBuffer[i]=0x00;}
```

```
92      RAM2LCD(DisplayBuffer,11);
93  }
94
95  /* LCD RAM Data       */
96  void LCD_DATA_DISPLAY(unsigned int LcdBuffer)
97  {
98      int i;
99      for(i=7;i>=2;i--)
100     {
101         DisplayBuffer[i]=seg[LcdBuffer%10];
102         LcdBuffer=LcdBuffer/10;
103     }
104     DisplayBuffer[dot]=DisplayBuffer[dot]|seg_h;
105     DisplayBuffer[8]=S_Bat1|S_Bat2|S_Bat3|S_Bat4|k2;
106     DisplayBuffer[9]=0x00|k1|k3;
107     DisplayBuffer[10]=S18|S19|S20|S21;
108     RAM2LCD(DisplayBuffer,11);
109 }
110
111 /* Initial CMP    */
112 void Ini_Touch(void)
113 {
114     DrvCMP_Enable();                    //CMP enable
115     DrvCMP_Open(1,1,1);                 //CMP open
116     DrvCMP_PInput(0);                    //Comparator positive input    CH1
117     DrvCMP_NInput(3);                   //Comparator negative input    RLO
118     DrvCMP_RLO_refV(2,1);
119     DrvCMP_DisableNonOverlap();
120     DrvCMP_RLO_Ctrl(0,1);
121     DrvCMP_InputSwitch(1);
122     //enable non-over lap
123     DrvCMP_InputSwitch(0);
124     DrvCMP_RLO_Ctrl(5,0);//5-->1
125     DrvTMB_ClearTMB();
126     DrvCLOCK_SelectIHOSC(0x01);
127     DrvCLOCK_SelectMCUClock(0,0);        //select MCU clock as LS_CK,div1
128     DrvTMBC_Clk_Source(0,0);//(1,0)      //TimerB Clock enable pre_scale 1
129     DrvTMB_Open(E_TMB_MODE0,E_TMB_CMP_HIGH,0xffff);  //TimerB overflow 0xffff->PWM period 0xffff
130     DrvTMB_ClearTMB();
131     //Set VDDA voltage
132     DrvPMU_VDDA_Voltage(E_VDDA2_4);
133     DrvPMU_VDDA_LDO_Ctrl(E_LDO);
134     DrvPMU_BandgapEnable();
135     DrvPMU_REFO_Enable();
136     DrvPMU_AnalogGround(Enable);        //ADC analog ground source selection.
137                                         //1 : Enable buffer and use internal
                                            source(need to work with ADC)
138     DrvPMU_LDO_LowPower(Enable);        //VDD LDO    power control.
                                            //0 : Normal;1 : Low power
```

```
139     Delay(0x1000);
140     k1=k2=k3=0;dot=4;
141     LED=0;LCD1=LCD2=LCD3=LCD4=0;
142 }
143
144 void Delay(unsigned int num)
145 {
146     for(;num>0;num--)
147         asm("NOP");
148 }
```

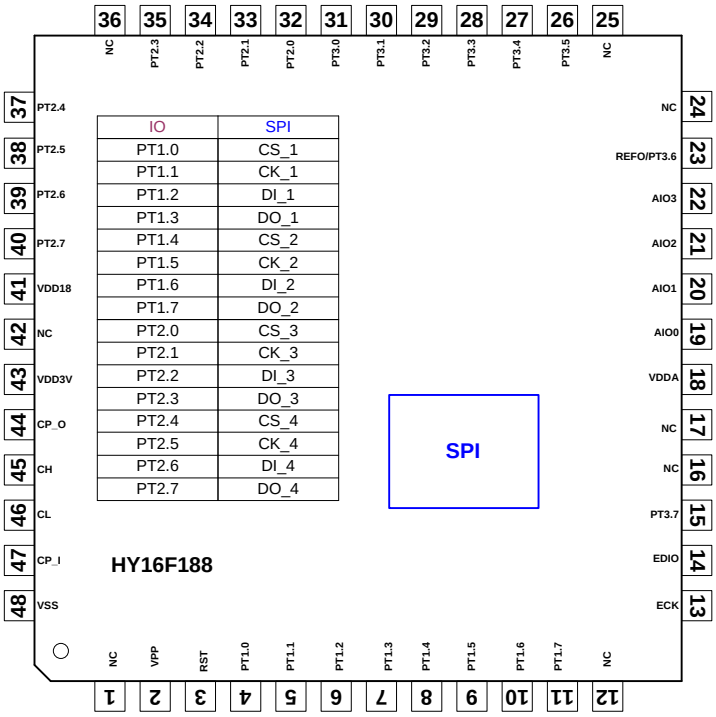
11.通訊 IP(SPI)

11.1 範例名稱

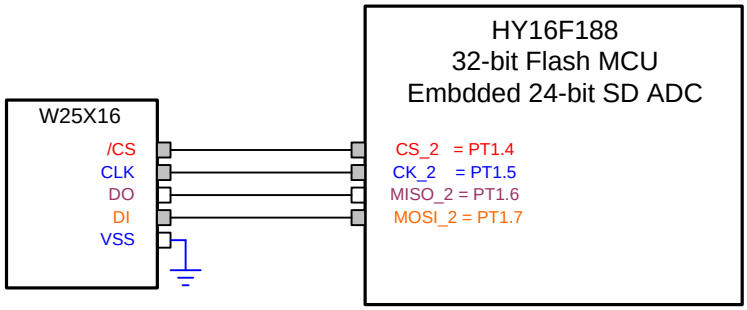
通信 SPI 測試

11.2 範例說明

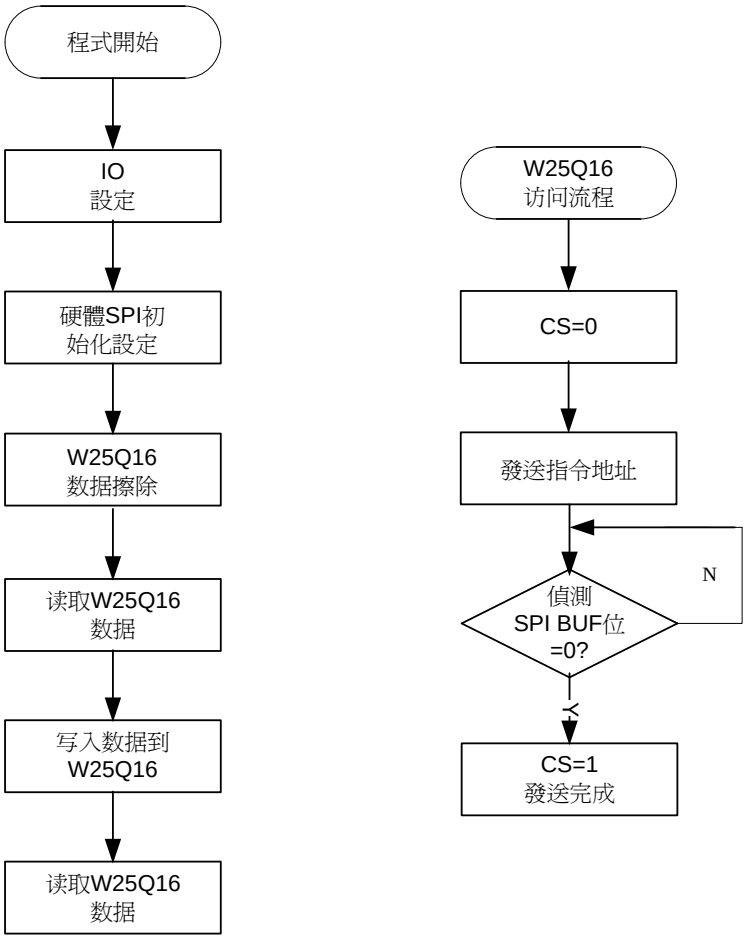
(1)透過 SPI 腳位與 SPI 暫存器設定。
(2)測試硬體 SPI 對 W25X16 寫入读取功能



11.3 系統設定



11.4 軟體流程



12.5 程式說明

(主程式)

0		
1	#include "HY16F188.h"	
2	#include "System.h"	
3	#include "DrvGPIO.h"	
4	#include "DrvSPI32.h"	
5	#include "DrvCLOCK.h"	
6	#include "SPI_Flash.h"	
7	#include "my define.h"	
8		
9		
10		
11	volatile typedef union _MCUSTATUS	//define mcu status bit
12	{	
13	char _byte;	
14	struct	
15	{	
16	unsigned b_ADCdone:1;	
17	unsigned b_TMAdone:1;	
18	unsigned b_TMBdone:1;	
19	unsigned b_TMC0done:1;	
20	unsigned b_TMC1done:1;	
21	unsigned b_RTCdone:1;	
22	unsigned b_UART_TxDone:1;	
23	unsigned b_UART_RxDone:1;	
24	};	
25	} MCUSTATUS;	
26		
27	#define FlashAddress 0x000000	
28	MCUSTATUS MCUSTATUSbits;	
29	unsigned char Flash_Write_Buffer[256];	
30	unsigned char Flash_Read_Buffer[256];	
31	unsigned short Flash_ID;	
32		
33	void Delay(unsigned int num);	
34		
35	int main(void)	
36	{	
37	const unsigned char String1[]={ "Hycon Technology	
38	was established in July, 2007 in Taipei, Taiwan.We dedicate	
39	ourselves to develop high precision and low drift analog	
40	signal related processing ICs." };	
41	const unsigned char String2[]={"The identity stands for	
42	the vision and values of Hycon Technology- To	

