HY17M26

HYCON IP 使用說明書

Table of Contents

注意：

1、 本說明書中的內容，隨著產品的改進，有可能不經過預告而更改。請客戶及時到本公司網站下載更新 http://www.hycontek.com。

2、 本規格書中的圖形、應用電路等，因第三方工業所有權引發的問題，本公司不承擔其責任。

3、 本產品在單獨應用的情況下，本公司保證它的性能、典型應用和功能符合說明書中的條件。當使用在客戶的產品或設備中，以上條件我們不作保證，建議客戶做充分的評估和測試。

4、 請注意輸入電壓、輸出電壓、負載電流的使用條件，使 IC 內的功耗不超過封裝的容許功耗。對於客戶在超出說明書中規定額定值使用產品，即使是瞬間的使用，由此所造成的損失，本公司不承擔任何責任。

5、 本產品雖內置防靜電保護電路，但請不要施加超過保護電路性能的過大靜電。

6、 本規格書中的產品，未經書面許可，不可使用在要求高可靠性的電路中。例如健康醫療器械、防災器械、車輛器械、車載器械及航空器械等對人體產生影響的器械或裝置，不得作為其部件使用。

7、 本公司一直致力於提高產品的品質和可靠度，但所有的半導體產品都有一定的失效概率，這些失效概率可能會導致一些人身事故、火災事故等。當設計產品時，請充分留意冗餘設計並採用安全指標，這樣可以避免事故的發生。

8、 本規格書中內容，未經本公司許可，嚴禁用於其他目的之轉載或複製。

1. 文件說明

HYCON IP(IP 矽智財 Intellectual Property 縮寫)，代表紘康 8 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 HY17M26，SOC 晶片內數位、類比及通訊等其它周邊 IP 做使用說明。

主要包含以下 4 大分類:

(1)數位 IP：TimerA/TimerB/WDT/PWM/GPIO

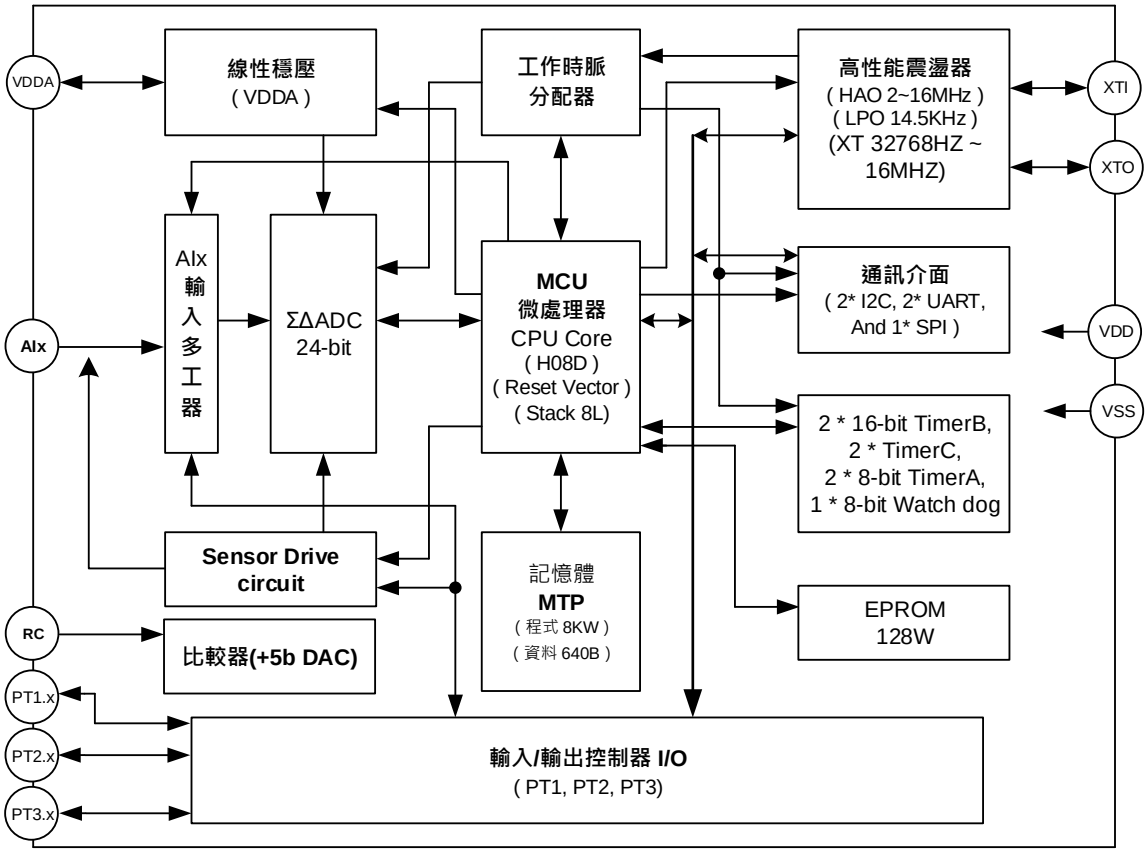
(2)類比 IP：ADC/CMP

(3)通訊 IP：Hardware UART/ Hardware I2C / Hardware SPI

(4)其他 IP：Power Management

2. 晶片說明

HY17M26 各功能 IP 基礎使用說明。



(01)採用 8-Bit RISC-Like 微控制器。

(02)電壓操作範圍 1.9~5.5V(類比電源沒開啟情況下)，以及-40℃~85℃工作溫度範圍。

(03)支援外部 16MHz 石英震盪器或內部 16MHz 高精度 RC 震盪器，

(04) 8K words MTP Program Memory (讀寫次數：100 cycles)、128 words EPROM Data Memory(讀寫次數：100 cycles)

(05)資料記憶體 640 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) ADC Gain: x1/4, x1/2, x1,x2,x4,x8,x16.。

(7.2)內置溫度感測器 TPS。

(08)16-bit Timer A

(09)16-bit Timer B 模組具 PWM 波形產生功能

(10)硬體串列通訊 2* I2C/2* UART/SPI 模組

3. 數位 IP(TMA)

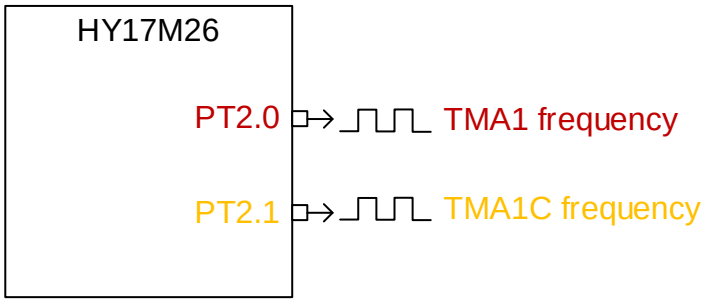
3.1. 範例名稱

TMA

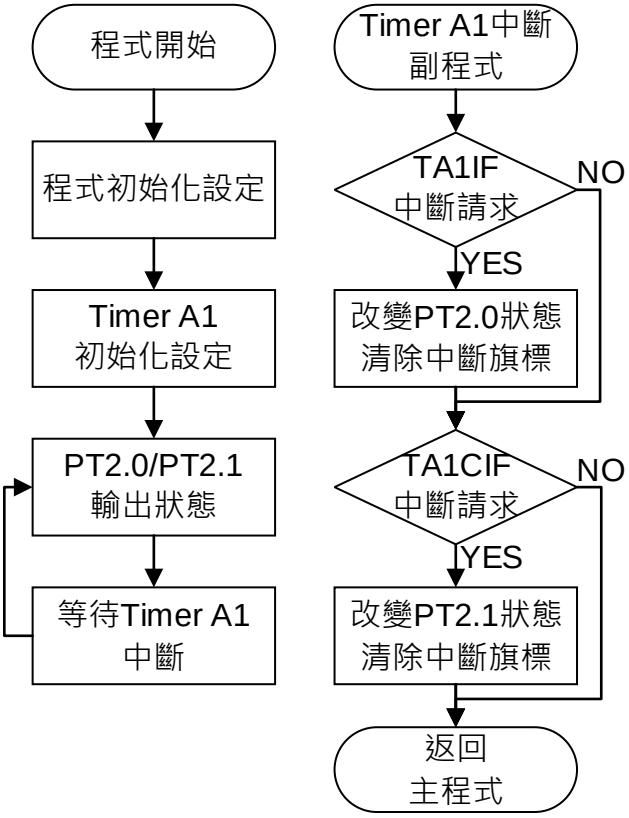
3.2. 範例說明

(1) Timer A 使用方式與說明。
(2) 設置系統工作頻率。
(3) 程式做好 Timer A 初始化動作與設置相關 Timer A 計數溢出值，開啟 GIE 等待 Timer A 中斷發生。Timer A 計數溢出值可決定 Timer A 進入中斷的時間。
(4) 每進一次 TA1 中斷，代表 TMA1R 加一，並改變 PT2.0 輸出狀態觀察變化速度。
(5) 每進一次 TA1C 中斷，代表 TMA1R 計數值等於 TMA1C，改變 PT2.1 輸出狀態觀察變化速度，可透過設置 TMA1C 暫存器增加 TMA 整體計數時間。

3.3. 範例配置



3.4. 軟體流程


程式開始
程式初始化設定
Timer A1
初始化設定
PT2.0/PT2.1
輸出狀態
等待Timer A1
中斷
Timer A1中斷
副程式
TA1IF
中斷請求
NO
YES
改變PT2.0狀態
清除中斷旗標
TA1CIF
中斷請求
NO
YES
改變PT2.1狀態
清除中斷旗標
返回
主程式

3.5. 程式碼

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * TMA_Demo                                                                    *
 * ---------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                             *
 * http://www.hycontek.com/                                                    *
 *                                                                             *
 *                                                                             *
 * Program Description:                                                        *
 *                                                                             *
 *           HY17M26                                                           *
 *--------------------                                                         *
 *                          |                                                  *
 *                    PT2.0|->_|-|_ TMA                                        *
 *                          |                                                  *
 *                    PT2.1|->_|-|_ TMAC                                       *
 *                          |                                                  *
 *--------------------                                                         *
 *                                                                             *
 * For External Input                                                          *
 * IC Body: HY17M26                                                            *
 * Project Name : TMA                                                          *
 * Created Date : 2021/6/4 By Cyril                                            *
 *                                                                             *
 *******************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <INT.h>
#include <CLK.h>
#include <GPIO.h>
#include <RST.h>
#include <TMR.h>

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

/*---------------------------------------------------------------------------*/
```

```
/* Function PROTOTYPES                                                    */
/*------------------------------------------------------------------------*/
void TMAInit(void);
void CLOCKInit(void);
void GPIOInit(void);
/*------------------------------------------------------------------------*/
/* Global CONSTANTS                                                       */
/*------------------------------------------------------------------------*/
unsigned int TMA1R_Count;
unsigned int TMA1C_Count;
/*------------------------------------------------------------------------*/
/* Main Function                                                          */
/*------------------------------------------------------------------------*/
void main(void)
{
      CLOCKInit();
      GPIOInit();
      TMAInit();
      TMA1R_Count=0;
      TMA1C_Count=0;

      GIE_Enable();

      while(1);

}

/*------------------------------------------------------------------------*/
/* Interrupt Service Routines                                             */
/*------------------------------------------------------------------------*/
void ISR(void) __interrupt
{
       if(TA1IF_IsFlag())          //TA1IF=1,TMA1R++
      {
           PT2= PT2^PT20_MSK;    //PT2.0 toggle
           TMA1R_Count++;
           TA1IF_ClearFlag();
      }

       if(TA1CIF_IsFlag())           //TMA1C=TMA1R ,then TA1CIF=1
      {
           PT2= PT2^PT21_MSK;    //PT2.1 toggle
           TMA1C_Count++;
           TMA1_ClearTMA1R();    //clear TMA1R
           TA1CIF_ClearFlag();
       }
}

/*------------------------------------------------------------------------*/
/* CLOCK Init Function                                                    */
/*------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif
```

```
#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

   CLK_DMSCKSelect(DMS_DHSCKDIV2);    //DMS_CK= HAO/1/2
}

/*---------------------------------------------------------------------------*/
/* TMA Init Function                                                         */
/*---------------------------------------------------------------------------*/
void TMAInit(void)
{
   TMA1_Open(TMAS_DMSCK,DTMA_TMACKDIV2,100); //DTMA1_CK=TMA1_CK/256/2
                                             //TMA1C_Flag=TMA1_Flag*100
   TA1CIF_ClearFlag();
   TA1CIE_Enable();
   TA1IF_ClearFlag();
   TA1IE_Enable();
}

/*---------------------------------------------------------------------------*/
/* GPIO Init Function                                                        */
/*---------------------------------------------------------------------------*/
void GPIOInit(void)
{
   GPIO_PT2OutputMode(PT20);
   GPIO_PT2DigitalEnable(PT20);
   GPIO_PT2OutputLow(PT20);

   GPIO_PT2OutputMode(PT21);
   GPIO_PT2DigitalEnable(PT21);
   GPIO_PT2OutputLow(PT21);
}
/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

4. 數位 IP(TMA2)