```
43  be the ideal solution partner in the area of analog circuit." };
44      unsigned short index_d,Size;
45
46      //DrvCLOCK_SelectIHOSC (1);///
47      DrvCLOCK_EnableHighOSC(E_EXTERNAL,50);          //Select HSXT 4MHz///
48      InitalSPI();
49
50      Flash_ID=SpiFlash_ReadMidDid();                 //Read flash ID
51
52      Size=sizeof(String1);
53      SpiFlash_ChipErase();                           //Erase DATA
54      Delay(256);
55      SpiFlash_ReadData(Flash_Read_Buffer,FlashAddress,Size);  //Read DATA
56      Delay(256);
57
58      for( index_d=0;index_d<Size;index_d++)
59      {
60          Flash_Write_Buffer[index_d]=String1[index_d];
61      }
62  SpiFlash_PageProgram(Flash_Write_Buffer,FlashAddress,Size);  //Write DATA
63      Delay(256);
64      SpiFlash_ReadData(Flash_Read_Buffer,FlashAddress,Size);  //Read DATA
65      Delay(256);
66
67
68      //
69      Size=sizeof(String2);
70      SpiFlash_SectorErase(0x00);                     //Erase DATA
71      Delay(256);
72      SpiFlash_ReadData(Flash_Read_Buffer,FlashAddress,Size);  //Read DATA
73      Delay(256);
74
75      for( index_d=0;index_d<Size;index_d++)
76      {
77          Flash_Write_Buffer[index_d]=String2[index_d];
78      }
79  SpiFlash_PageProgram(Flash_Write_Buffer,FlashAddress,Size);  //Write DATA
80      Delay(256);
81      SpiFlash_ReadData(Flash_Read_Buffer,FlashAddress,Size);  //Read DATA
82      Delay(256);
83
84
85      while(1);
86      return 0;
87  }
```

(副程式)

```
89  // For W25X40, Manufacturer ID: 0xEF; Device ID: 0x12
90  // For W25X16, Manufacturer ID: 0xEF; Device ID: 0x14
91  unsigned int SpiFlash_ReadMidDid(void)
92  {
93      unsigned int au32SourceData;
94      unsigned int au32DestinationData;
95
96      DrvGPIO_ClrBit(Flash_PORT,SCS_PIN);          // /CS: active
97
98      DrvSPI32_BitLength(8);                       // configure transaction length as 8 bits
99      // Send Command: 0x90, Read Manufacturer/Device ID
100     au32SourceData = sFLASH_CMD_REMS<<24;
101     DrvSPI32_Write(au32SourceData);
102     while((spi_00>>BUF) & 0x01);                 // Wait Busy
103
104     DrvSPI32_BitLength(24);                      // configure transaction length as 24bits
105     DrvSPI32_Write(0x0);                         // Send 24-bit '0', dummy
106     while((spi_00>>BUF) & 0x01);                 // Wait Busy
107
108     DrvSPI32_BitLength(16);                      // configure transaction length as 16bits
109     DrvSPI32_Write(0xFFFFFFFF);                  // dump Rx register
110     while((spi_00>>BUF) & 0x01);                 // Wait Busy
111     au32DestinationData=DrvSPI32_Read();
112
113     DrvGPIO_SetBit(Flash_PORT,SCS_PIN);          // /CS: de-active
114     return    (au32DestinationData & 0xFFFF);
115 }
116
117 void SpiFlash_ChipErase(void)
118 {
119     unsigned int au32SourceData;
120
121     DrvGPIO_ClrBit(Flash_PORT,SCS_PIN);          // /CS: active
122     DrvSPI32_BitLength(8);                       // configure transaction length as 8 bits
123     // send Command: 0x06, Write enable
124     au32SourceData = sFLASH_CMD_WREN<<24;
125     DrvSPI32_Write(au32SourceData);
126     while((spi_00>>BUF) & 0x01);                 // Wait Busy
127     DrvGPIO_SetBit(Flash_PORT,SCS_PIN);          // /CS: de-active
128
129     DrvGPIO_ClrBit(Flash_PORT,SCS_PIN);          // /CS: active
130     au32SourceData = sFLASH_CMD_CE<<24;          // send Command: 0xC7, Chip Erase
131     DrvSPI32_Write(au32SourceData);
132     while((spi_00>>BUF) & 0x01);    // Wait Busy
133     DrvGPIO_SetBit(Flash_PORT,SCS_PIN);    // /CS: de-active
```

134		
135	SpiFlash_WaitReady();	
136	}	
137		
138	void SpiFlash_SectorErase(unsigned int StartAddress)	
139	{	
140	unsigned int au32SourceData;	
141		
142	DrvGPIO_ClrBit(Flash_PORT,SCS_PIN);	// /CS: active
143	DrvSPI32_BitLength(8);	// configure transaction length as 8 bits
144	// send Command: 0x06, Write enable	
145	au32SourceData = sFLASH_CMD_WREN<<24;	
146	DrvSPI32_Write(au32SourceData);	
147	while((spi_00>>BUF) & 0x01);	// Wait Busy
148	DrvGPIO_SetBit(Flash_PORT,SCS_PIN); // /CS: de-active	
149		
150	DrvGPIO_ClrBit(Flash_PORT,SCS_PIN);	// /CS: active
151	au32SourceData = sFLASH_CMD_SE<<24;	// send Command: 0xD8, Sector Erase
152	DrvSPI32_Write(au32SourceData);	
153	while((spi_00>>BUF) & 0x01); // Wait Busy	
154		
155	DrvSPI32_BitLength(24);	// configure transaction length as 24bits
156	au32SourceData = StartAddress<<8;	
157	DrvSPI32_Write(au32SourceData);	
158	while((spi_00>>BUF) & 0x01);	// Wait Busy
159	DrvGPIO_SetBit(Flash_PORT,SCS_PIN);	// /CS: de-active
160		
161	SpiFlash_WaitReady();	
162	}	
163		
164	void SpiFlash_ReadData(unsigned char *DataBuffer, unsigned int StartAddress, unsigned int ByteCount)	
165	{	
166	unsigned int au32SourceData;	
167	unsigned int au32DestinationData;	
168	unsigned int Counter;	
169		
170	DrvGPIO_ClrBit(Flash_PORT,SCS_PIN); // /CS: active	
171		
172	DrvSPI32_BitLength(8);	// configure transaction length as 8 bits
173	// send Command: 0x03, Read data	
174	au32SourceData = sFLASH_CMD_READ<<24;	
175	DrvSPI32_Write(au32SourceData); //	
176	while((spi_00>>BUF) & 0x01); // Wait Busy	
177		
178	DrvSPI32_BitLength(24);	// configure transaction length as 24bits
179	au32SourceData = StartAddress<<8;	

```
180     DrvSPI32_Write(au32SourceData);    //
181     while((spi_00>>BUF) & 0x01);                  // Wait Busy
182
183     DrvSPI32_BitLength(8);                        // configure transaction length as 8 bits
184     for (Counter = 0; Counter < ByteCount; Counter++)
185     {
186         DrvSPI32_Write(0xFFFFFFFF);               // dump Rx register
187         while((spi_00>>BUF) & 0x01);              // Wait Busy
188
189         au32DestinationData=DrvSPI32_Read();    // dump Rx register
190         DataBuffer[Counter] = (unsigned char) au32DestinationData;
191     }
192
193     DrvGPIO_SetBit(Flash_PORT,SCS_PIN);           // /CS: de-active
194 }
195
196 void SpiFlash_PageProgram(unsigned char *DataBuffer, unsigned int
    StartAddress, unsigned int ByteCount)
197 {
198     unsigned int au32SourceData;
199     unsigned int Counter;
200
201     DrvGPIO_ClrBit(Flash_PORT,SCS_PIN);           // /CS: active
202     DrvSPI32_BitLength(8);                        // configure transaction length as 8 bits
203     // send Command: 0x06, Write enable
204     au32SourceData = sFLASH_CMD_WREN<<24;
205     DrvSPI32_Write(au32SourceData);
206     while((spi_00>>BUF) & 0x01);    // Wait Busy
207     DrvGPIO_SetBit(Flash_PORT,SCS_PIN);    // /CS: de-active
208
209     DrvGPIO_ClrBit(Flash_PORT,SCS_PIN);    // /CS: active
210     // send Command: 0x02, Page program
211     au32SourceData = sFLASH_CMD_WRITE<<24;
212     DrvSPI32_Write(au32SourceData);
213     while((spi_00>>BUF) & 0x01);    // Wait Busy
214
215     DrvSPI32_BitLength(24);                       // configure transaction length as 8 bits
216     au32SourceData = StartAddress<<8;
217     DrvSPI32_Write(au32SourceData);    //
218     while((spi_00>>BUF) & 0x01);    // Wait Busy
220     DrvSPI32_BitLength(8);                        // configure transaction length as 8 bits
221     for (Counter = 0; Counter < ByteCount; Counter++)
222     {
223         // send data to program
224         au32SourceData = DataBuffer[Counter]<<24;
225         DrvSPI32_Write(au32SourceData);
226         while((spi_00>>BUF) & 0x01);    // Wait Busy
```

```
227     }
228     DrvGPIO_SetBit(Flash_PORT,SCS_PIN);                    // /CS: de-active
229     SpiFlash_WaitReady();
230 }
231
232 void InitalSPI(void)
233 {
234     //Set SPI input pin
235 DrvGPIO_Open(Flash_PORT,Flash_MOSI|Flash_SCLK|Flash_CS,E_    //CS=PT2.0 output
    IO_OUTPUT);
236     DrvGPIO_Open(Flash_PORT,Flash_MISO,E_IO_INPUT);      //
237     DrvGPIO_SetPortBits(Flash_PORT,Flash_CS);
238     DrvSPI32_Open(E_DRVSPI_MASTER2,E_DRVSPI_TYPE0,1,3);
239     // Master2,3-wire mode
240     // Type0 : CPHA=0 CPOL=0
241     // 1 : PT1.4=CS PT1.5=CK PT1.6=MISO PT1.7=MOSI
242     // 3 : CLOCK/8
243     DrvSPI32_Enable();
244 }
245
246 void SpiFlash_WaitReady(void)
247 {
248     unsigned int au32SourceData;
249     unsigned int au32DestinationData;
250
251     do
252     {
253         // configure transaction length as 16 bits
254         DrvSPI32_BitLength(16);
256         DrvGPIO_ClrBit(Flash_PORT,SCS_PIN);                // /CS: active
258         au32SourceData = 0x05FF<<16;                       // send Command: 0x05, Read status
                                                               register 1
259         DrvSPI32_Write(au32SourceData);    //
260         while((spi_00>>BUF) & 0x01);    // Wait Busy
261         au32DestinationData=DrvSPI32_Read();               // dump Rx register
262         au32DestinationData = au32DestinationData & 1;
263
264         DrvGPIO_SetBit(Flash_PORT,SCS_PIN);                // /CS: de-active
265     }
266     while (au32DestinationData != 0);                      // check the BUSY bit
267 }
269
270 void Delay(unsigned int num)
271 {
272     for(;num>0;num--)
273         asm("NOP");
274 }
```

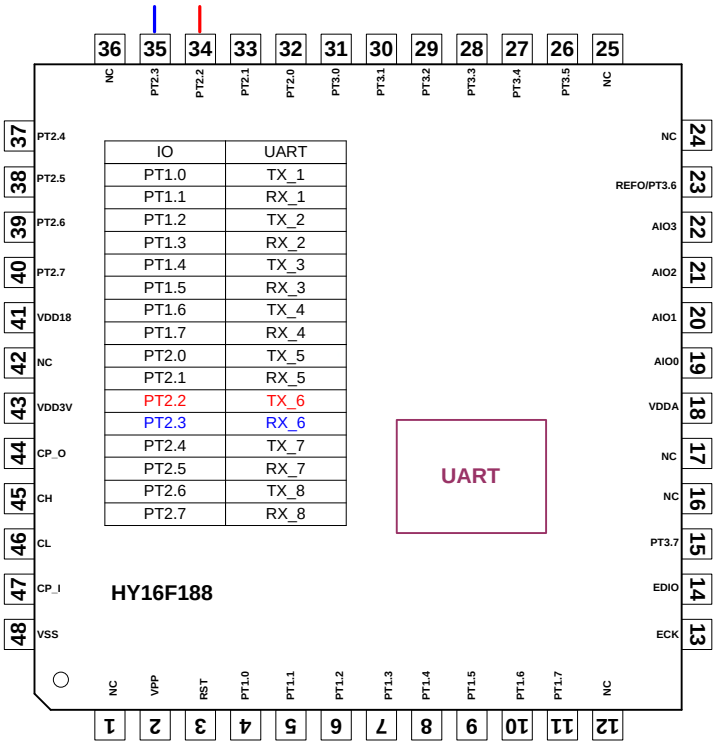
12.通訊 IP(UART)

12.1 範例名稱

通信協定 UART 中斷測試

12.2 範例說明

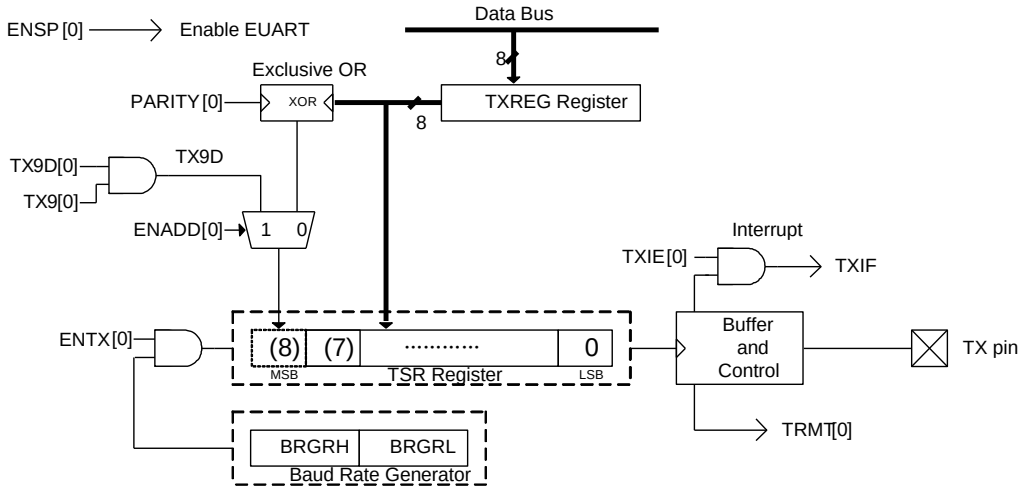
(1)將 TX 與 RX PIN 連接 RS232 相關電路。



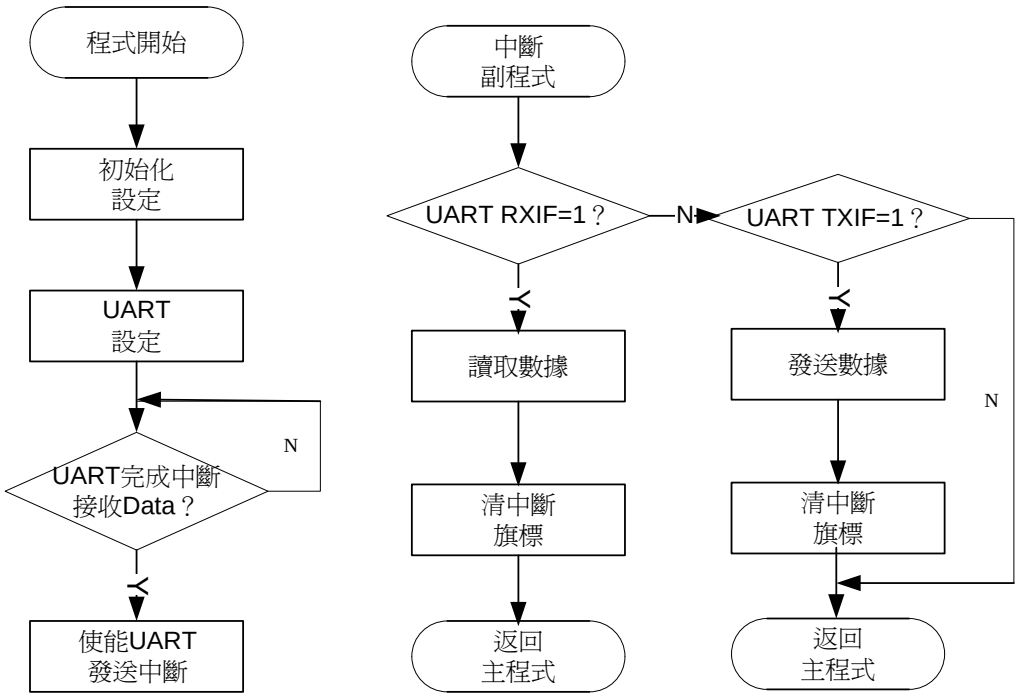
IO	UART
PT1.0	TX_1
PT1.1	RX_1
PT1.2	TX_2
PT1.3	RX_2
PT1.4	TX_3
PT1.5	RX_3
PT1.6	TX_4
PT1.7	RX_4
PT2.0	TX_5
PT2.1	RX_5
PT2.2	TX_6
PT2.3	RX_6
PT2.4	TX_7
PT2.5	RX_7
PT2.6	TX_8
PT2.7	RX_8

12.3 系統設定

EUART TRANSMIT BLOCK DIAGRAM



12.4 軟體流程



12.5 程式說明