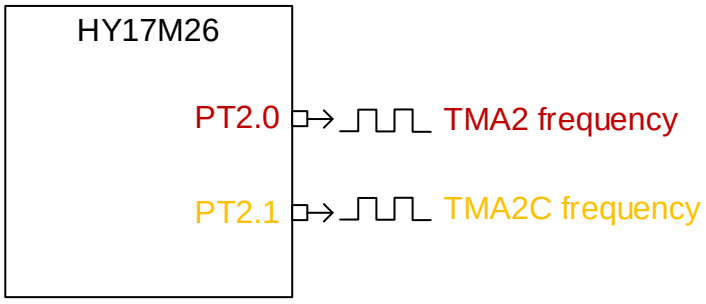
4.1. 範例名稱

TMA2

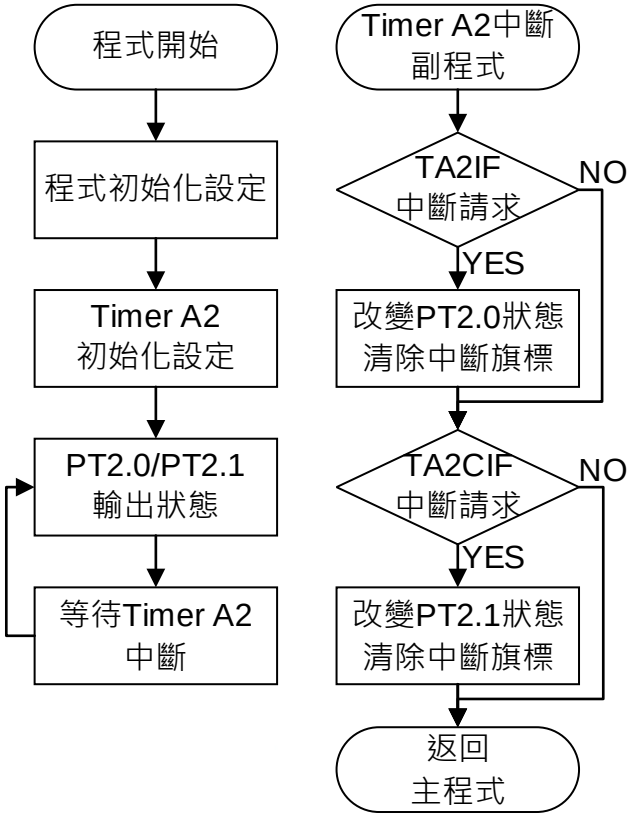
4.2. 範例說明

(6) Timer A2 使用方式與說明。
(7) 設置系統工作頻率。
(8) 程式做好 Timer A2 初始化動作與設置相關 Timer A2 計數溢出值，開啟 GIE 等待 Timer A2 中斷發生。Timer A2 計數溢出值可決定 Timer A2 進入中斷的時間。
(9) 每進一次 TA2 中斷，代表 TMA2R 加一，並改變 PT2.0 輸出狀態觀察變化速度。
(10) 每進一次 TA2C 中斷，代表 TMA2R 計數值等於 TMA2C，改變 PT2.1 輸出狀態觀察變化速度，可透過設置 TMA2C 暫存器增加 TMA2 整體計數時間。

4.3. 範例配置



4.4. 軟體流程


程式開始
程式初始化設定
Timer A2
初始化設定
PT2.0/PT2.1
輸出狀態
等待Timer A2
中斷
Timer A2中斷
副程式
TA2IF
中斷請求
NO
YES
改變PT2.0狀態
清除中斷旗標
TA2CIF
中斷請求
NO
YES
改變PT2.1狀態
清除中斷旗標
返回
主程式

4.5. 程式碼

```
#define   USE_HY17M26_2M
/*****************************************************************************
 * TMA2_Demo.c                                                               *
 * ------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                           *
 * http://www.hycontek.com/                                                  *
 *                                                                           *
 * Program Description:                                                      *
 *                                                                           *
 *            HY17M26                                                        *
 *--------------------                                                       *
 *                        |                                                  *
 *                    PT2.0|->_|-|_ TMA2                                     *
 *                        |                                                  *
 *                    PT2.1|->_|-|_ TMA2C                                    *
 *                        |                                                  *
 *--------------------                                                       *
 *                                                                           *
 * For External Input                                                        *
 * IC Body: HY17M26                                                          *
 * Project Name : TMA2                                                       *
 * Created Date : 2021/6/4 BY Cyril                                          *
 *                                                                           *
 *****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <INT.h>
#include <CLK.h>
#include <GPIO.h>
#include <RST.h>
#include <TMR.h>

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void TMAInit(void);
void CLOCKInit(void);
void GPIOInit(void);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned int TMA2R_Count;
unsigned int TMA2C_Count;
```

```
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
    CLOCKInit();
    GPIOInit();
    TMAInit();
    TMA2R_Count=0;
    TMA2C_Count=0;

    GIE_Enable();

    while(1);

}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{
     if(TA2IF_IsFlag())          //TA1IF=1,TMA1R++
    {
        PT2= PT2^PT20_MSK;    //PT2.0 toggle
        TMA2R_Count++;
        TA2IF_ClearFlag();
    }

     if(TA2CIF_IsFlag())           //TMA1C=TMA1R ,then TA1CIF=1
    {
        PT2= PT2^PT21_MSK;    //PT2.1 toggle
        TMA2C_Count++;
        TMA2_ClearTMA2R();    //clear TMA1R
        TA2CIF_ClearFlag();
     }
}

/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
  //HAO=1.843MHz
  CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
  //HAO=4.147MHz
  CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
  //HAO=8.755MHz
  CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
  //HAO=17.51MHz
  CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

  CLK_DMSCKSelect(DMS_DHSCKDIV2);    //DMS_CK= HAO/1/2
}
```

```
/*---------------------------------------------------------------------------*/
/* TMA Init Function                                                         */
/*---------------------------------------------------------------------------*/
void TMAInit(void)
{
   TMA2_Open(TMAS_DMSCK,DTMA_TMACKDIV2,100); //DTMA_CK=TMA_CK/256/2
                                             //TMAxC_Flag=TMAx_Flag*100
   TA2CIF_ClearFlag();
   TA2CIE_Enable();
   TA2IF_ClearFlag();
   TA2IE_Enable();
}

/*---------------------------------------------------------------------------*/
/* GPIO Init Function                                                        */
/*---------------------------------------------------------------------------*/
void GPIOInit(void)
{
   GPIO_PT2OutputMode(PT20);
   GPIO_PT2DigitalEnable(PT20);
   GPIO_PT2OutputLow(PT20);

   GPIO_PT2OutputMode(PT21);
   GPIO_PT2DigitalEnable(PT21);
   GPIO_PT2OutputLow(PT21);
}
/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

5. 數位 IP(TMB)

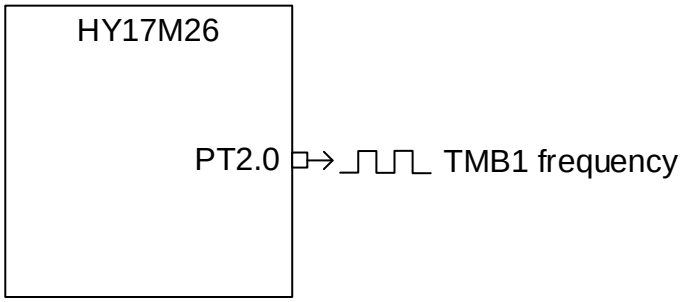
5.1. 範例名稱

TMB

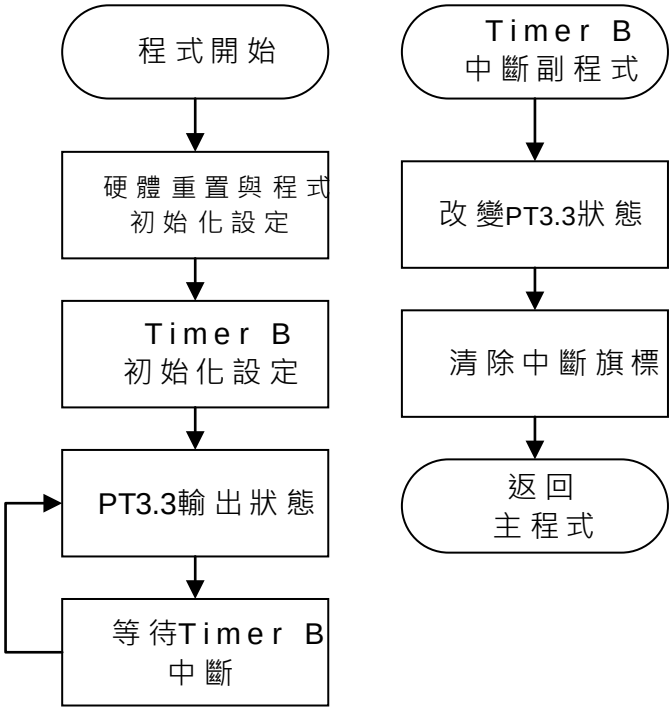
5.2. 範例說明

(1) Timer B 使用方式與說明。
(2) 利用程式碼#define 來選擇要編譯執行 mode_16bit, mode_17bit, mode_2_8bit 或 mode_8_8bit
(3) 設置系統工作頻率。
(4) 程式做好 Timer B 初始化動作與設置相關 Timer B 計數溢出值，開啟 GIE 等待 Timer B 中斷發生。TimerB 計數溢出值可決定 TimerB 進入中斷的時間。
(5) 每進一次 TMB 中斷，會在 TMB 中斷副程式內改變 PT2.0 輸出狀態，觀察轉換速度。

5.3. 範例配置



5.4. 軟體流程


程式開始
硬體重置與程式初始化設定
Timer B初始化設定
PT3.3輸出狀態
等待Timer B中斷
Timer B中斷副程式
改變PT3.3狀態
清除中斷旗標
返回主程式

5.5. 程式碼

```
#define   USE_HY17M26_2M
/*****************************************************************************
 * TMB_Demo.c                                                                *
 * ------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                           *
 * http://www.hycontek.com/                                                  *
 *                                                                           *
 * Program Description:                                                      *
 *                                                                           *
 *            HY17M26                                                        *
 *--------------------                                                       *
 *                          |                                                *
 *                    PT2.0|->_|-|_ TMB                                      *
 *                          |                                                *
 *--------------------                                                       *
 *                                                                           *
 * For External Input                                                        *
 * IC Body: HY17M26                                                          *
 * Project Name : TMB                                                        *
 * Created Date : 2021/6/4 BY Cyril                                          *
 *                                                                           *
 *****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <TMR.h>
#include <CLK.h>
#include <INT.h>
#include <GPIO.h>
#include <PWR.h>
#include <RST.h>
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

#define mode_16bit
//#define mode_17bit
//#define mode_2_8bit
//#define mode_8_8bit
```

```
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void CLOCKInit(void);
void TMBInit(void);
void GPIOInit(void);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
        CLOCKInit();
        TMBInit();
        GPIOInit();

        GIE_Enable();
        while(1);
}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{
        if(TB1IF_IsFlag())          //TB1R=TB1C ,TMBIF=1
        {
                PT2= PT2^PT20_MSK;    //PT2.0 Inverted
                TB1IF_ClearFlag();
        }
}

/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
   unsigned char t;
   for(;ms>0;ms--)              //@HAO=1.843MHz
      for(t=114;t>0;t--)
      { __asm__("NOP"); }
```

```
}

/*---------------------------------------------------------------------------*/
/* TMB Init Function                                                         */
/*---------------------------------------------------------------------------*/
void TMBInit(void)              //select TMB mode
{
#if defined(mode_16bit)
   TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV2 ,TB1M_16bit ,TB1RT_LogicH );
   TB1C0Set(0x00ff);     //Set TMB count
#endif
#if defined(mode_17bit)
   TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV2 ,TB1M_17bit ,TB1RT_LogicH );
   TB1C0Set(0x02ff);     //Set TMB count
#endif
#if defined(mode_2_8bit)
   TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV2 ,TB1M_2_8bit ,TB1RT_LogicH );
   TB1C0Set(0x7fff);     //Set TMB count
#endif
#if defined(mode_8_8bit)
   TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV2 ,TB1M_8_8bit ,TB1RT_LogicH );
   TB1C0Set(0x7fff);     //Set TMB count
#endif
   TB1Enable();
}

/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* GPIO Init Function                                                        */
/*---------------------------------------------------------------------------*/
void GPIOInit(void)
```

```
{
   GPIO_PT2OutputMode(PT20);
   GPIO_PT2DigitalEnable(PT20);
   GPIO_PT2OutputLow(PT20);

}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

6. 數位 IP(TMB2)

6.1. 範例名稱

TMB2

6.2. 範例說明

(6) Timer B2 使用方式與說明。

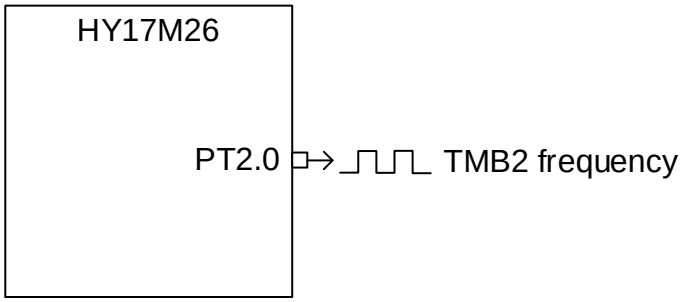
(7) 利用程式碼#define 來選擇要編譯執行 mode_16bit, mode_17bit, mode_2_8bit 或 mode_8_8bit

(8) 設置系統工作頻率。

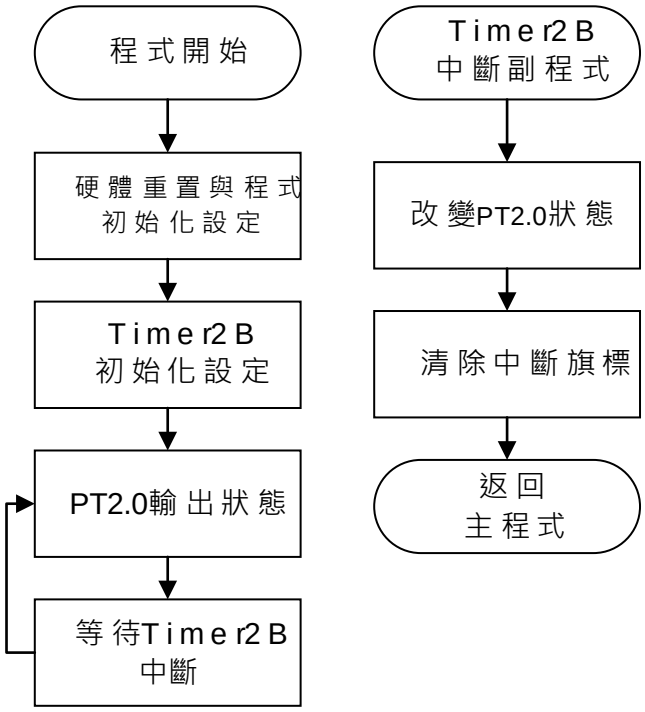
(9) 程式做好 Timer B2 初始化動作與設置相關 Timer B2 計數溢出值，開啟 GIE 等待 Timer B2 中斷發生。TimerB2 計數溢出值可決定 TimerB2 進入中斷的時間。

(10) 每進一次 TMB2 中斷，會在 TMB2 中斷副程式內改變 PT2.0 輸出狀態，觀察轉換速度。

6.3. 範例配置



6.4. 軟體流程


程式開始
硬體重置與程式初始化設定
Timer2B初始化設定
PT2.0輸出狀態
等待Timer2B中斷
Timer2B中斷副程式
改變PT2.0狀態
清除中斷旗標
返回主程式

6.5. 程式碼

```
#define   USE_HY17M26_2M
/*****************************************************************************
 * TMB2_Demo.c                                                               *
 * ------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                           *
 * http://www.hycontek.com/                                                  *
 *                                                                           *
 * Program Description:                                                      *
 *                                                                           *
 *            HY17M26                                                        *
 *--------------------                                                       *
 *                          |                                                *
 *                    PT2.0|->_|-|_ TMB2                                     *
 *                          |                                                *
 *--------------------                                                       *
 *                                                                           *
 * For External Input                                                        *
 * IC Body: HY17M26                                                          *
 * Project Name : TMB2                                                       *
 * Created Date : 2021/6/4 BY Cyril                                          *
 *                                                                           *
 *****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <TMR.h>
#include <CLK.h>
#include <INT.h>
#include <GPIO.h>
#include <RST.h>
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

#define mode_16bit
//#define mode_17bit
//#define mode_2_8bit
//#define mode_8_8bit
```

```
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void CLOCKInit(void);
void TMBInit(void);
void GPIOInit(void);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char tmb_flag=0;
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
        CLOCKInit();
        TMBInit();
        GPIOInit();

        GIE_Enable();
        while(1);
}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{
        if(TB2IF_IsFlag())          //TB1R=TB1C ,TMBIF=1
        {
                PT2= PT2^PT20_MSK;    //PT2.0 Inverted
                TB2IF_ClearFlag();
        }
}

/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
   unsigned char t;
   for(;ms>0;ms--)              //@HAO=1.843MHz
      for(t=114;t>0;t--)
      { __asm__("NOP"); }
}
```

```
/*---------------------------------------------------------------------------*/
/* TMB Init Function                                                         */
/*---------------------------------------------------------------------------*/
void TMBInit(void)                  //select TMB mode
{
#if defined(mode_16bit)
   TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV2 ,TB2M_16bit ,TB2RT_LogicH );
   TB2C0Set(0x00ff);     //Set TMB count
#endif
#if defined(mode_17bit)
   TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV2 ,TB2M_17bit ,TB2RT_LogicH );
   TB2C0Set(0x02ff);     //Set TMB count
#endif
#if defined(mode_2_8bit)
   TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV2 ,TB2M_2_8bit ,TB2RT_LogicH );
   TB2C0Set(0x7fff);      //Set TMB count
#endif
#if defined(mode_8_8bit)
   TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV2 ,TB2M_8_8bit ,TB2RT_LogicH );
   TB2C0Set(0x7fff);      //Set TMB count
#endif
   TB2Enable();
}

/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* GPIO Init Function                                                        */
/*---------------------------------------------------------------------------*/
void GPIOInit(void)
{
```

```
    GPIO_PT2OutputMode(PT20);
    GPIO_PT2DigitalEnable(PT20);
    GPIO_PT2OutputLow(PT20);

}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

7. 數位 IP(PWM)

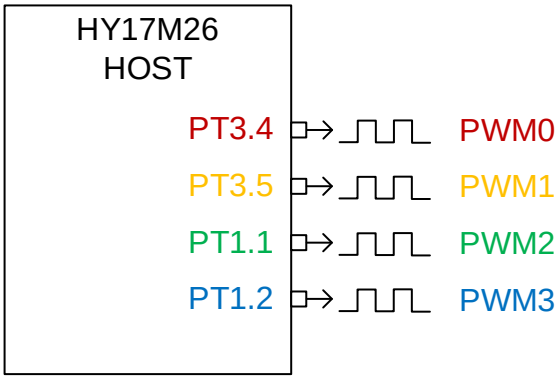
7.1. 範例名稱

PWM

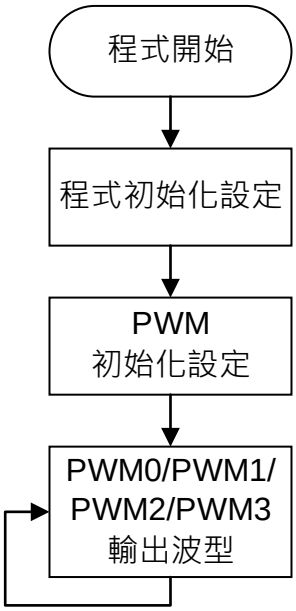
7.2. 範例說明

(1) PWM 使用方式與說明。
(2) 利用程式碼#define 來選擇 PWM 模式要編譯執行 PWM1O, PWM2O, PWM3O, PWM4O, PWM5O, PWM6O 或 PWM7O。
(3) 利用程式碼#define 來選擇 PWM0,PWM1,PWM2,PWM3 輸出腳位。
(4) 設置系統工作頻率。
(5) 開啟 PWM 暫存器設定與 PWM Duty 設定。
(6) 可透過上述規劃輸出腳位觀察 PWM 輸出頻率與 Duty。

7.3. 範例配置



7.4. 軟體流程



7.5. 程式碼

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * PWM_Demo.c                                                                  *
 * ---------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                             *
 * http://www.hycontek.com/                                                    *
 *                                                                             *
 * Program Description:                                                        *
 *                                                                             *
 *    HY17M26                                                                  *
 * -----------                                                                 *
 *              |                                                              *
 *       PT3.4 |-->PWM0                                                        *
 *              |                                                              *
 *       PT3.5 |-->PWM1                                                        *
 *              |                                                              *
 *       PT1.1 |-->PWM2                                                        *
 *              |                                                              *
 *       PT1.2 |-->PWM3                                                        *
 *              |                                                              *
 * -----------                                                                 *
 *                                                                             *
 * For External Input                                                          *
 * IC Body: HY17M26                                                            *
 * Project Name : PWM                                                          *
 * Created Date : 2021/6/4 By Cyril                                            *
 *                                                                             *
 *******************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <TMR.h>
#include <CLK.h>
#include <INT.h>
#include <RST.h>
#include <GPIO.h>
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif
```

```
////     Select PWM MODE              //
//#define PWM1O
//#define PWM2O
//#define PWM3O
//#define PWM4O
//#define PWM5O
//#define PWM6O
#define PWM7O

//TMB_PWM PORT
//PWM0
//#define PWM0_PT30
//#define PWM0_PT32
#define PWM0_PT34
//#define PWM0_PT36

//PWM1
//#define PWM1_PT31
//#define PWM1_PT33
#define PWM1_PT35
//#define PWM1_PT37

//PWM2
#define PWM2_PT11
//#define PWM2_PT13
//#define PWM2_PT20
//#define PWM2_PT22

//PWM3
#define PWM3_PT12
//#define PWM3_PT14
//#define PWM3_PT21
//#define PWM3_PT23
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void CLOCKInit(void);
void PWMInit(void);

/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
    CLOCKInit();
    PWMInit();


    while(1);
```

```
}
/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* PWM Init Function                                                         */
/*---------------------------------------------------------------------------*/
void PWMInit(void)
{
#if defined(PWM0_PT30)
   GPIO_GPWM0Set(GPWM0_PT30);
#endif
#if defined(PWM0_PT32)
   GPIO_GPWM0Set(GPWM0_PT32);
#endif
#if defined(PWM0_PT34)
   GPIO_GPWM0Set(GPWM0_PT34);
#endif
#if defined(PWM0_PT36)
   GPIO_GPWM0Set(GPWM0_PT36);
#endif

#if defined(PWM1_PT31)
   GPIO_GPWM1Set(GPWM1_PT31);
#endif
#if defined(PWM1_PT33)
   GPIO_GPWM1Set(GPWM1_PT33);
#endif
#if defined(PWM1_PT35)
```

```
   GPIO_GPWM1Set(GPWM1_PT35);
#endif
#if defined(PWM1_PT37)
   GPIO_GPWM1Set(GPWM1_PT37);
#endif

#if defined(PWM2_PT11)
   GPIO_GPWM2Set(GPWM2_PT11);
#endif
#if defined(PWM2_PT13)
   GPIO_GPWM2Set(GPWM2_PT13);
#endif
#if defined(PWM2_PT20)
   GPIO_GPWM2Set(GPWM2_PT20);
#endif
#if defined(PWM2_PT22)
   GPIO_GPWM2Set(GPWM2_PT22);
#endif

#if defined(PWM3_PT12)
   GPIO_GPWM3Set(GPWM3_PT12);
#endif
#if defined(PWM3_PT14)
   GPIO_GPWM3Set(GPWM3_PT14);
#endif
#if defined(PWM3_PT21)
   GPIO_GPWM3Set(GPWM3_PT21);
#endif
#if defined(PWM3_PT23)
   GPIO_GPWM3Set(GPWM3_PT23);
#endif

//***** setting the timer B & PWM *********//
/*-------------------------------------PWM1O-------------------------------------------*/
#ifdef PWM1O
      TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB1M_16bit ,TB1RT_LogicH );
      TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB2M_16bit ,TB2RT_LogicH );

      TB1_PWM0_PHASE(PA0IV_NORMAL);          //PWM0 Output Phase Normal
      TB1_PWM1_PHASE(PA1IV_INVER);           //PWM1 Output Phase Normal
      TB2_PWM2_PHASE(PA2IV_NORMAL);          //PWM2 Output Phase Normal
      TB2_PWM3_PHASE(PA3IV_INVER);           //PWM3 Output Phase Normal

      TB1_PWM0ModeSelect(PWMA0_PWM1O);          //PWM0 Output mode Select PWM1O
      TB1_PWM1ModeSelect(PWMA1_PWM1O);          //PWM1 Output mode Select PWM1O
      TB2_PWM2ModeSelect(PWMA2_PWM1O);          //PWM2 Output mode Select PWM1O
      TB2_PWM3ModeSelect(PWMA3_PWM1O);          //PWM3 Output mode Select PWM1O

      TB1_PWMO0(PWMO0_OUTPUT);                  //PWM0 Enable
      TB1_PWMO1(PWMO1_OUTPUT);                  //PWM1 Enable
      TB2_PWMO2(PWMO2_OUTPUT);                  //PWM2 Enable
      TB2_PWMO3(PWMO3_OUTPUT);                  //PWM3 Enable

      TB1C0Set(0x0800);     // TB1C0[15:0] is setting the PWM frequency.
      TB1C1Set(0x0400);     // TB1C1[15:0] is setting the PWM Duty Cycle.
      TB2C0Set(0x0800);     // TB2C0[15:0] is setting the PWM frequency.
      TB2C1Set(0x0400);     // TB2C1[15:0] is setting the PWM Duty Cycle.
```

```
#endif

/*-------------------------------------PWM2O-------------------------------------------*/
#ifdef PWM2O
    TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB1M_16bit ,TB1RT_LogicH );
    TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB2M_16bit ,TB2RT_LogicH );

    TB1_PWM0_PHASE(PA0IV_NORMAL);       //PWM0 Output Phase Normal
    TB1_PWM1_PHASE(PA1IV_INVER);        //PWM1 Output Phase Normal
    TB2_PWM2_PHASE(PA2IV_NORMAL);       //PWM2 Output Phase Normal
    TB2_PWM3_PHASE(PA3IV_INVER);        //PWM3 Output Phase Normal

    TB1_PWM0ModeSelect(PWMA0_PWM2O);        //PWM0 Output mode Select PWM2O
    TB1_PWM1ModeSelect(PWMA1_PWM2O);        //PWM1 Output mode Select PWM2O
    TB2_PWM2ModeSelect(PWMA2_PWM2O);        //PWM2 Output mode Select PWM2O
    TB2_PWM3ModeSelect(PWMA3_PWM2O);        //PWM3 Output mode Select PWM2O

    TB1_PWMO0(PWMO0_OUTPUT);                //PWM0 Enable
    TB1_PWMO1(PWMO1_OUTPUT);                //PWM1 Enable
    TB2_PWMO2(PWMO2_OUTPUT);                //PWM2 Enable
    TB2_PWMO3(PWMO3_OUTPUT);                //PWM3 Enable

    TB1C0Set(0x0833);   // TB1C0[15:0] is setting the PWM frequency.
    TB1C2Set(0x01a4);   // TB1C2[15:0] is setting the PWM Duty Cycle.
    TB2C0Set(0x0833);   // TB2C0[15:0] is setting the PWM frequency.
    TB2C2Set(0x01a4);   // TB2C2[15:0] is setting the PWM Duty Cycle.
#endif

/*-------------------------------------PWM3O-------------------------------------------*/
#ifdef PWM3O
    TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB1M_2_8bit ,TB1RT_LogicH );
    TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB2M_2_8bit ,TB2RT_LogicH );

    TB1_PWM0_PHASE(PA0IV_NORMAL);       //PWM0 Output Phase Normal
    TB1_PWM1_PHASE(PA1IV_INVER);        //PWM1 Output Phase Normal
    TB2_PWM2_PHASE(PA2IV_NORMAL);       //PWM2 Output Phase Normal
    TB2_PWM3_PHASE(PA3IV_INVER);        //PWM3 Output Phase Normal

    TB1_PWM0ModeSelect(PWMA0_PWM3O);        //PWM0 Output mode Select PWM3O
    TB1_PWM1ModeSelect(PWMA1_PWM3O);        //PWM1 Output mode Select PWM3O
    TB2_PWM2ModeSelect(PWMA2_PWM3O);        //PWM2 Output mode Select PWM3O
    TB2_PWM3ModeSelect(PWMA3_PWM3O);        //PWM3 Output mode Select PWM3O

    TB1_PWMO0(PWMO0_OUTPUT);                //PWM0 Enable
    TB1_PWMO1(PWMO1_OUTPUT);                //PWM1 Enable
    TB2_PWMO2(PWMO2_OUTPUT);                //PWM2 Enable
    TB2_PWMO3(PWMO3_OUTPUT);                //PWM3 Enable

    TB1C0Set(0x0029);   // TB1C0L[7:0] is setting the PWM frequency.
    TB1C1Set(0x0016);   // TB1C0L[7:0] is setting the PWM Duty Cycle.
    TB2C0Set(0x0029);   // TB2C0L[7:0] is setting the PWM frequency.
    TB2C1Set(0x0016);   // TB2C0L[7:0] is setting the PWM Duty Cycle.
#endif

/*-------------------------------------PWM4O-------------------------------------------*/
#ifdef PWM4O
    TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB1M_2_8bit ,TB1RT_LogicH );
```

```
        TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB2M_2_8bit ,TB2RT_LogicH );

        TB1_PWM0_PHASE(PA0IV_NORMAL);          //PWM0 Output Phase Normal
        TB1_PWM1_PHASE(PA1IV_INVER);           //PWM1 Output Phase Normal
        TB2_PWM2_PHASE(PA2IV_NORMAL);          //PWM2 Output Phase Normal
        TB2_PWM3_PHASE(PA3IV_INVER);           //PWM3 Output Phase Normal

        TB1_PWM0ModeSelect(PWMA0_PWM4O);            //PWM0 Output mode Select PWM4O
        TB1_PWM1ModeSelect(PWMA1_PWM4O);            //PWM1 Output mode Select PWM4O
        TB2_PWM2ModeSelect(PWMA2_PWM4O);            //PWM2 Output mode Select PWM4O
        TB2_PWM3ModeSelect(PWMA3_PWM4O);            //PWM3 Output mode Select PWM4O

        TB1_PWMO0(PWMO0_OUTPUT);                    //PWM0 Enable
        TB1_PWMO1(PWMO1_OUTPUT);                    //PWM1 Enable
        TB2_PWMO2(PWMO2_OUTPUT);                    //PWM2 Enable
        TB2_PWMO3(PWMO3_OUTPUT);                    //PWM3 Enable

        TB1C0Set(0xD100);     // TB1C0H[15:8] is setting the PWM frequency.
        TB1C2Set(0x0088);     // TB1C2L[7:0] is setting the PWM Duty Cycle.
        TB2C0Set(0xD100);     // TB2C0H[15:8] is setting the PWM frequency.
        TB2C2Set(0x0088);     // TB2C2L[7:0] is setting the PWM Duty Cycle.
#endif

/*--------------------------------------PWM5O-------------------------------------------*/
#ifdef PWM5O
        TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB1M_8_8bit ,TB1RT_LogicH );
        TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB2M_8_8bit ,TB2RT_LogicH );

        TB1_PWM0_PHASE(PA0IV_NORMAL);          //PWM0 Output Phase Normal
        TB1_PWM1_PHASE(PA1IV_INVER);           //PWM1 Output Phase Normal
        TB2_PWM2_PHASE(PA2IV_NORMAL);          //PWM2 Output Phase Normal
        TB2_PWM3_PHASE(PA3IV_INVER);           //PWM3 Output Phase Normal

        TB1_PWM0ModeSelect(PWMA0_PWM5O);            //PWM0 Output mode Select PWM5O
        TB1_PWM1ModeSelect(PWMA1_PWM5O);            //PWM1 Output mode Select PWM5O
        TB2_PWM2ModeSelect(PWMA2_PWM5O);            //PWM2 Output mode Select PWM5O
        TB2_PWM3ModeSelect(PWMA3_PWM5O);            //PWM3 Output mode Select PWM5O

        TB1_PWMO0(PWMO0_OUTPUT);                    //PWM0 Enable
        TB1_PWMO1(PWMO1_OUTPUT);                    //PWM1 Enable
        TB2_PWMO2(PWMO2_OUTPUT);                    //PWM2 Enable
        TB2_PWMO3(PWMO3_OUTPUT);                    //PWM3 Enable

        TB1C0Set(0x00C8);     // TB1C0L[7:0] is setting the PWM frequency.
        TB1C1Set(0x0064);     // TB1C1L[7:0] is setting the PWM Duty Cycle.
        TB1C2Set(0x0080);     // TB1C2L[7:0] is setting the PWM Duty Cycle fine tuning.
        TB2C0Set(0x00C8);     // TB2C0L[7:0] is setting the PWM frequency.
        TB2C1Set(0x0064);     // TB2C1L[7:0] is setting the PWM Duty Cycle.
        TB2C2Set(0x0080);     // TB2C2L[7:0] is setting the PWM Duty Cycle fine tuning.
#endif

/*--------------------------------------PWM6O-------------------------------------------*/
#ifdef PWM6O
        TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB1M_17bit ,TB1RT_LogicH );
        TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB2M_17bit ,TB2RT_LogicH );

        TB1_PWM0_PHASE(PA0IV_NORMAL);          //PWM0 Output Phase Normal
```

```
        TB1_PWM1_PHASE(PA1IV_INVER);            //PWM1 Output Phase Normal
        TB2_PWM2_PHASE(PA2IV_NORMAL);           //PWM2 Output Phase Normal
        TB2_PWM3_PHASE(PA3IV_INVER);            //PWM3 Output Phase Normal

        TB1_PWM0ModeSelect(PWMA0_PWM6O);            //PWM0 Output mode Select PWM6O
        TB1_PWM1ModeSelect(PWMA1_PWM6O);            //PWM1 Output mode Select PWM6O
        TB2_PWM2ModeSelect(PWMA2_PWM6O);            //PWM2 Output mode Select PWM6O
        TB2_PWM3ModeSelect(PWMA3_PWM6O);            //PWM3 Output mode Select PWM6O

        TB1_PWMO0(PWMO0_OUTPUT);                    //PWM0 Enable
        TB1_PWMO1(PWMO1_OUTPUT);                    //PWM1 Enable
        TB2_PWMO2(PWMO2_OUTPUT);                    //PWM2 Enable
        TB2_PWMO3(PWMO3_OUTPUT);                    //PWM3 Enable

        TB1C0Set(0x0002);     // TB1C0L[7:0] is setting the PWM frequency.
        TB1C1Set(0x0001);     // TB1C1L[7:0] is setting the PWM Duty Cycle.
        TB1C2Set(0x0080);     // TB1C2L[7:0] is setting the PWM Duty Cycle
        TB2C0Set(0x0002);     // TB2C0L[7:0] is setting the PWM frequency.
        TB2C1Set(0x0001);     // TB2C1L[7:0] is setting the PWM Duty Cycle.
        TB2C2Set(0x0080);     // TB2C2L[7:0] is setting the PWM Duty Cycle
                                                    // TB1C0 > TB1C2 > TB1C1
#endif

/*-------------------------------------PWM7O-------------------------------------------*/
#ifdef PWM7O
        TMB1_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB1M_16bit ,TB1RT_LogicH );
        TMB2_Open(TMBS_HSCK ,DTMB_TMBCKDIV1 ,TB2M_16bit ,TB2RT_LogicH );

        TB1_PWM0_PHASE(PA0IV_NORMAL);           //PWM0 Output Phase Normal
        TB1_PWM1_PHASE(PA1IV_INVER);            //PWM1 Output Phase Normal
        TB2_PWM2_PHASE(PA2IV_NORMAL);           //PWM2 Output Phase Normal
        TB2_PWM3_PHASE(PA3IV_INVER);            //PWM3 Output Phase Normal

        TB1_PWM0ModeSelect(PWMA0_PWM7O);            //PWM0 Output mode Select PWM7O
        TB1_PWM1ModeSelect(PWMA1_PWM7O);            //PWM1 Output mode Select PWM7O
        TB2_PWM2ModeSelect(PWMA2_PWM7O);            //PWM2 Output mode Select PWM7O
        TB2_PWM3ModeSelect(PWMA3_PWM7O);            //PWM3 Output mode Select PWM7O

        TB1_PWMO0(PWMO0_OUTPUT);                    //PWM0 Enable
        TB1_PWMO1(PWMO1_OUTPUT);                    //PWM1 Enable
        TB2_PWMO2(PWMO2_OUTPUT);                    //PWM2 Enable
        TB2_PWMO3(PWMO3_OUTPUT);                    //PWM3 Enable

        TB1C0Set(0x3FFF);     // TB1C0[15:0] is setting the PWM frequency.
                                                    // PWM Duty Cycle is always 50%
        TB2C0Set(0x3FFF);     // TB2C0[15:0] is setting the PWM frequency.
                                                    // PWM Duty Cycle is always 50%
#endif

        TB1Enable();
        TB2Enable();
}

/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
```

```
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
   unsigned char t;
   for(;ms>0;ms--)              //@HAO=1.843MHz
      for(t=114;t>0;t--)
      { __asm__("NOP"); }
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

8. 數位 IP(WDT)

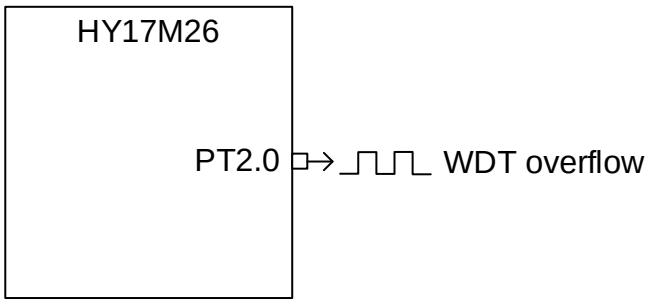
8.1. 範例名稱

WDT

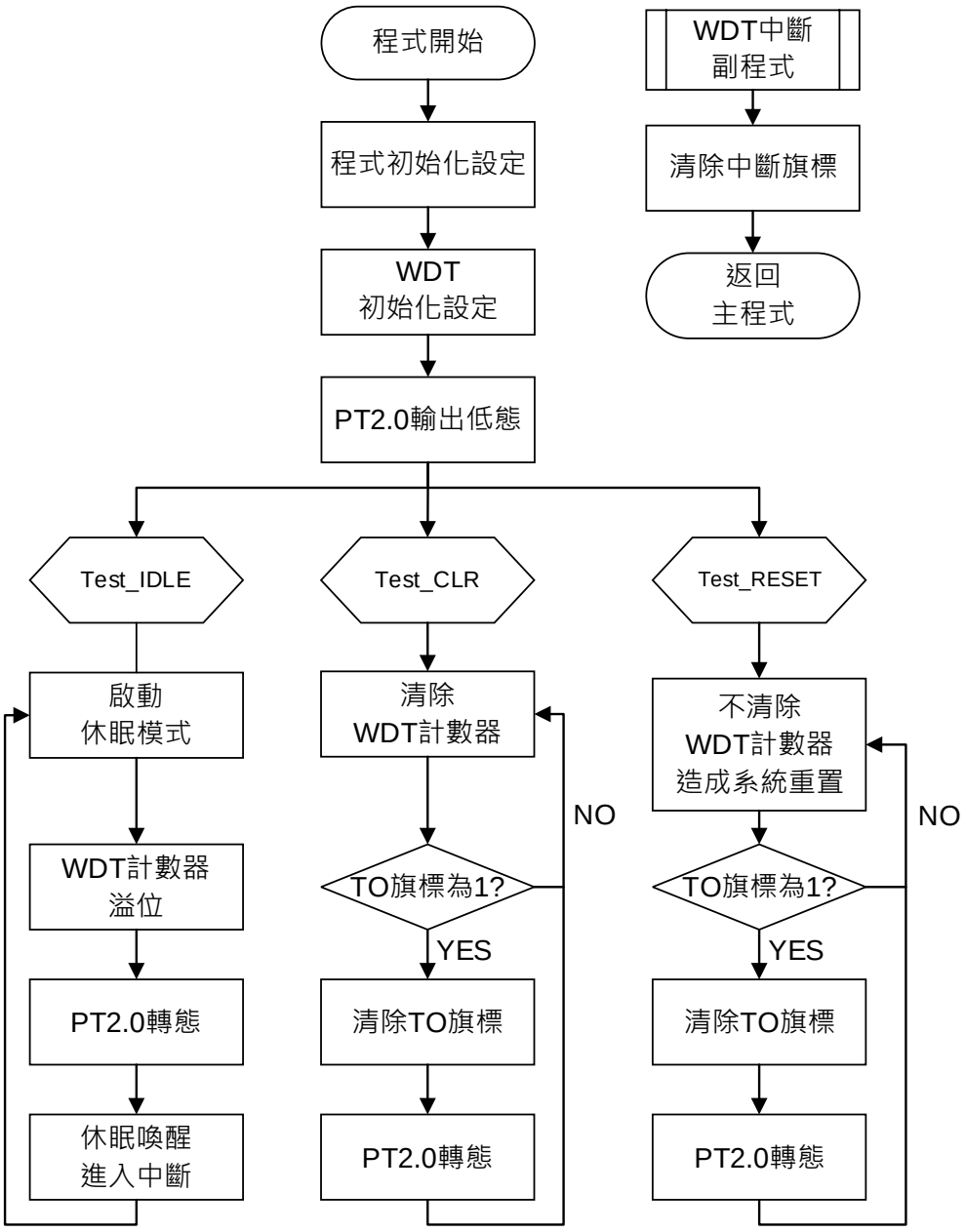
8.2. 範例說明

(1) WDT 使用方式與說明
(2) 利用程式碼#define 來選擇要編譯執行 Test_IDLE、 Test_CLR 或 Test_RESET。
(3) 設置系統工作頻率。
(4) 程式做好 WDT 初始化動作與設置 WDT 計數溢出值條件，開啟 GIE 等待 WDT 中斷發生。WDT 計數溢出值可決定 WDT 進出中斷的速度。
(5) 每進一次 WDT 中斷，會在 WDT 中斷副程式內改變 PT2.0 輸出高態，並於離開中斷前清除 WDT Count。
(6) 在 Test_IDLE 下，當 WDT 計數值溢出時，系統將從 Idle 模式恢復至一般模式並進入中斷執行對應動作，於離開中斷前清除 WDT 計數值，避免系統重置。
(7) 在 Test_CLR 下，當 WDT 計數值溢出時，系統將 Reset 並設置 TO 旗標為 1。
(8) 在 Test_RESET 下，在 WDT 計數值溢出前清除 WDT Count，避免系統重置。

8.3. 範例配置



8.4. 軟體流程


程式開始
程式初始化設定
WDT
初始化設定
PT2.0輸出低態
Test_IDLE
Test_CLR
Test_RESET
啟動
休眠模式
WDT計數器
溢位
PT2.0轉態
休眠喚醒
進入中斷
清除
WDT計數器
TO旗標為1?
NO
YES
清除TO旗標
PT2.0轉態
不清除
WDT計數器
造成系統重置
TO旗標為1?
NO
YES
清除TO旗標
PT2.0轉態
WDT中斷
副程式
清除中斷旗標
返回
主程式

8.5. 程式碼