```
0
1  #include "HY16F188.h"
2  #include "System.h"
3  #include "DrvGPIO.h"
4  #include "DrvUART.h"
5  #include "DrvI2C.h"
6  #include "DrvHWI2C.h"
7  #include "DrvCLOCK.h"
8  #include "HY2613.h"
9  #include "my define.h"
10
11 volatile typedef union _MCUSTATUS                    //define mcu status bit
12 {
13     char    _byte;
14     struct
15     {
16         unsigned b_ADCdone:1;
17         unsigned b_TMAdone:1;
18         unsigned b_TMBdone:1;
19         unsigned b_TMC0done:1;
20         unsigned b_TMC1done:1;
21         unsigned b_RTCdone:1;
22         unsigned b_UART_TxDone:1;
23         unsigned b_UART_RxDone:1;
24     };
25 } MCUSTATUS;
26
27 #define    UART_PORT E_PT2
28 #define    UART_TXD    BIT2
29 #define    UART_RXD    BIT3
30 #define    UartBufferSize 128
31
32 MCUSTATUS    MCUSTATUSbits;
33 unsigned char UartRxBuffer[UartBufferSize]={0},
34 unsigned char UartTxBuffer[UartBufferSize]={0};
35 unsigned char
   UartTxIndex,UartTxLength,UartRxIndex,UartRxLength;
36 extern unsigned char seg[16];
37
38 void InitalUart(void);
39 void Delay(unsigned int num);
40
41 int main(void)
42 {
43     unsigned int Index;
44
```

```
45    DrvCLOCK_SelectIHOSC (1);///
46    DrvCLOCK_EnableHighOSC(E_INTERNAL,50);                  // Select HAO 4MHz
47    InitalI2C();
48    InitalUart();
49    SYS_EnableGIE(7,0x3F);                                  // Enable GIE(Global Interrupt)
50
51    Ini_Display();
52    ClearLCDframe();
53    Delay(10000);
54    DisplayHYcon();
55    Delay(50000);
56    MCUSTATUSbits._byte = 0;
57
58    while(1)
59    {
60      if(MCUSTATUSbits.b_UART_RxDone==ENABLE)               //the data has been received
61      {
62        for(Index=0;Index<UartBufferSize;Index++)
63        {
64          UartTxBuffer[Index]=UartRxBuffer[Index];
65          if(UartRxBuffer[Index]=='\n')
66            break;
67        }
68
69        MCUSTATUSbits.b_UART_TxDone=DISABLE;
70        UartTxLength=Index;
71        UartTxIndex=NULL;
72        DrvUART_EnableInt(ENABLE,ENABLE);                   //Enable UART Tx Interrupt; Rx Interrupt
73        while(!MCUSTATUSbits.b_UART_TxDone);                //wait   data be sent
74
75        UartRxIndex=NULL;
76        MCUSTATUSbits.b_UART_RxDone=DISABLE;
77      }
78    }
79    return 0;
80  }
81
82
83  /*---------------------------------------------------------------------------*/
84  /* Hardware Communication Interrupt                        */
85  /*I2C Interrupt Service Routines                           */
86  /*---------------------------------------------------------------------------*/
87  void HW0_ISR(void)
88  {
89    unsigned char I2C_Status;
90
91    if(DrvUART_GetRxFlag())                                 // Get UART_RX Interrupt Flag
```

```
92    {
93        UartRxBuffer[UartRxIndex]=DrvUART_Read();          // Read the received data
94        if(UartRxBuffer[UartRxIndex]=='\n')
95        {
96            MCUSTATUSbits.b_UART_RxDone=1;
97        }
98        UartRxIndex++;
99        if(UartRxIndex>=UartBufferSize)
100           UartRxIndex=0;
101       DrvUART_ClrRxFlag();
102   }
103   if(DrvUART_GetTxFlag()&&(MCUSTATUSbits.b_UART_TxDone==0))
104   {
105       DrvUART_Write(UartTxBuffer[UartTxIndex++]);        //send the data
106       DrvUART_ClrTxFlag();    //
107       if(UartTxIndex>=UartTxLength)
108       {
109           DrvUART_EnableInt(DISABLE,ENABLE);             //Disable UART Tx Interrupt
110                                                          //Enable UART Rx Interrupt
111           MCUSTATUSbits.b_UART_TxDone=1;
112       }
113   }
115   void InitalUart(void)
116   {
117   DrvGPIO_Open(UART_PORT,UART_TXD|SDA_BIT|SCL_BIT,E_IO_OUTPUT);
118       DrvGPIO_Open(UART_PORT,UART_RXD,E_IO_INPUT);
119   DrvGPIO_Open(UART_PORT,UART_RXD|UART_TXD|SDA_BIT|SCL_BIT,E_IO_PullHigh);
120
121       //4 : oscillator frequency 4MHz Unit
122       //None parity
123       //8 data bits.
124       //5 : Port 2.2 =TX, Port 2.3 =RX
125   DrvUART_Open(4,B9600,DRVUART_PARITY_NONE,DRVUART_DATABITS_8,5);
126       DrvUART_ClkEnable(1,0);                            //Choose the internal osc as clock source
127       DrvUART_EnableInt(DISABLE,ENABLE);                 //Disable UART Tx Interrupt;
128                                                          //Enable UART Rx Interrupt
129       DrvUART_DisableAutoBaudrate();
130       DrvUART_Close();
131       DrvUART_Enable();
132   }
134   void Delay(unsigned int num)
135   {
136       for(;num>0;num--)
137           asm("NOP");
138   }
```

13.通訊 IP(I2C)

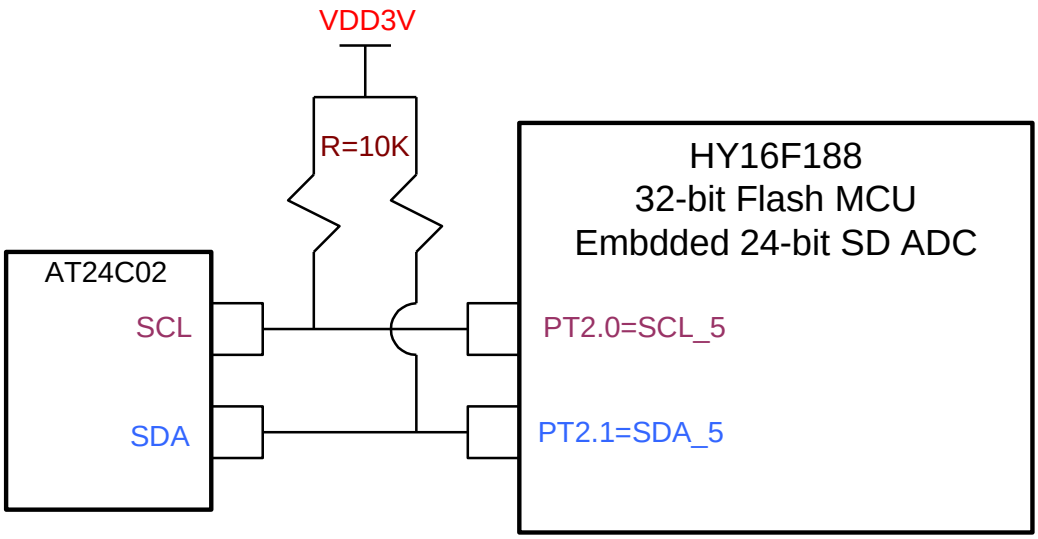
13.1 範例名稱

測試硬體 I2C 中斷

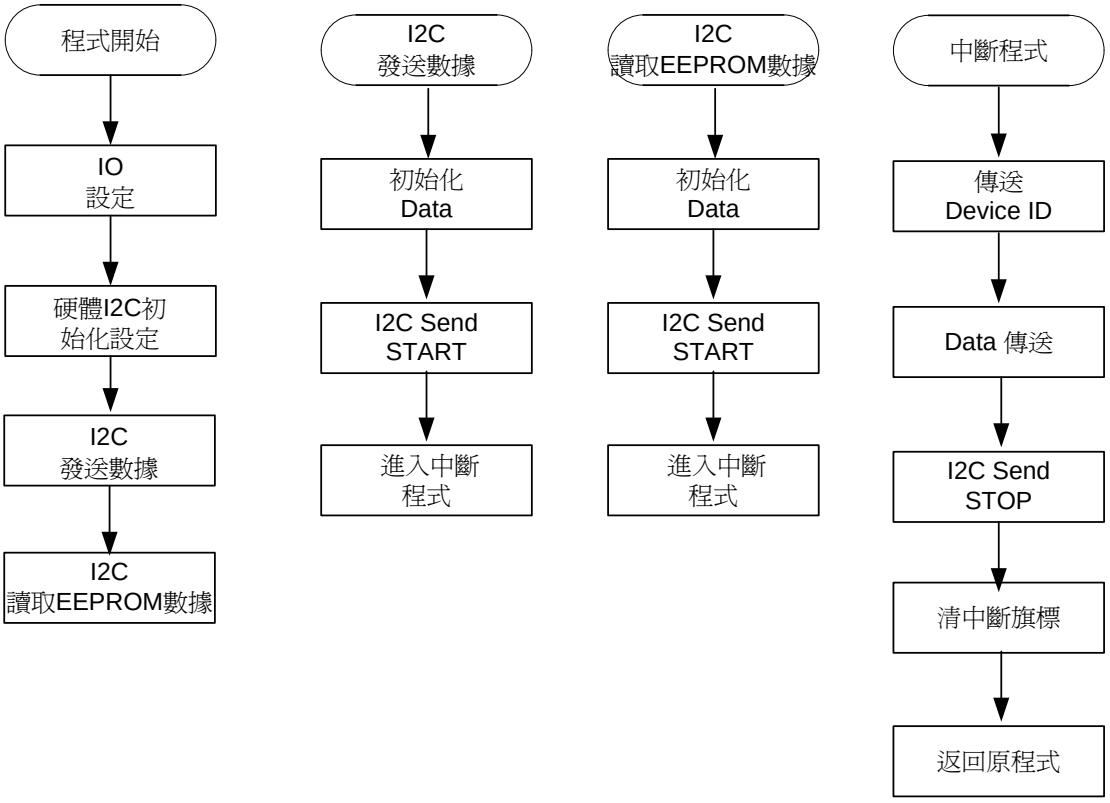
13.2 範例說明

(1)透過 I2C 腳位與 I2C 暫存器設定。
(2)測試硬體 I2C 對 AT24C02 寫入功能

13.3 系統設定



13.4 軟體流程


程式開始
IO
設定
硬體I2C初
始化設定
I2C
發送數據
I2C
讀取EEPROM數據
I2C
發送數據
初始化
Data
I2C Send
START
進入中斷
程式
I2C
讀取EEPROM數據
初始化
Data
I2C Send
START
進入中斷
程式
中斷程式
傳送
Device ID
Data 傳送
I2C Send
STOP
清中斷旗標
返回原程式

13.5 程式說明

0		
1	#include "HY16F188.h"	
2	#include "System.h"	
3	#include "DrvGPIO.h"	
4	#include "DrvTimer.h"	
5	#include "DrvI2C.h"	
6	#include "DrvHWI2C.h"	
7	#include "DrvCLOCK.h"	
8	#include "HY2613.h"	
9	#include "my define.h"	
10		
11	volatile typedef union _MCUSTATUS	//define mcu status bit
12	{	
13	char _byte;	
14	struct	
15	{	
16	unsigned b_ADCdone:1;	
17	unsigned b_TMAdone:1;	
18	unsigned b_TMBdone:1;	
19	unsigned b_TMC0done:1;	
20	unsigned b_TMC1done:1;	
21	unsigned b_RTCdone:1;	
22	unsigned b_UART_TxDone:1;	
23	unsigned b_UART_RxDone:1;	
24	};	
25	} MCUSTATUS;	
26		
27	MCUSTATUS MCUSTATUSbits;	
28	unsigned char Buffer[256];	
29	extern unsigned char seg[16];	
30		
31	void Delay(unsigned int num);	
32		
33	int main(void)	
34	{	
35	unsigned int Count; //	
36	DrvCLOCK_SelectIHOSC (1);	// Select HAO 4MHz
37	DrvCLOCK_EnableHighOSC(E_INTERNAL,10);	
38	InitalI2C();	
39	SYS_EnableGIE(7,0x3F);	// Enable GIE(Global Interrupt)
40		
41	Ini_Display();	

```
42      ClearLCDframe();
43      Delay(10000);
44      DisplayHYcon();
45      Delay(50000);
46      MCUSTATUSbits._byte = 0;
47
48      for(Count=0;Count<256;Count++)
49      {
50          EEPROM_ByteWrite(Count,~Count);          // Write byte
51          Delay(3000);
52          Buffer[Count]=EEPROM_ByteRead(Count);    //Read byte
53          Delay(3000);
54
55      }
56
57      while(1);
58      return 0;
59  }
60
61  /*---------------------------------------------------------------------------*/
62  /* Hardware Communication Interrupt                                          */
63  /*I2C Interrupt Service Routines                                             */
64  /*---------------------------------------------------------------------------*/
65  void HW0_ISR(void)
66  {
67      unsigned char I2C_Status;
68
69      if(DrvI2C_ReadIntFlag()==E_DRVI2C_INT)     // Get I2C Interrupt Flag
70      {
71          I2C_Status=DrvI2C_GetStatusFlag();     // Get I2C Status Flag
72          switch(I2C_Status)
73          {
74              case 0x90: //MACTFlag+RWFlag
75                  { /* START has been transmitted */
76                      DrvI2C_WriteData(I2C_TARGET);          // Send Slave Address & R/W Bit
77                      DrvI2C_Ctrl(0,0,0,0);                  // Clear all I2C flag
78                      break;
79                  };
80              case 0x84: //MACTFlag+ACKFlag
81                  { /* Slave A + W has been transmitted. ACK has been
    received. */
82                      DrvI2C_WriteData(Sendbuf[DataTxIndex++]);  // Send Data to Slave
83                      DrvI2C_Ctrl(0,0,0,0);                      // Clear all I2C flag
84                      break;
85                  };
86              case 0x80: //MACTFlag
87                  { /* Slave A + W has been transmitted. NACK has been
    received. */
```

88	DrvI2C_WriteData(Sendbuf[DataTxIndex++]);	// Send Data to Slave
89	DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag	
90	break;	
91	};	
92	case 0x30:	
93	{/*A STOP has been transmitted.*/	
94	DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag	
95	EndFlag=1;	
96	break;	
97	};	
98	case 0x8C: // MACTFlag+DFFlag+ACKFlag	
99	{ /* DATA has been transmitted and ACK has been received */	
100	if(DataTxIndex<DataTxLen)	
101	{	
102	DrvI2C_WriteData(Sendbuf[DataTxIndex++]);	//Send Data to Slave
103	DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag	
104	}	
105	else	
106	{	
107	if(I2C_RW == WRITE)	
108	{	
109	DrvI2C_Ctrl(0,1,0,0);	// I2C as master sends STOP signal
110	EndFlag=1;	
111	}	
112	else if(I2C_RW == READ)	
113	DrvI2C_Ctrl(1,0,0,0);	// I2C as master sends START signal
114	DataTxIndex=0;	
115	}	
116	break;	
117	};	
118	case 0x88:	//MACTFlag+DFFlag
119	{ /* DATA has been transmitted and NACK has been received */	
120	DrvI2C_Ctrl(0,1,0,0);	// I2C as master sends STOP signal
121	DataTxIndex=0;	
122	EndFlag=1;	
123	break;	
124	};	
125	case 0xB0:	
126	{ /* A repeated START has been transmitted. */	
127	DrvI2C_WriteData(I2C_TARGET | READ);	//Send Slave Address & R/W Bit
128	DrvI2C_Ctrl(0,0,0,0);	// Clear all I2C flag
129	break;	
130	}	
131	case 0x94:	//MACTFlag+RWFlag
132	{ /* Slave A + R has been transmitted. ACK has been received. */	

```
133                     DrvI2C_Ctrl(0,0,0,1);                   // Set ACK bit
134                     break;
135                 };
136             case 0x9C:                                      //MACTFlag+RWFlag+DFFlag+ACKFlag
137                 { /* Data byte has been received. ACK has been
    transmitted. */
138                     if(DataRxLen>DataRxIndex)
139                     {
140                         Recbuf[DataRxIndex++]=DrvI2C_ReadData();
141                         DrvI2C_Ctrl(0,0,0,0);               // Clear all I2C flag
142                     }
143                     else
144                     {
145                         Recbuf[DataRxIndex++]=DrvI2C_ReadData();
146                         DrvI2C_Ctrl(0,1,0,0);               // I2C as master sends STOP signal
147                         EndFlag=1;
148                     }
149                     break;
150                 };
151             case 0x98:                                      //MACTFlag+RWFlag+DFFlag
152                 { /* Data byte has been received. NACK has been
    transmitted. */
153                     DrvI2C_Ctrl(0,1,0,0);                   // I2C as master sends STOP signal
154                     EndFlag=1;
155                     break;
156                 };
157             default:
158                 {
159                     DrvI2C_Ctrl(0,0,0,0);                   // Clear all I2C flag
160                     EndFlag=1;
161                     break;
162                 };
163         }
164         DrvI2C_ClearEIRQ();
165         DrvI2C_ClearIntFlag(2);                             // Clear I2C Interrupt Flag(I2CIF)
166         SYS_EnableGIE(7,0X3F);                              // Enable GIE(Global Interrupt)
167     }
168     if(DrvI2C_ReadIntFlag()==E_DRVI2C_ERROR_INT)            //Get I2C Error Interrupt Flag
169     {
170         EndFlag=1;
171         DrvI2C_ClearIRQ();
172         DrvI2C_ClearIntFlag(2);                             // Clear I2C Interrupt Flag(I2CEIF)
173         DrvI2C_Ctrl(0,1,0,0);                               // I2C as master sends STOP signal
174         SYS_EnableGIE(7,0X3F);                              // Enable GIE(Global Interrupt)
175     }
176 }
177
178 void EEPROM_ByteWrite(unsigned int addr, unsigned char dat)
```

```
{
    I2C_TARGET    = EEPROM_ADDR;                //Device_ID_Addr=0XA0
#if defined(Small)
    Sendbuf[0]    = (addr&0xFF);                //24Cxx Memory Address
    Sendbuf[1]    = dat;                        //24Cxx Memory Data
    DataTxLen=2;
#else
    Sendbuf[0]    = ((addr>>8)&0xFF);           //24Cxx Memory Address
    Sendbuf[1]    = (addr&0xFF);                //24Cxx Memory Address
    Sendbuf[2]    = dat;                        //24Cxx Memory Data
    DataTxLen=3;
#endif