```
#define   USE_HY17M26_2M
// /*****************************************************************************
  // * WDT_Demo.c                                                                *
  // * ------------------------------------------------------------------------- *
  // * Copyright 2021 HYCON Technology                                           *
  // * http://www.hycontek.com/                                                  *
  // *                                                                           *
  // * Program Description:                                                      *
  // *                                                                           *
  // * Test_IDLE: IDLE mode. WDT count overflow, wake up from IDLE, WDTIF=1      *
  // * Test_CLR : Normal mode. Clear WDT count, IC doesn't reset                 *
  // * Test_NOP: Normal mode. Not clear WDT count,IC will reset, TO=1            *
  // *                                                                           *
  // * For External Input                                                        *
  // * IC Body: HY17M26                                                          *
  // * Project Name : WDT                                                        *
  // * Created Date : 2021/6/4 BY Cyril                                          *
  // *                                                                           *
  // *****************************************************************************/

// /*---------------------------------------------------------------------------*/
// /* Includes                                                                  */
// /*---------------------------------------------------------------------------*/
// #include <SFRType.h>
// #include <WDT.h>
// #include <INT.h>
// #include <RST.h>
// #include <CLK.h>
// #include <GPIO.h>
// /*---------------------------------------------------------------------------*/
// /* DEFINITIONS                                                               */
// /*---------------------------------------------------------------------------*/
// #ifdef USE_HY17M26_2M
// #define HAO_2MHZ
// #endif
// #ifdef USE_HY17M26_4M
// #define HAO_4MHZ
// #endif
// #ifdef USE_HY17M26_8M
// #define HAO_8MHZ
// #endif
// #ifdef USE_HY17M26_16M
// #define HAO_17MHZ
// #endif

#define Test_IDLE
#define Test_CLR
// #define Test_RESET

// /*---------------------------------------------------------------------------*/
// /* Global CONSTANTS                                                          */
// /*---------------------------------------------------------------------------*/
```

```
// /*---------------------------------------------------------------------------*/
// /* Function PROTOTYPES                                                       */
// /*---------------------------------------------------------------------------*/
// void WDTInit(void);
// void CLOCKInit(void);
// void GPIOInit(void);
// /*---------------------------------------------------------------------------*/
// /* Main Function                                                             */
// /*---------------------------------------------------------------------------*/
// void main(void)
// {
    // CLOCKInit();
    // GPIOInit();
    // WDTInit();
    // GIE_Enable();


// #ifdef Test_IDLE       //wake up Idle
    // while(1)
    // {
      // WDT_WDTCKDivSelect(DWDT_WDTCKDIV64);
        // Idle();
        // NOP();
      // PT2= PT2^PT20_MSK;    //PT2.0 toggle
     // }
// #endif

// #ifdef Test_CLR     //clear WDT Count, won't reset
    // while(1)
    // {
      // if(SYS_ReadWDT()!=0)    //if program reset, TO=1
      // {
            // SYS_ClearWDT();
            // PT2= PT2^PT20_MSK;    //PT2.0 toggle
      // }
      // WDT_Clear();
    // }
// #endif

// #ifdef Test_RESET     //nop, will reset
    // while(1)
    // {
      // if(SYS_ReadWDT()!=0)    //if program reset, TO=1
      // {
            // SYS_ClearWDT();
            // PT2= PT2^PT20_MSK;    //PT2.0 toggle
      // }
      // NOP();
    // }
// #endif

// }

// /*---------------------------------------------------------------------------*/
// /* Interrupt Service Routines                                                */
// /*---------------------------------------------------------------------------*/
// void ISR(void) __interrupt    //WDT Reset
```

```
// {
        // WDTIF_ClearFlag();
        // SYS_ClearIDLE();      //CLR IDLE Flag
        // WDT_Clear();          //CLR WDT count.
// }

// /*---------------------------------------------------------------------------*/
// /* Subroutine Function                                                       */
// /*---------------------------------------------------------------------------*/
// /*---------------------------------------------------------------------------*/
// /* CLOCK Init Function                                                       */
// /*---------------------------------------------------------------------------*/
// void CLOCKInit(void)
// {
// #if defined(HAO_2MHZ)
   HAO=1.843MHz
   // CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
// #endif

// #if defined(HAO_4MHZ)
   HAO=4.147MHz
   // CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
// #endif

// #if defined(HAO_8MHZ)
   HAO=8.755MHz
   // CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
// #endif

// #if defined(HAO_17MHZ)
   HAO=17.51MHz
   // CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
// #endif

// }

// /*---------------------------------------------------------------------------*/
// /* WDT Init Function                                                         */
// /*---------------------------------------------------------------------------*/
// void WDTInit(void)
// {
   // WDT_Open(DWDT_WDTCKDIV2048);         //WDT Open
   // WDT_Clear();                         // CLEAR WDT COUNT
   // WDTIF_ClearFlag();
   // WDTIE_Enable();
   // WDT_Enable();
// }

// /*---------------------------------------------------------------------------*/
// /* GPIO Init Function                                                        */
// /*---------------------------------------------------------------------------*/
// void GPIOInit(void)
// {
   // GPIO_PT2OutputMode(PT20);
   // GPIO_PT2DigitalEnable(PT20);
   // GPIO_PT2OutputLow(PT20);
```

```
// }
// /*---------------------------------------------------------------------------*/
// /* End Of File                                                               */
// /*---------------------------------------------------------------------------*/
```

9. 數位 IP(BIE)

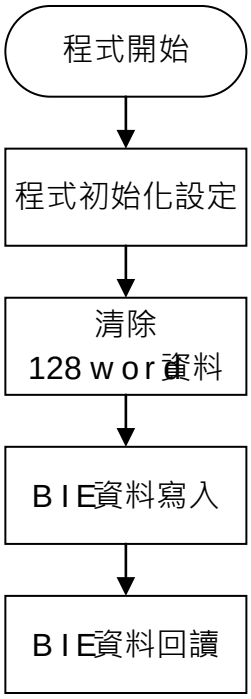
9.1. 範例名稱

BIE

9.2. 範例說明

(1) BIE 自我燒錄與讀取使用說明
(2) 設置系統工作頻率。
(3) 開啟 BIE 功能，並清除 128 word 資料。
(4) 回讀確認資料清除
(5) 設置燒錄位址及燒錄資料，執行 BIE 燒錄功能。
(6) 設置讀取位址，執行 BIE 讀取功能。

9.3. 軟體流程



9.4. 程式碼

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * BIE_Demo.c                                                                  *
 * ---------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                               *
 * http://www.hycontek.com/                                                    *
 *                                                                               *
 * Program Description:                                                        *
 *                                                                               *
 *     BIE Address                                                             *
 * addr[0x00]=0x0123                                                           *
 * addr[0x01]=0x4567                                                           *
 * addr[0x02]=0x89AB                                                           *
 * addr[0x03]=0xCDEF                                                           *
 *                                                                               *
 * For External Input                                                          *
 * IC Body: HY17M26                                                              *
 * Project Name : BIE                                                          *
 * Created Date : 2021/6/4 BY Cyril                                            *
 *                                                                               *
 *******************************************************************************/
/*-----------------------------------------------------------------------------*/
/* Includes                                                                    */
/*-----------------------------------------------------------------------------*/
#include <SFRType.h>
#include <BIE.h>
#include <CLK.h>
#include <RST.h>


/*-----------------------------------------------------------------------------*/
/* DEFINITIONS                                                                 */
/*-----------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif
/*-----------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                         */
/*-----------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
/*-----------------------------------------------------------------------------*/
/* Global CONSTANTS                                                            */
/*-----------------------------------------------------------------------------*/
```

```
unsigned int BIE_ReadData[10];
unsigned char Addr;
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{    unsigned char i;

     BIE1_Enable();
     BIE1_Erase();          //Erase BIE(addr 0x00~0x7F), total 128 Word
     for(i=0;i<=9;i++)
     {
          BIE_ReadData[i]=BIE1_ReadData(i);    // Read Data
     }

     //BIE Write
     Addr=0x00;
     BIE1_WriterData(Addr,0x01,0x23);    //Addr=0x00, DataH=0x01, DataL=0x23
     BIE1_WriterData(Addr+1,0x45,0x67);    //Addr=0x01, DataH=0x45, DataL=0x67
     BIE1_WriterData(Addr+2,0x89,0xAB);    //Addr=0x02, DataH=0x89, DataL=0xAB
     BIE1_WriterData(Addr+3,0xCD,0xEF);    //Addr=0x03, DataH=0xCD, DataL=0xEF

     //BIE Read
     for(i=0;i<=9;i++)
     {
          BIE_ReadData[i]=BIE1_ReadData(i);    // Read Data
     }

     while(1);
}

/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
```

```
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* Subroutines Function                                                      */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
   unsigned char t;
   for(;ms>0;ms--)              //@HAO=1.843MHz
      for(t=114;t>0;t--)
      { __asm__("NOP"); }
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

10. 類比 IP(ADC)

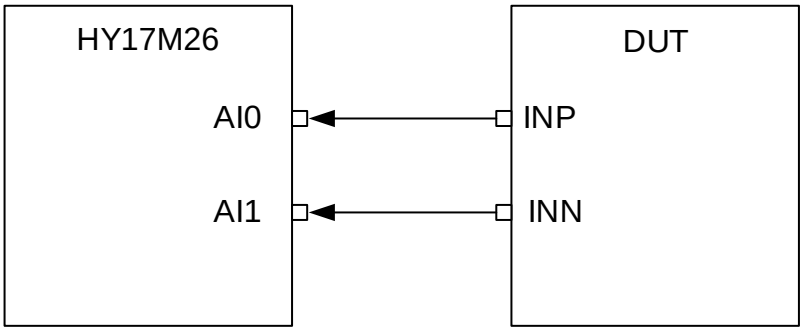
10.1. 範例名稱

ADC

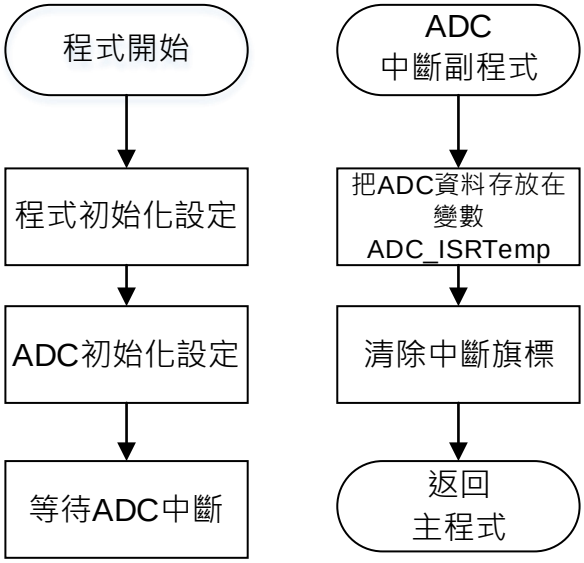
10.2. 範例說明

(1) 設置系統工作頻率。
(2) 透過 ADC 相關設定，可以實現 ADC 進入中斷抓取 ADC 輸出暫存器資料。
(3) ADC 初始設置包含，設置 ADC 的類比電源為 VDDA，設置 ADC 的取樣頻率 OSR 為 65536，設置 ADC 的輸入端設置為 AIO0-AIO1，設置 ADC 的參考電壓設置為 VDDA 對 VSS。
(4) ADC 初始設置完成後，開啟 GIE 等待 ADC 中斷發生。ADC OSR 設置可決定 ADC 進出中斷的速度。

10.3. 範例配置



10.4. 軟體流程



10.5. 程式碼

```
#define   USE_HY17M26_8M
/*******************************************************************************
 * ADC_Demo.c                                                                  *
 * --------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                             *
 * http://www.hycontek.com/                                                    *
 *                                                                             *
 * Program Description:                                                        *
 *                                                                             *
 *       HY17M26                                                               *
 *--------------------                                                         *
 *                        |                                                    *
 *                  AI0   |->INP                                               *
 *                  AI1   |->INN                                               *
 *                        |                                                    *
 *                        |                                                    *
 *--------------------                                                         *
 *                                                                             *
 *                                                                             *
 * For External Input                                                          *
 * IC Body: HY17M26                                                            *
 * Project Name : ADC                                                          *
 * Created Date : 2021/6/4 BY Cyril                                            *
 *                                                                             *
 *******************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
```

```
#include <SFRType.h>
#include <INT.h>
#include <CLK.h>
#include <PWR.h>
#include <ADC.h>
#include <RST.h>

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

//#define ADC_ChopperMode
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void SysInit(void);
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void ADCInit(void);
void CLOCKInit(void);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
volatile typedef union _Flag
{
  unsigned int    _byte;
    struct
    {
       unsigned b_ADCdone:1;
       unsigned Reserved:15;
    };
} Flag;

//signed long
//range 8388607~-8388608 (dec)
//range 0x007F FFFF ~ 0xFF80 0000 (hex)
signed long    ADC_ISRTemp;
signed long    ADC_ISRTemp_Buffer[8];
unsigned int counterA;
Flag     Flagbits;
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
```

```
    counterA=0;
    Flagbits.b_ADCdone=0;
    CLOCKInit();
    ADCInit();
    GIE_Enable();

    while(1)
    {
      if(Flagbits.b_ADCdone==1)
      {
        ADC_ISRTemp_Buffer[counterA]=ADC_ISRTemp;
        Flagbits.b_ADCdone=0;
        counterA++ ;
        if( counterA==8 )
        {
          counterA=0;
        }
      }
    }
}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{
    NOP();

    //ADC Event
    if(ADC_INT_IsFlag())
    {
      ADC_INT_ClearFlag();
      ADC_ISRTemp=ADCR; //ADC_GetData();
      Flagbits.b_ADCdone=1;
    }

}

/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
      for(;num>0;num--)
    __asm__("NOP");
}


void Delayms(unsigned int ms)
{
    unsigned char t;
    for(;ms>0;ms--)             //@HAO=1.843MHz
      for(t=114;t>0;t--)
```

```
    { __asm__("NOP"); }
}

/*---------------------------------------------------------------------------*/
/* Clock Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
  //HAO=1.843MHz
  CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
  //HAO=4.147MHz
  CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
  //HAO=8.755MHz
  CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
  //HAO=17.51MHz
  CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif
}

/*---------------------------------------------------------------------------*/
/* ADC Init Function                                                         */
/*---------------------------------------------------------------------------*/
void ADCInit(void)
{
   //==========Setup Power====================================
  PWR_BGREnable();                //bandgap Vref Enable
  PWR_VDDAOpen(LDOC_2V4);  //VDDA Enable