    DataTxIndex=0;
    EndFlag=0;
    I2C_RW = WRITE;                             //
    DrvI2C_Ctrl(1,0,0,0);                       /* I2C as master sends START signal */
    asm("NOP");
    while (EndFlag == 0);                       /* Wait I2C Tx Finish */
    EndFlag = 0;
}

unsigned char EEPROM_ByteRead(unsigned int addr)
{
    unsigned char retval;                       // Holds the return value

    I2C_TARGET    = EEPROM_ADDR;                //Device_ID_Addr=0XA0
#if defined(Small)
    Sendbuf[0]    = (addr&0xFF);                //24Cxx Memory Address
    DataTxLen=1;
#else
    Sendbuf[0]    = ((addr>>8)&0xFF);           //24Cxx Memory Address
    Sendbuf[1]    = (addr&0xFF);                //24Cxx Memory Address
    DataTxLen=2;
#endif

    DataTxIndex=0;
    DataRxLen=1;
    DataRxIndex=0;
    EndFlag=0;
    I2C_RW = READ;                              //
    DrvI2C_Ctrl(1,0,0,0);                        /* I2C as master sends START signal */
    asm("NOP");
    while (EndFlag == 0);                       /* Wait I2C Tx Finish */
    EndFlag = 0;

    retval=Recbuf[0];
```

```
226     return retval;}
227
228 void Delay(unsigned int num)
229 {
230     for(;num>0;num--)
231         asm("NOP");
232 }
```

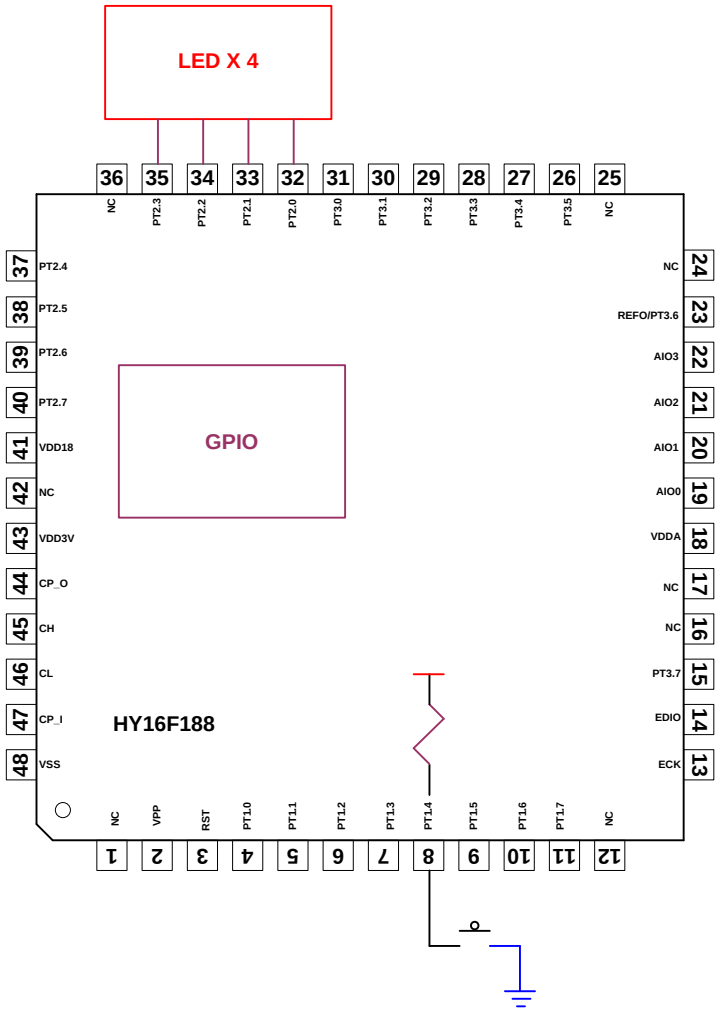
14.其他 IP(GPIO)

14.1 範例名稱

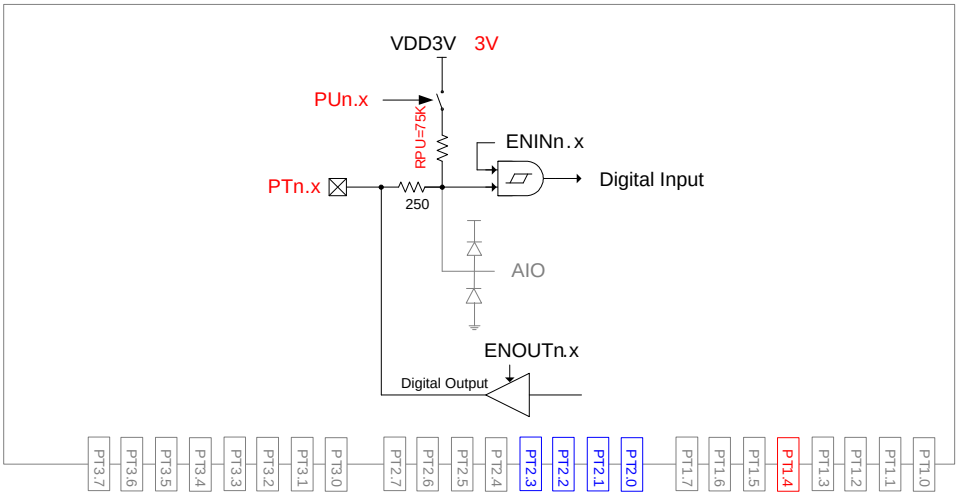
GPIO 設定方式

14.2 範例說明

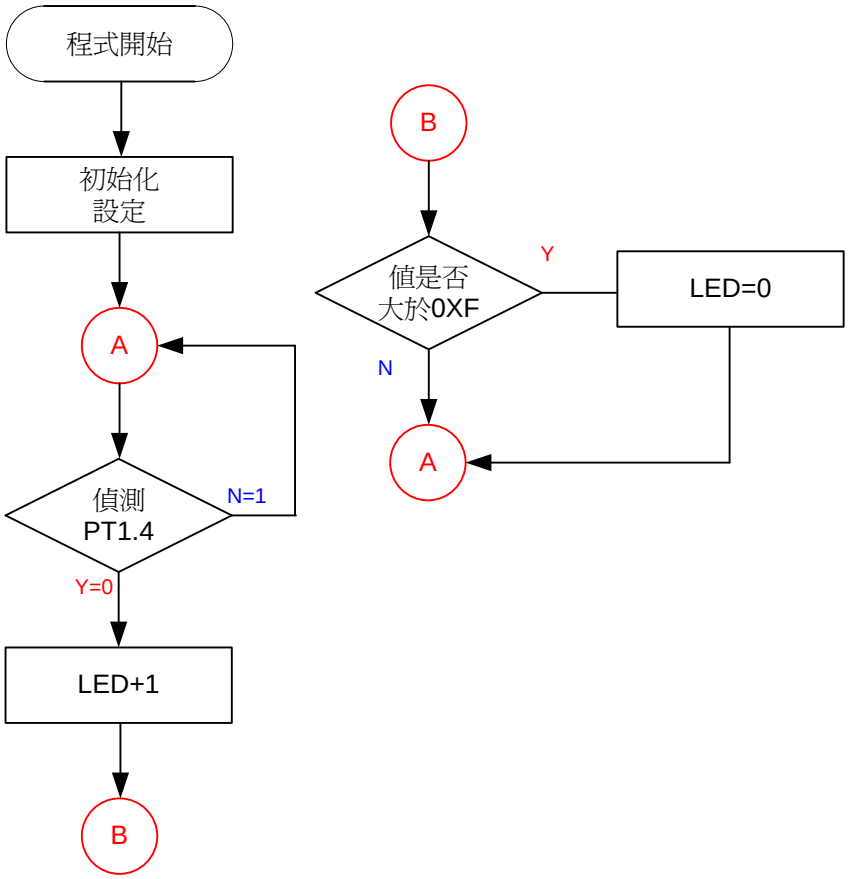
(1)每按 1 次 PT1.4，LED 加 1 顯示。
(2)若 LED 加到 0XF，則歸 0。回到每按 1 次 PT1.4，LED 加 1 顯示。



14.3 系統設定



14.4 軟體流程



14.5 程式說明

```
00
01  #include "HY16F188.h"
02
03  void Delay (unsigned int num);
04
05  int main(void)
06  {
07      DrvCLOCK_SelectIHOSC (1);///                    // Select HAO 4MHz
08      DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
09      unsigned int i=0,j=0;
11      DrvGPIO_Open(E_PT1,0X10,E_IO_INPUT);            //Set PT1_4 INPUT
12      DrvGPIO_Open(E_PT1,0X10,E_IO_PullHigh);         //Enable PT1 4 pull hi R
13      DrvGPIO_Open(E_PT2,0X0F,E_IO_OUTPUT);           //Set PT2_0~3 OUTPUT
14      while(1)
15      {
16          i=DrvGPIO_GetBit(E_PT1,4);                  //Read PT1.4 high or low
17          if(i==0)                                    //IF PT1.4 is low
18          {
19              DrvGPIO_SetPortBits(E_PT2, j++);        //J++
20              if(j>0X0F)j=0X00;                       //IF J>0XF J=0
21          }
22      Delay(0X8000);
23  }
24
25  void Delay(unsigned int num)
26  {
27      int a;for(a=0;a<=num;a++);                      //Delay loop
28  }
29
```

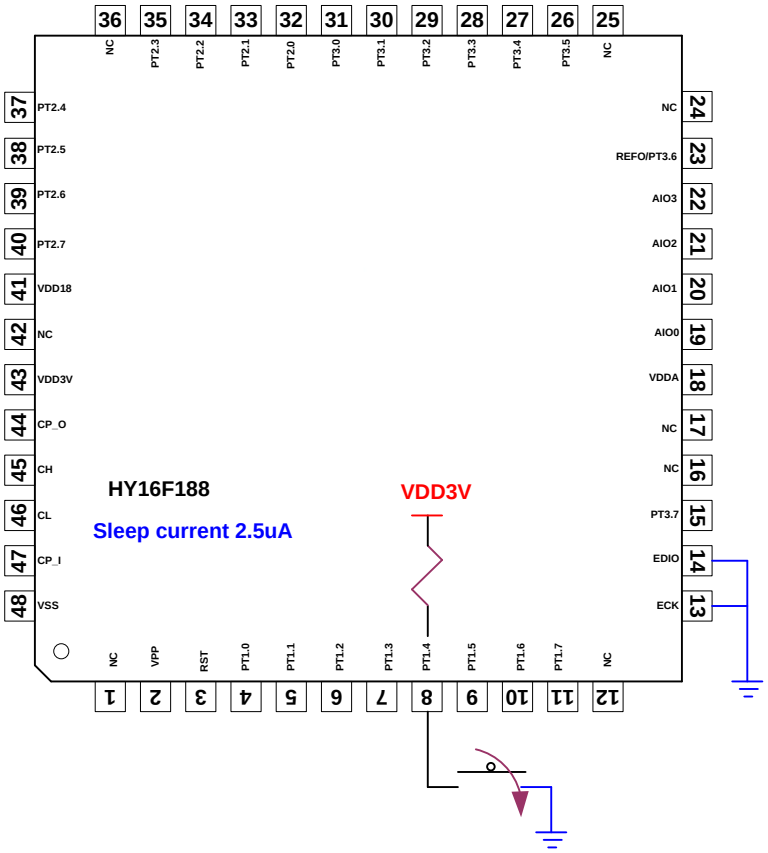
15.其他 IP(Power)

15.1 範例名稱

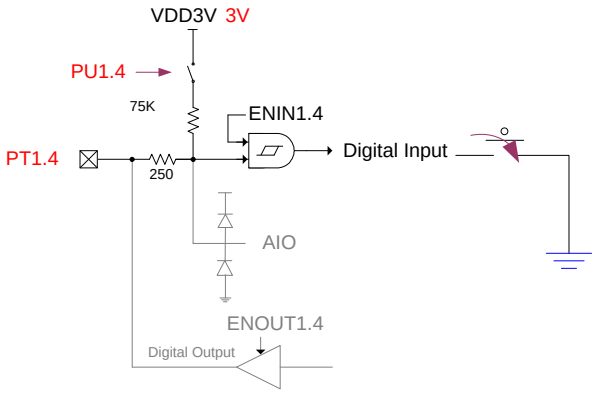
測試 Sleep 與 Idle 電流

15.2 範例說明

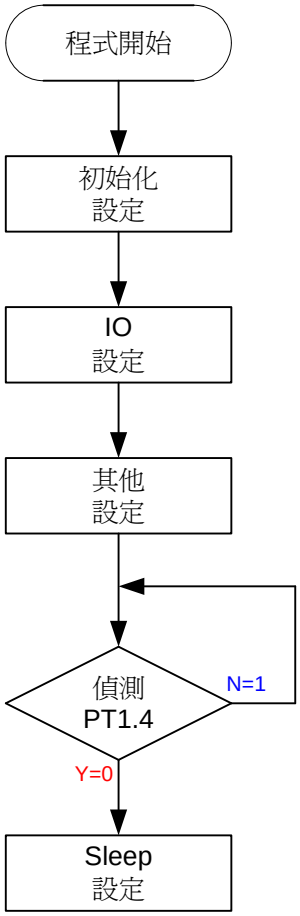
(1)各腳位不浮接,例如 EDM 的 2PIN(第 13 與 14 腳位),可接地可讓 Sleep 電流低於 2.5(uA)。
(2)需按下 PT1.4 後，晶片才會進入 Sleep。



15.3 系統設定



15.4 軟體流程



此程式進 Sleep 後可量到低於 2.5(uA)的電流。但要重新喚醒，只能用 Power ON Reset 其餘喚醒方式。

15.5 程式說明

```
00
01  #include "HY16F188.h"
02
03  void Delay (unsigned int num);
04
05  int main(void)
06  {
07      char i;
08      DrvGPIO_ClearIntFlag(E_PT2,0XFF);          //clear PT2.0~7 interrupt flag
09
11      DrvGPIO_Open(E_PT1,0X3F,E_IO_OUTPUT);      //SET PT1.0~6 AS OUTPUT PT1.6~7 INPUT
12      DrvGPIO_Open(E_PT3,0XFF,E_IO_OUTPUT);      //SET PT3 AS OUTPUT
13
14      DrvGPIO_Open(E_PT1,0XC0,E_IO_INPUT);       //PT1_6~7 set input
15      DrvGPIO_Open(E_PT1,0XC0,E_IO_PullHigh );   //PT1_6~7 pull internal high R enable
16
17      DrvGPIO_Open(E_PT2,0XFF,E_IO_INPUT);       //PT2_0~7 set input
18      DrvGPIO_Open(E_PT2,0XFF,E_IO_PullHigh );   //PT2_0~7 pull internal high R enable
19      DrvCLOCK_SelectIHOSC (0);
20      DrvPMU_LDO_LowPower(1);                    //SET low power mode
21      i=DrvGPIO_GetBit(E_PT1,7);                 //read PT1.7 pin high or low
22
23      DrvCLOCK_EnableHighOSC(E_INTERNAL,50);     //select HSRC 2MHz
24      DrvGPIO_ClkGenerator(E_HS_CK,1);           //GPIO CLK enable
25                                                 // trigger method is negative edge
26      DrvGPIO_IntTrigger(E_PT2,0XFF,E_N_Edge);   //PT2.0~7 interrupt
27
28      DrvGPIO_Open(E_PT2,0XFF,E_IO_IntEnable);   //PT2.0~7 interrupt enable
29
30      SYS_EnableGIE(7,0x3F);                     //Enable GIE (Global Interrupt Enable)
31
32
33
34
35
```

36		
37	while(1)	
38	{	
39	i=DrvGPIO_GetBit(E_PT1,7);	//read PT1.7 pin high or low
40	if(i==0)	//if PT1.7 low
41	{	
42	clk_08=0XFF80FF03;	//[CPU/2][RTC Clock On]
43	clk_00=0XFF00FF04;	//[Internal OSC OFF][32768]
44	asm("NOP");	
45	SYS_LowPower(0); ;	//sleep mode
46	// SYS_LowPower(1);	//idle mode
47		
48	asm("NOP");	
49	}	
50	Delay(0X8000);	
51	}	
52	}	
53	void HW5_ISR(void)	
54	{	
55	DrvGPIO_ClearIntFlag(E_PT2,0XFF);	//Clear PT2 interrupt flag
56	}	
57	void Delay(unsigned int num)	
58	{	
59	int a;for(a=0;a<=num;a++);	
60	}	

SYS Base Address + 0X04 (0X40104)																
Symbol	SYS0 (SYS Control Register 0)															
Bit	[31]	[30]	[29]	[28]	[27]	[26]	[25]	[24]	[23]	[22]	[21]	[20]	[19]	[18]	[17]	[16]
Name	RSV															
RW	R-0															
Bit	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Name	MASK								-	-	F_{CRst}	IDLE	$F_{SLP/IDLE}$	F_{WDog}	F_{RST}	F_{BOR}
RW	R0W-0								-	-	RW0-0					RW0-1

Bit	Name	Descriptions	
Bit[4]	IDLE	IDLE Mode Control Register	
		0	Sleep Mode
		1	IDLE Mode

16.數位 IP(RTC)

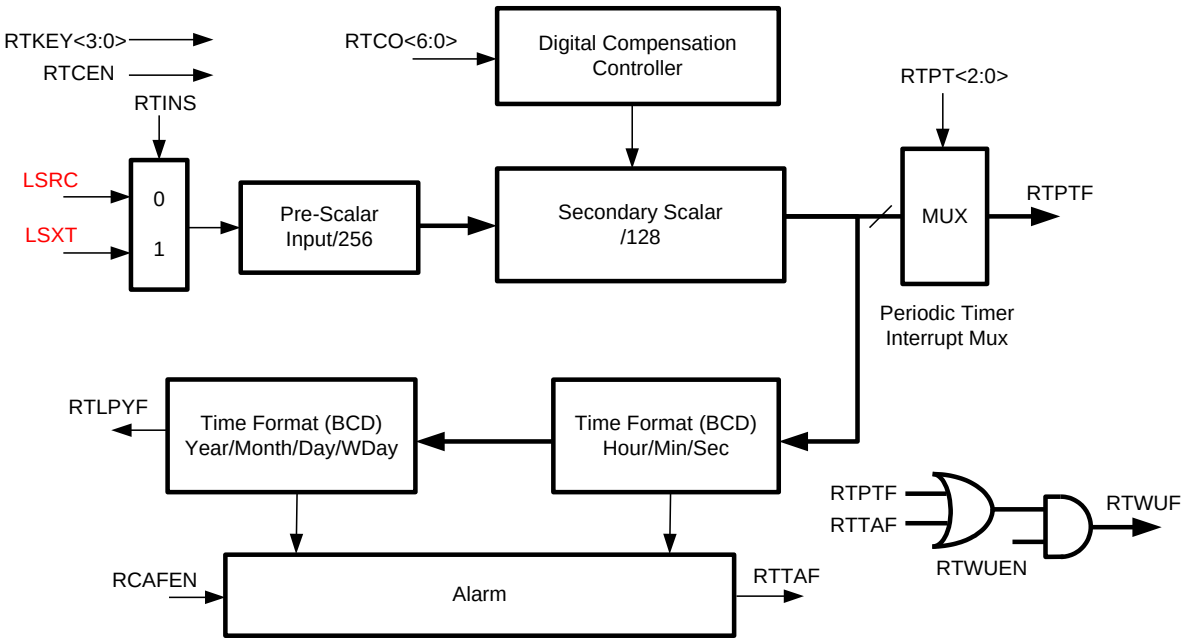
16.1 範例名稱

測試硬體 RTC

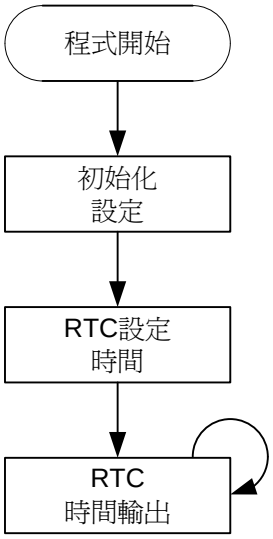
16.2 範例說明

測試硬體 RTC 萬年曆功能

16.3 系統設定



16.4 軟體流程



16.5 程式說明

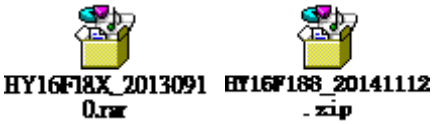
```
0
1  #include "HY16F188.h"
2  #include "System.h"
3  #include "DrvGPIO.h"
4  #include "DrvI2C.h"
5  #include "DrvHWI2C.h"
6  #include "DrvCLOCK.h"
7  #include "DrvRTC.h"
8  #include "HY2613.h"
9  #include "my define.h"
10
11 volatile typedef union _MCUSTATUS                    //define mcu status bit
12 {
13     char    _byte;
14     struct
15     {
16         unsigned b_ADCdone:1;
17         unsigned b_TMAdone:1;
18         unsigned b_TMBdone:1;
19         unsigned b_TMC0done:1;
20         unsigned b_TMC1done:1;
21         unsigned b_RTCdone:1;
22         unsigned b_UART_TxDone:1;
23         unsigned b_UART_RxDone:1;
24     };
25 } MCUSTATUS;
26
27 void InitalRTC(void);
28 int ComputeWeek(int TempYear, int TempMonth, int TempDay);
29 void Delay(unsigned int num);
30
31
32 int main(void)
33 {
34     unsigned int TimerDtata;
35     DrvCLOCK_SelectIHOSC (1);                        // Select HAO 4MHz
36     DrvCLOCK_EnableHighOSC(E_INTERNAL,10);
37     InitalI2C();
38     InitalRTC();
39     SYS_EnableGIE(7,0x3F);                           // Enable GIE(Global Interrupt)
40
41     Ini_Display();
42     ClearLCDframe();
43     Delay(10000);
44     DisplayHYcon();
```