  //==========Setup ADC Clock=================================
#if defined(HAO_2MHZ)
  //HAO=2MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV2,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif

#if defined(HAO_4MHZ)
  //HAO=4MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV4,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif

#if defined(HAO_8MHZ)
  //HAO=8MHz, ADC_CK=1MHz
```

```
ADC_Open(DADC_DHSCKDIV8,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif

#if defined(HAO_17MHZ)
   //HAO=17MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV16,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_
65536);
#endif


#if defined ADC_ChopperMode
   CSFON_Enable();
   ADC_ENINXCH_Enable();
   ADC_DAFM_CHOPPER();
   ADC_ENCH_Enable();
   ADC_Enable();
   ADC_CMFREnable();        //CMFR=1, Comb Filter Reset
#endif

}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

11. 類比 IP(MFC)

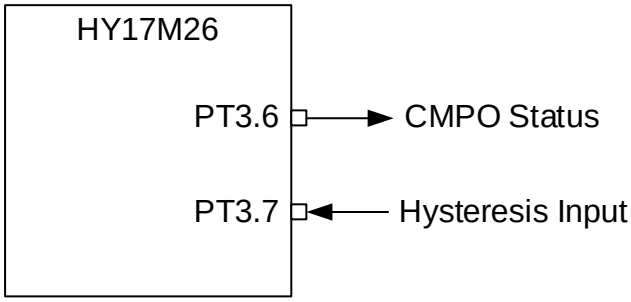
11.1. 範例名稱

MFC

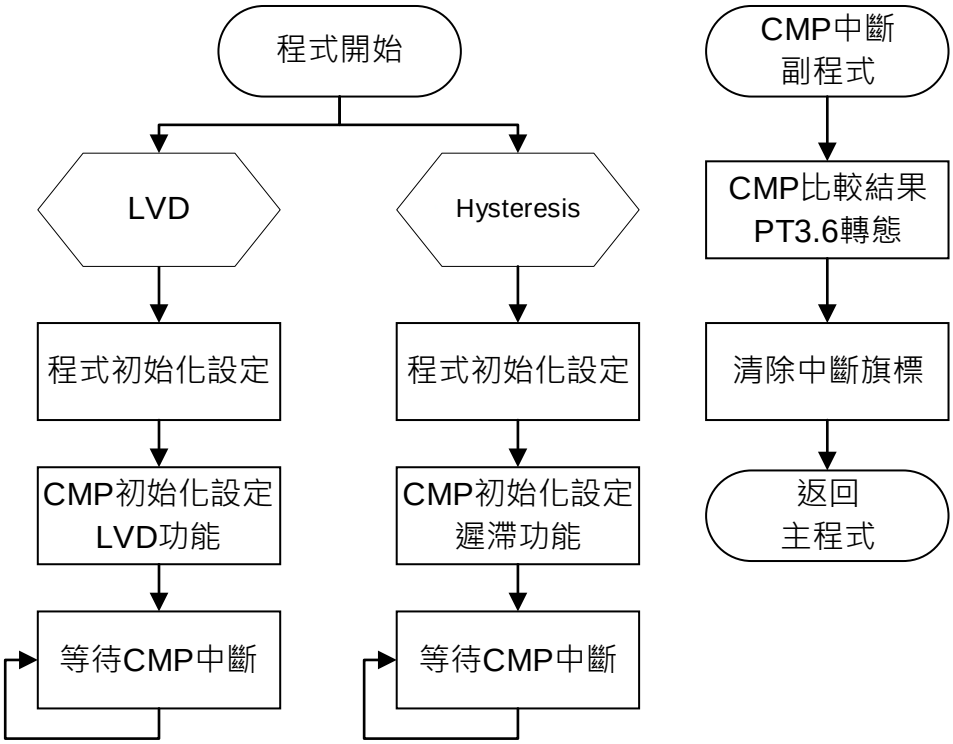
11.2. 範例說明

(1) CMP 使用方式說明
(2) 設置系統工作頻率。
(3) 透過 CMP 相關設定實現 LVD 低電壓檢測。
(4) CMP 初始設置包含, 設置類比電源 VDDA，設置 CMP 的輸入端為 RLO-1.2V，設置 CMP 內建多節點電阻分壓值(RLO)。
(5) 透過 CMP 相關設定實現遲滯控制器。
(6) 設置類比電源 VDDA，設置 CMP 的輸入端為 RLO-CH2(PT3.7)，設置 CMP 內建多節點電阻分壓值(RLO)。
(7) 透過遲滯控制器 CPDM 暫存器與節點控制器 CPDA 暫存器控制遲滯切換區間。當 PT3.7 高於 RLO，PT3.6 輸出”1”，則 CPOB= 0= CPDA　改變節點電壓(RLO)；當 PT3.7 低於 RLO，PT3.6 輸出”0”，則 CPOB=1= CPDA 改變節點電壓。節點選擇器在兩節點之間切換，實現遲滯功能。