```
45      Delay(50000);
46      MCUSTATUSbits._byte = 0;
47
48      while(1)
49      {
50          if(MCUSTATUSbits.b_RTCdone==1)
51          {
52              DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
53  TimerDtata=sCurTime.u32cSecond+sCurTime.u32cMinute*100+sCurTime
    .u32cHour*10000;
54              LCD_DATA_DISPLAY(TimerDtata);          //LCD display the data of
                                                       TimerData
55              MCUSTATUSbits.b_RTCdone=0;
56          }
57      }
58      return 0;
59  }
60
62  /* WDT & RTC & Timer A/B/C Interrupt Service Routines     */
64  void HW1_ISR(void)
65  {
66      if(DrvRTC_ReadState()&0x4)
67      {
68          DrvRTC_ClearState(E_DRVRTC_CLEAR_ALL);
69          DrvRTC_ClearIntFlag();                     // clear RTC interrupt flag
70          MCUSTATUSbits.b_RTCdone=1;
71      }
72  }
73
74  void InitalRTC()
75  {
76      clk_08=0x8080ff04;                              //RTC CLK;
77  //rtc_00=0xff60ff00;
78      DrvRTC_WriteEnable();                           //RTC KEY;
79      DrvRTC_ClockSource(0);
80      DrvRTC_AlarmDisable();
81      DrvRTC_PeriodicTimeDisable();
82
83      DrvRTC_ClearState(2);                          //TAF=0    PTF=0
84      rtc_00=0x0A0A0000;                              //LPYF=0    WUF=0
85      DrvRTC_Enable();
86
87      DrvRTC_WriteEnable();
88      //clk_00=0x0404;    //32768 ON                ///
89      Delay(0x10);
90      DrvRTC_ClockSource(1);    //32768Hz             ///
91      DrvRTC_Enable();
92      rtc_00=0x0400;    //DrvRTC_HourFormat(0);      //HRF=1:12hour 0:24hour
```

93	S_DRVRTC_TIME_DATA_T sCurTime;	//setting start
94	DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);	
95	//DRVRTC_CURRENT_TIME or Alarm Time	
96	sCurTime.u8cClockDisplay=1;	//DRVRTC_CLOCK_12//DRVRTC_CLOCK_24
97	sCurTime.u8cAmPm=0;	//DRVRTC_AM//DRVRTC_PM
98	sCurTime.u32cSecond=50;	//設定_秒
99	sCurTime.u32cMinute=59;	//設定_分
100	sCurTime.u32cHour=10;	//設定_小時
101	sCurTime.u32cDay=18;	//設定_日期
102	sCurTime.u32cMonth=3;	//設定_月份
103	sCurTime.u32Year=2014;	//設定_年
104	sCurTime.u32cDayOfWeek=ComputeWeek(sCurTime.u32Year,sCurTime.u32cMonth,sCurTime.u32cDay);;	
105	sCurTime.u8IsEnableWakeUp=0;	//WK
106	DrvRTC_Write(DRVRTC_CURRENT_TIME,&sCurTime);	
107	Delay(0x10);	
108		
109	DrvRTC_PeriodicTimeEnable(E_DRVRTC_1_4_SEC);	//set 1/4
110	DrvRTC_ClearIntFlag();	
111	DrvRTC_EnableInt();	/* Enable RTC interrupt */
112	}	
113		
114	int ComputeWeek(int TempYear, int TempMonth, int TempDay)	
115	{	
116	int TempWeek;	
117	if (TempMonth >= 3)	
118	{	
119	TempMonth = TempMonth - 2;	
120	}	
121	else	
122	{	
123	TempMonth = TempMonth + 10;	
124	TempYear--;	
125	}	
126		
127	TempWeek = TempYear + (int)(TempYear / 4) - (int)(TempYear / 100) +(int)(TempYear / 400) + (int)(2.6 * TempMonth - 0.2) + TempDay;	
128	TempWeek = TempWeek - 7*(int)(TempWeek / 7);	
129	return (TempWeek);	
130	}	
131		
132	void Delay(unsigned int num)	
133	{ for(;num>0;num--) asm("NOP");}	
134		

17.程式附件

	數位 IP	類比 IP	通訊 IP	其他 IP
01	TimerA	8-bit DAC	Hardware 32-bit SPI	GPIO
02	TimerB	R2R OPAMP	Hardware UART	Sleep Mode
03	TimerC	24-bit SD ADC	Hardware I2C	Idle Mode
04	WDT	Analog CMP	-	-
05	PWM	-	-	-
06	Hardware RTC	-	-	-



18.修訂紀錄

以下描述本檔差異較大的地方，而標點符號與字形的改變不在此描述範圍。

版本	頁次	變更摘要	日期
V01	ALL	初版發行	2013/03/20
V02	ALL	2 版發行	2013/07/20
V03	ALL	3 版發行 (1)新增 RTC 等 IP 說明 (2)刪除 BOR3 引腳	2013/09/11
V04	ALL	3 版發行 (1)腳位名稱更動 VDD->VDD18	2013/12/13
V05	ALL	(1)更新 Timer 等 IP 內容 (2)PIN 3 腳標識由 PT4.0/RST 改為 RST (3)RTC 修改 rtc_00=0x0400 即 HRF=0 為 24 小時制	2014/11/12