11.3. 範例配置



11.4. 軟體流程



11.5. 程式碼

```
#define   USE_HY17M26_2M
/*****************************************************************************
 * MFC_Demo.c                                                                *
 * ------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                           *
 * http://www.hycontek.com/                                                  *
 *                                                                           *
```

```
 * Program Description:                                                    *
 *                                                                          *
 *            HY17M26                                                       *
 *--------------------                                                     *
 *                          |                                               *
 *                    PT3.6|->Output                                       *
 *                          |                                               *
 *                    PT3.7|<-Input                                        *
 *                          |                                               *
 *--------------------                                                     *
 *                                                                          *
 * For External Input                                                      *
 * IC Body: HY17M26                                                         *
 * Project Name : MFC                                                       *
 * Created Date : 2021/6/4 BY Cyril                                        *
 *                                                                          *
 ****************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <PWR.h>
#include <GPIO.h>
#include <RST.h>
#include <ADC.h>
#include <CLK.h>
#include <CMP.h>

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void CMP_LVDInit(void);
void CMP_HysteresisInit(void);
void CLOCKInit(void);
void GPIOInit(void);
void Delay(unsigned int num);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
```

```
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
      CMP_LVDInit();
      //CMP_HysteresisInit();

        GIE_Enable();
        while(1);
}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{

}
/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* LVD Mode Init Function                                                    */
/*---------------------------------------------------------------------------*/
void CMP_LVDInit(void)
{
   CMP_RLOSet(CPRH_VDD,CPRL_OPEN,CPDA_26DIV32,CPDM_DISABLE );    //LVD=1.92V (When VDD >1.92V
PT36=Low, When VDD<1.92V PT36=High)
   //Set CPDA[4:0]
```

```
   //CPDA_6DIV32, LVD=3.39V
   //CPDA_7DIV32, LVD=3.27V
   //CPDA_8DIV32, LVD=3.15V
   //CPDA_9DIV32, LVD=3.04V
   //CPDA_10DIV32, LVD=2.94V
   //CPDA_11DIV32, LVD=2.85V
   //CPDA_12DIV32, LVD=2.76V
   //CPDA_13DIV32, LVD=2.67V
   //CPDA_14DIV32, LVD=2.59V
   //CPDA_15DIV32, LVD=2.52V
   //CPDA_16DIV32, LVD=2.45V
   //CPDA_17DIV32, LVD=2.38V
   //CPDA_18DIV32, LVD=2.32V
   //CPDA_19DIV32, LVD=2.26V
   //CPDA_20DIV32, LVD=2.21V
   //CPDA_21DIV32, LVD=2.15V
   //CPDA_22DIV32, LVD=2.10V
   //CPDA_23DIV32, LVD=2.05V
   //CPDA_24DIV32, LVD=2.01V
   //CPDA_25DIV32, LVD=1.96V
   //CPDA_26DIV32, LVD=1.92V

   CMP_Open(CPPS_1V2,CPNS_RLO,CMPHS_NORMAL,CPOR_NORMAL,ENRCC_PT36);     //RLO=
VDD*((26+20)/(22.5+32+20))

}

/*-------------------------------------------------------------------------*/
/* HysteresisInit Mode Init Function                                       */
/*-------------------------------------------------------------------------*/
void CMP_HysteresisInit(void)
{
        PWR_VDDAOpen(LDOC_2V4);
        CMP_RLOSet(CPRH_VDDA,CPRL_SHORT,CPDA_24DIV32,0x08);
      //CPDM[4:0]=01000
      //CPDA[4:0]=11000, RLO= VDDA*(24/32) <-switch over-> CPDA[4:0]=10000, RLO= VDDA*(16/32)
      CMP_Open(CPPS_CH2,CPNS_RLO,CMPHS_NORMAL,CPOR_NORMAL,ENRCC_PT36);       // When
PT3.7<1.2V PT36=Low, When PT3.7 >1.8V PT36=High,

}

/*-------------------------------------------------------------------------*/
/* Delay Function                                                          */
/*-------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
   unsigned char t;
   for(;ms>0;ms--)            //@HAO=1.843MHz
      for(t=114;t>0;t--)
      { __asm__("NOP"); }
```

```
}


/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

12. 通訊 IP(UART)

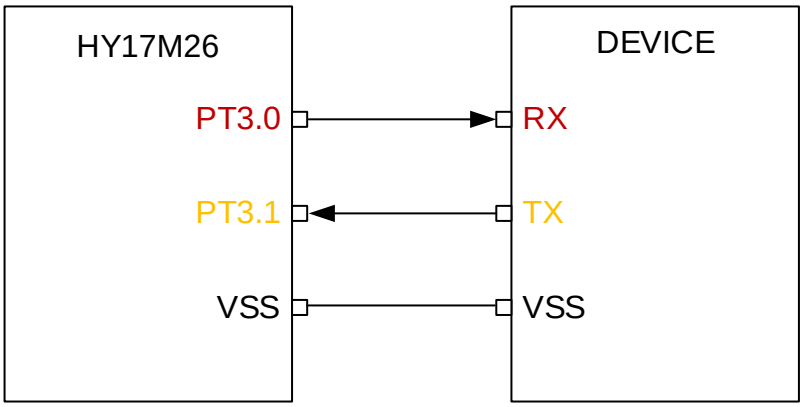
12.1. 範例名稱

UART

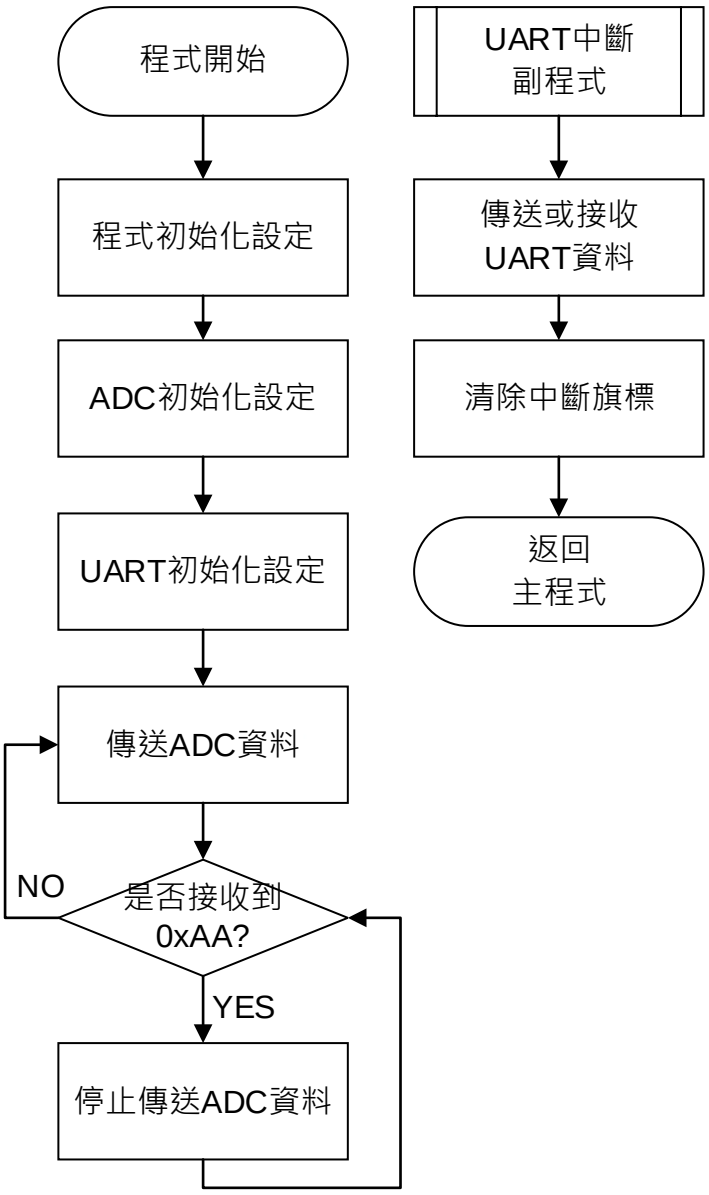
12.2. 範例說明

(1) HY17M26 使用 UART 傳輸 ADC 資料範例程式
(2) 設置系統工作頻率。
(3) 開啟 ADC 功能，設置 ADC 相關設定。
(4) 開啟 UART 功能，設置 UART 頻率源，設置 UART 傳送格式，設置 TX、RX 引腳，設置字串傳送速率 9600。
程式開始執行，UART 會一直送 ADC 資料，直到收到 0xAA，TX 停止傳送。當收到字串不是 0xAA， 則會繼續發送 ADC 資料。

12.3. 範例配置



12.4. 軟體流程



12.5. 程式碼

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * UART_Demo.c                                                                 *
 * ---------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                          *
 * http://www.hycontek.com/                                                *
 *                                                                          *
 * Program Description:                                                     *
 *            HY17M26                                                       *
```

```
 * ------------------                                                    *
 *                   |                                                   *
 *          PT3.0|->TX                                                   *
 *          PT3.1|<-RX                                                   *
 *                   |                                                   *
 *-------------------                                                    *
 *                                                                       *
 * For External Input                                                    *
 * IC Body: HY17M26                                                      *
 * Project Name : UART                                                   *
 * Created Date : 2021/6/4 BY Cyril                                      *
 *                                                                       *
 *************************************************************************/

/*-------------------------------------------------------------------------*/
/* Includes                                                                */
/*-------------------------------------------------------------------------*/
#include <SFRType.h>
#include <INT.h>
#include <CLK.h>
#include <RST.h>
#include <UART.h>
#include <PWR.h>
#include <ADC.h>
#include <GPIO.h>
/*-------------------------------------------------------------------------*/
/* DEFINITIONS                                                             */
/*-------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

//UART PORT
#define Tx_PT30
//#define Tx_PT32
//#define Tx_PT34
//#define Tx_PT36

#define   Uart_RX_BufferSize 1
#define   Uart_TX_BufferSize 10

//#define UART_ABD
//#define ADC_ChopperMode
/*-------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                     */
/*-------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
```

```
void SysInit(void);
void CLOCKInit(void);
void UART0Init(void);
void ADCInit(void);
void SendUART(signed long    data);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
volatile typedef union _Flag
{
  unsigned int    _byte;
    struct
    {
       unsigned b_ADCdone:1;
       unsigned b_UART_TxDone:1;
       unsigned b_UART_RxDone:1;
       unsigned Reserved:13;
    };
} Flag;
//signed long
//range 8388607~-8388608 (dec)
//range 0x007F FFFF ~ 0xFF80 0000 (hex)
signed long    ADC_ISRTemp;
signed long    ADC_ISRTemp_Buffer[8];
signed long    ADCData1;
Flag     Flagbits;
unsigned char UartTxBuffer[Uart_TX_BufferSize]={0};
unsigned char UartRxBuffer[Uart_RX_BufferSize]={0};
unsigned char UartTxIndex,UartTxLength;
unsigned char UartRxIndex,UartRxLength;
unsigned char STOP_TO_SEND_UART=0;
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
    //ADC_UART Demo
    UartTxIndex= 0;
    UartRxIndex= 0;
    Flagbits.b_ADCdone= 0;
    Flagbits.b_UART_TxDone= 0;
    Flagbits.b_UART_RxDone= 0;

    CLOCKInit();
    ADCInit();
    UART0Init();
    Flagbits.b_UART_TxDone=1;
    GIE_Enable();

    while(1)
    {

        if(Flagbits.b_UART_RxDone==1)
      {
            if( UartRxBuffer[0]==0xAA)          //If receive 0xAA, stop to send UART
            {
                STOP_TO_SEND_UART= 1;
```

```
                Flagbits.b_UART_RxDone= 0;
            }
            else
                STOP_TO_SEND_UART=0;
        }


        Flagbits.b_ADCdone=0;
        while(Flagbits.b_ADCdone==0);    //until get ADC_ISRTemp
        Flagbits.b_ADCdone=0;

        if(Flagbits.b_UART_TxDone==1 && STOP_TO_SEND_UART==0)
        {
            ADC_INT_Disable();
            SendUART(ADC_ISRTemp);
            ADC_INT_Enable();
        }
    }

}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{

    NOP();

    //ADC Event
    if(ADC_INT_IsFlag())
    {
        ADC_INT_ClearFlag();
        ADC_ISRTemp=ADCR; //ADC_GetData();
        Flagbits.b_ADCdone=1;
    }

    //UART0 RX Event
    if(UART0_INT_RCIsFlag())
    {
        UartRxBuffer[UartRxIndex]=RC0REG;
        UartRxIndex++;
        if(UartRxIndex>=Uart_RX_BufferSize)
        {
            UartRxIndex=0;
            Flagbits.b_UART_RxDone=1;
        }
        UART0_INT_RCClearFlag();
    }

    //UART0 TX Event
    if(UART0_INT_TXIsFlag())
    {
        if(Flagbits.b_UART_TxDone==0)
        {
            TX0R=UartTxBuffer[UartTxIndex++];
            UART0_INT_TXClearFlag();
```

```
        if(UartTxIndex>=UartTxLength)
        {
          UART0_INT_TXEnable();       //TXIE=1b
          UART0_INT_RCEnable();    //RCIE=1b
          Flagbits.b_UART_TxDone=1;
          UartTxIndex=0;
        }
      }
      if(Flagbits.b_UART_TxDone==1)
      {
        UART0_INT_TXDisable();       //TXIE=0b
        UART0_INT_RCEnable();    //RCIE=1b
        UART0_INT_TXClearFlag();
      }
    }

}

/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* UART Init Function                                                        */
/*---------------------------------------------------------------------------*/
void UART0Init(void)
{
  //GPIO Init
#if defined(Tx_PT30)
  GPIO_GTXSet(GTX_PT30);
#endif
#if defined(Tx_PT32)
  GPIO_GTXSet(GTX_PT32);
#endif
#if defined(Tx_PT34)
  GPIO_GTXSet(GTX_PT34);
#endif
#if defined(Tx_PT36)
  GPIO_GTXSet(GTX_PT36);
#endif

  //BRGR[15:0]=((CPUCK/Baudrate)/4)-1
#if defined(HAO_2MHZ)
  //((1843000/9600)/4)-1 =47, HAO=1.843M, baudrate=9600
  UART0_Open(47 ,8 ,PARITY_None);
#endif
#if defined(HAO_4MHZ)
  //((4147000/9600)/4)-1 =107, HAO=4.147M, baudrate=9600
  UART0_Open(107 ,8 ,PARITY_None);
#endif
#if defined(HAO_8MHZ)
  //((8755000/9600)/4)-1 =227, HAO=8.755M, baudrate=9600
  UART0_Open(227 ,8 ,PARITY_None);
#endif
#if defined(HAO_16MHZ)
  //((17510000/9600)/4)-1 =455, HAO=17.51M, baudrate=9600
  UART0_Open(455 ,8 ,PARITY_None);
```

```
#endif

#ifdef UART_ABD
   UART0_Open(0 ,8 ,PARITY_None);
   UART0_WUEDisable();                    // Wake-up disable
   UART0_ABDEnable();                     //Enable Auto Baudrate
   while((BA0CN & 0x01) == 1 );          //Wait for master send 055H
   UART0_INT_RCClearFlag();               //Clear RC interrupt flag
#endif

   TXIE_Enable();
   RCIE_Enable();
}
/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* ADC Init Function                                                         */
/*---------------------------------------------------------------------------*/
void ADCInit(void)
{
    //==========Setup Power=====================================
   PWR_BGREnable();                  //bandgap Vref Enable
   PWR_VDDAOpen(LDOC_2V4);   //VDDA Enable

   //==========Setup ADC Clock================================
#if defined(HAO_2MHZ)
   //HAO=2MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV2,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif
```

```
#if defined(HAO_4MHZ)
   //HAO=4MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV4,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif

#if defined(HAO_8MHZ)
   //HAO=8MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV8,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif

#if defined(HAO_17MHZ)
   //HAO=17MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV16,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_
65536);
#endif


#if defined ADC_ChopperMode
   CSFON_Enable();
   ADC_ENINXCH_Enable();
   ADC_DAFM_CHOPPER();
   ADC_ENCH_Enable();
   ADC_Enable();
   ADC_CMFREnable();        //CMFR=1, Comb Filter Reset
#endif

}
/*---------------------------------------------------------------------------*/
/* Function Name: void SendUART(int data)                                    */
/* Description    : UART send data.                                          */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void SendUART(signed long    data)
{

   //display range +8388607~-8388608 (dec)
   ADCData1=data;
   if((ADCData1<0)||(ADCData1>0x80000000))    // plus-minus sign judgment
   {
          ADCData1=~ADCData1;
          ADCData1++;
        UartTxBuffer[0]='-';
   }
   else
   {
      UartTxBuffer[0]='+';
   }
   UartTxBuffer[7]=ADCData1%10 | '0';    //unit
   ADCData1=ADCData1/10;
   UartTxBuffer[6]=ADCData1%10 | '0';    //unit
```

```
    ADCData1=ADCData1/10;
    UartTxBuffer[5]=ADCData1%10 | '0';    //ten
    ADCData1=ADCData1/10;
    UartTxBuffer[4]=ADCData1%10 | '0';    //hundred
    ADCData1=ADCData1/10;
    UartTxBuffer[3]=ADCData1%10 | '0';    //thousand
    ADCData1=ADCData1/10;
    UartTxBuffer[2]=ADCData1%10 | '0';    //ten thousand
    ADCData1=ADCData1/10;
    UartTxBuffer[1]=ADCData1%10 | '0';    //hundred thousand
    ADCData1=ADCData1/10;
    UartTxBuffer[8]='\r';
    UartTxBuffer[9]='\n';
    Flagbits.b_UART_TxDone=0;
    UartTxLength=10;
    UartTxIndex=0;
    TXIE_Enable();       //Enable UART Tx Interrupt;
    RCIE_Enable();     //Enable UART Rx Interrupt
    while(!Flagbits.b_UART_TxDone);    //If    Flagbits.b_UART_TxDone=DISABLE, stop at here
}

/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
    unsigned char t;
    for(;ms>0;ms--)              //@HAO=1.843MHz
       for(t=114;t>0;t--)
       { __asm__("NOP"); }
}


/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

13. 通訊 IP(UART2)

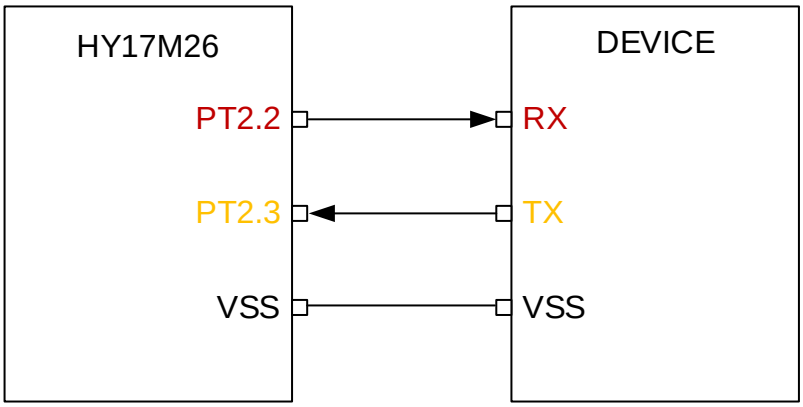
13.1. 範例名稱

UART2

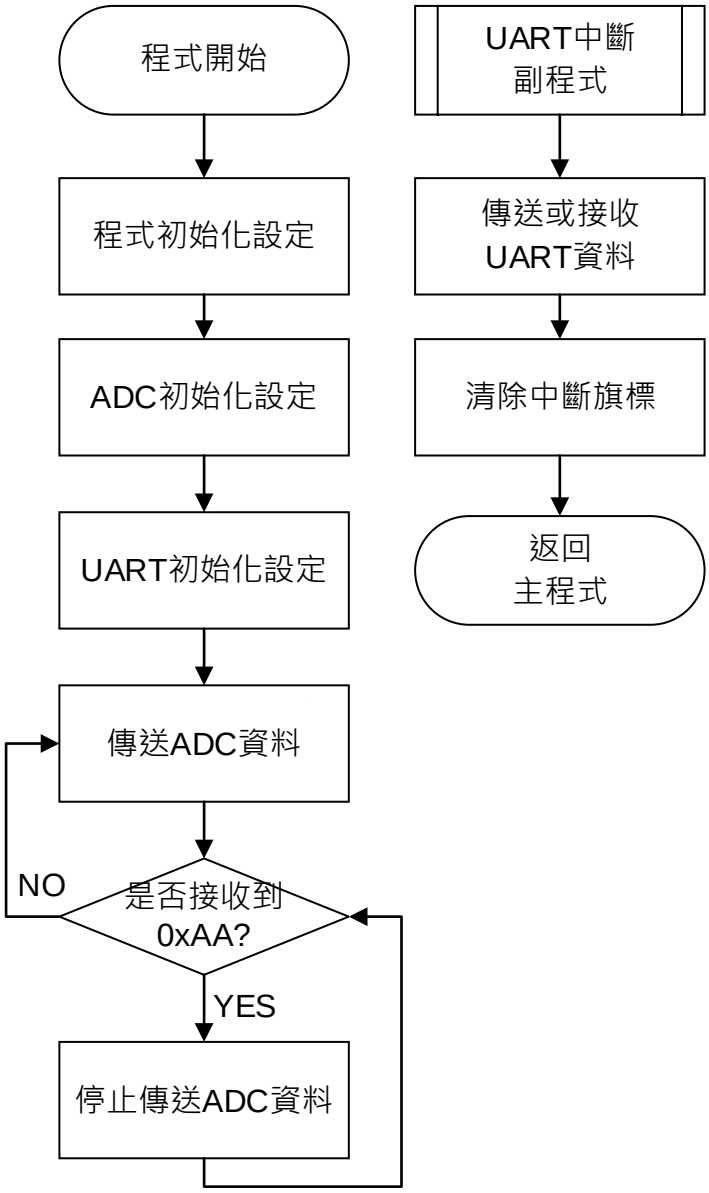
13.2. 範例說明

(5) HY17M26 使用 UART2 傳輸 ADC 資料範例程式
(6) 設置系統工作頻率。
(7) 開啟 ADC 功能，設置 ADC 相關設定。
(8) 開啟 UART2 功能，設置 UART2 頻率源，設置 UART2 傳送格式，設置 TX、RX 引腳，設置字串傳送速率 9600。
程式開始執行，UART2 會一直送 ADC 資料，直到收到 0xAA，TX 停止傳送。當收到字串不是 0xAA， 則會繼續發送 ADC 資料。

13.3. 範例配置



13.4. 軟體流程



13.5. 程式碼

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * UART2_Demo.c                                                                *
 * ---------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                          *
 * http://www.hycontek.com/                                                 *
 *                                                                          *
 * Program Description:                                                     *
```

```
 *            HY17M26                                                  *
 * ------------------                                           *
 *                         |                                           *
 *                   PT2.2|->TX                                        *
 *                   PT2.3|<-RX                                        *
 *                         |                                           *
 *-------------------                                           *
 * For External Input                                              *
 * IC Body: HY17M26                                                    *
 * Project Name : UART2                                               *
 * Created Date : 2021/6/4 BY Cyril                                *
 *                                                                     *
 *****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <INT.h>
#include <CLK.h>
#include <RST.h>
#include <UART.h>
#include <PWR.h>
#include <ADC.h>
#include <GPIO.h>
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

//UART PORT
//#define Tx2_PT11
//#define Tx2_PT15
#define Tx2_PT22
//#define Tx2_PT26

#define   Uart_RX_BufferSize 1
#define   Uart_TX_BufferSize 10

//#define UART2_ABD
//#define ADC_ChopperMode
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void SysInit(void);
void CLOCKInit(void);
```

```
void UART2Init(void);
void ADCInit(void);
void SendUART(signed long    data);
void Delay(unsigned int num);
void Delayms(unsigned int ms);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
volatile typedef union _Flag
{
  unsigned int    _byte;
    struct
    {
       unsigned b_ADCdone:1;
       unsigned b_UART_TxDone:1;
       unsigned b_UART_RxDone:1;
       unsigned Reserved:13;
    };
} Flag;
//signed long
//range 8388607~-8388608 (dec)
//range 0x007F FFFF ~ 0xFF80 0000 (hex)
signed long    ADC_ISRTemp;
signed long    ADC_ISRTemp_Buffer[8];
signed long    ADCData1;
Flag     Flagbits;
unsigned char UartTxBuffer[Uart_TX_BufferSize]={0};
unsigned char UartRxBuffer[Uart_RX_BufferSize]={0};
unsigned char UartTxIndex,UartTxLength;
unsigned char UartRxIndex,UartRxLength;
unsigned char STOP_TO_SEND_UART=0;
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
    //ADC_UART Demo
    UartTxIndex= 0;
    UartRxIndex= 0;
    Flagbits.b_ADCdone= 0;
    Flagbits.b_UART_TxDone= 0;
    Flagbits.b_UART_RxDone= 0;

    CLOCKInit();
    ADCInit();
    UART2Init();
    Flagbits.b_UART_TxDone=1;
    GIE_Enable();

    while(1)
    {

        if(Flagbits.b_UART_RxDone==1)
      {
            if( UartRxBuffer[0]==0xAA)          //If receive 0xAA, stop to send UART
            {
                STOP_TO_SEND_UART= 1;
```

```
                Flagbits.b_UART_RxDone= 0;
            }
            else
                STOP_TO_SEND_UART=0;
        }


        Flagbits.b_ADCdone=0;
        while(Flagbits.b_ADCdone==0);    //until get ADC_ISRTemp
        Flagbits.b_ADCdone=0;

        if(Flagbits.b_UART_TxDone==1 && STOP_TO_SEND_UART==0)
        {
            ADC_INT_Disable();
            SendUART(ADC_ISRTemp);
            ADC_INT_Enable();
        }
    }

}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{

    NOP();

    //ADC Event
    if(ADC_INT_IsFlag())
    {
        ADC_INT_ClearFlag();
        ADC_ISRTemp=ADCR; //ADC_GetData();
        Flagbits.b_ADCdone=1;
    }

    //UART0 RX Event
    if(UART2_INT_RCIsFlag())
    {
        UartRxBuffer[UartRxIndex]=RC2REG;
        UartRxIndex++;
        if(UartRxIndex>=Uart_RX_BufferSize)
        {
            UartRxIndex=0;
            Flagbits.b_UART_RxDone=1;
        }
        UART2_INT_RCClearFlag();
    }

    //UART0 TX Event
    if(UART2_INT_TXIsFlag())
    {
        if(Flagbits.b_UART_TxDone==0)
        {
            TX2R=UartTxBuffer[UartTxIndex++];
            UART2_INT_TXClearFlag();
```

```
        if(UartTxIndex>=UartTxLength)
        {
          UART2_INT_TXEnable();       //TXIE=1b
          UART2_INT_RCEnable();    //RCIE=1b
          Flagbits.b_UART_TxDone=1;
          UartTxIndex=0;
        }
      }
      if(Flagbits.b_UART_TxDone==1)
      {
        UART2_INT_TXDisable();       //TXIE=0b
        UART2_INT_RCEnable();    //RCIE=1b
        UART2_INT_TXClearFlag();
      }
    }

}

/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* UART Init Function                                                        */
/*---------------------------------------------------------------------------*/
void UART2Init(void)
{
  //GPIO Init
#if defined(Tx2_PT11)
  GPIO_GTX2Set(GTX2_PT11);
#endif
#if defined(Tx2_PT15)
  GPIO_GTX2Set(GTX2_PT15);
#endif
#if defined(Tx2_PT22)
  GPIO_GTX2Set(GTX2_PT22);
#endif
#if defined(Tx2_PT26)
  GPIO_GTX2Set(GTX2_PT26);
#endif

  //BRGR[15:0]=((CPUCK/Baudrate)/4)-1
#if defined(HAO_2MHZ)
  //((1843000/9600)/4)-1 =47, HAO=1.843M, baudrate=9600
  UART2_Open(47 ,8 ,PARITY_None);
#endif
#if defined(HAO_4MHZ)
  //((4147000/9600)/4)-1 =107, HAO=4.147M, baudrate=9600
  UART2_Open(107 ,8 ,PARITY_None);
#endif
#if defined(HAO_8MHZ)
  //((8755000/9600)/4)-1 =227, HAO=8.755M, baudrate=9600
  UART2_Open(227 ,8 ,PARITY_None);
#endif
#if defined(HAO_16MHZ)
  //((17510000/9600)/4)-1 =455, HAO=17.51M, baudrate=9600
  UART2_Open(455 ,8 ,PARITY_None);
```

```
#endif

#ifdef UART2_ABD
   UART2_Open(0 ,8 ,PARITY_None);
   UART2_WUEDisable();                    // Wake-up disable
   UART2_ABDEnable();                     //Enable Auto Baudrate
   while((BA2CN & 0x01) == 1 );          //Wait for master send 055H
   UART2_INT_RCClearFlag();               //Clear RC interrupt flag
#endif

   TX2IE_Enable();
   RC2IE_Enable();
}
/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* ADC Init Function                                                         */
/*---------------------------------------------------------------------------*/
void ADCInit(void)
{
    //==========Setup Power=====================================
   PWR_BGREnable();                  //bandgap Vref Enable
   PWR_VDDAOpen(LDOC_2V4);   //VDDA Enable

   //==========Setup ADC Clock================================
#if defined(HAO_2MHZ)
   //HAO=2MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV2,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif
```

```
#if defined(HAO_4MHZ)
   //HAO=4MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV4,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif

#if defined(HAO_8MHZ)
   //HAO=8MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV8,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_6
5536);
#endif

#if defined(HAO_17MHZ)
   //HAO=17MHz, ADC_CK=1MHz

ADC_Open(DADC_DHSCKDIV16,INP_AI0,INN_AI1,VRH_VDDA,VRL_VSS,ADGN_1,VREGN_DIV2,DCSET_P0,OSR_
65536);
#endif


#if defined ADC_ChopperMode
   CSFON_Enable();
   ADC_ENINXCH_Enable();
   ADC_DAFM_CHOPPER();
   ADC_ENCH_Enable();
   ADC_Enable();
   ADC_CMFREnable();        //CMFR=1, Comb Filter Reset
#endif

}
/*---------------------------------------------------------------------------*/
/* Function Name: void SendUART(int data)                                    */
/* Description    : UART send data.                                          */
/* Arguments      : None.                                                    */
/* Return Value : None.                                                      */
/* Remark         :                                                          */
/*---------------------------------------------------------------------------*/
void SendUART(signed long    data)
{

   //display range +8388607~-8388608 (dec)
   ADCData1=data;
   if((ADCData1<0)||(ADCData1>0x80000000))    // plus-minus sign judgment
   {
          ADCData1=~ADCData1;
          ADCData1++;
        UartTxBuffer[0]='-';
   }
   else
   {
      UartTxBuffer[0]='+';
   }
   UartTxBuffer[7]=ADCData1%10 | '0';    //unit
   ADCData1=ADCData1/10;
   UartTxBuffer[6]=ADCData1%10 | '0';    //unit
```

```
    ADCData1=ADCData1/10;
    UartTxBuffer[5]=ADCData1%10 | '0';    //ten
    ADCData1=ADCData1/10;
    UartTxBuffer[4]=ADCData1%10 | '0';    //hundred
    ADCData1=ADCData1/10;
    UartTxBuffer[3]=ADCData1%10 | '0';    //thousand
    ADCData1=ADCData1/10;
    UartTxBuffer[2]=ADCData1%10 | '0';    //ten thousand
    ADCData1=ADCData1/10;
    UartTxBuffer[1]=ADCData1%10 | '0';    //hundred thousand
    ADCData1=ADCData1/10;
    UartTxBuffer[8]='\r';
    UartTxBuffer[9]='\n';
    Flagbits.b_UART_TxDone=0;
    UartTxLength=10;
    UartTxIndex=0;
    TX2IE_Enable();       //Enable UART Tx Interrupt;
    RC2IE_Enable();     //Enable UART Rx Interrupt
    while(!Flagbits.b_UART_TxDone);    //If    Flagbits.b_UART_TxDone=DISABLE, stop at here
}

/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
    unsigned char t;
    for(;ms>0;ms--)             //@HAO=1.843MHz
       for(t=114;t>0;t--)
       { __asm__("NOP"); }
}


/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*//
```

14. 通訊 IP(I2C)

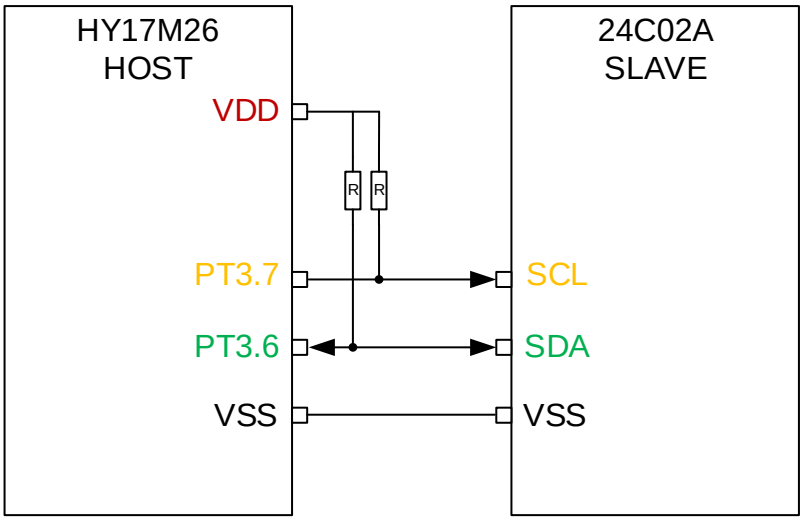
14.1. 範例名稱

I2C_Master

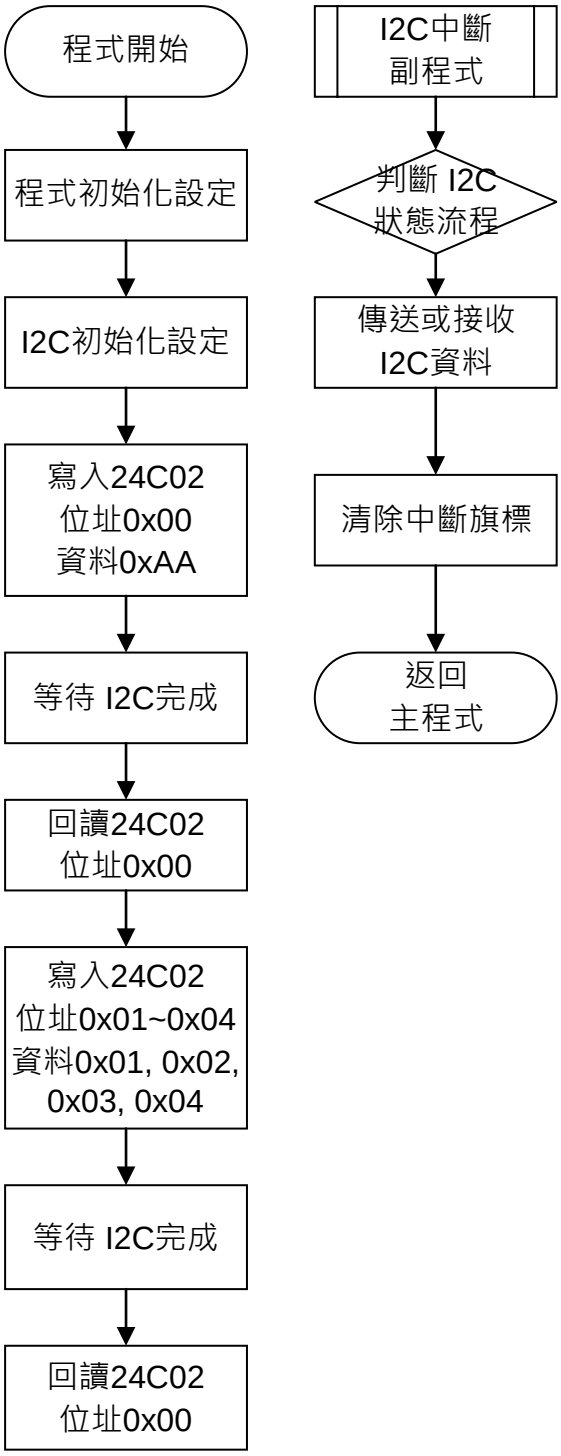
14.2. 範例說明

(1) HY17M26 工作在 I2C Master Mode, 對 I2C EEPROM 24C02 做寫入與讀取控制範例.
(2) 範例程式內容包含 HY17M26 之 I2C Master 設置初始化過程，並且對一個 I2C EEPROM(24C02)裝置做單次的資料寫入和讀取與連續的資料寫入讀取範例
(3) 程式第一段功能為做單次性的資料寫入和讀取在 EEPROM 裝置，首先由 I2C Master 寫入資料 0xAA 在 EEPROM 的 WORD ADDRESS 0x00，然後再下 Read 指令讀回 EEPROM 的 WORD ADDRESS 0x00 位置。
(4) 程式第二段功能為做連續性的資料寫入和讀取在 EEPROM 裝置，從 EEPROM 的 WORD ADDRESS 0x01 開始寫入 4bytes 資料直到 WORD ADDRESS 0x04，再依序從 EEPROM 的 WORD ADDRESS 0x00 連續讀取，分別為 2、5、6 筆資料，觀察資料是否已經正確被寫入。
(5) 在本文範例程式裡面， 當執行完程式第一段和第二段的正確的執行結果為 EEPROM Address 0x00 資料等於 0xAA，Address 0x01 資料等於 0x02，Address 0x02 資料等於 0x03，Address 0x03 資料等於 0x04，Address 0x04 資料等於 0x05。

14.3. 範例配置



14.4. 軟體流程


程式開始
程式初始化設定
I2C初始化設定
寫入24C02
位址0x00
資料0xAA
等待 I2C完成
回讀24C02
位址0x00
寫入24C02
位址0x01~0x04
資料0x01, 0x02,
0x03, 0x04
等待 I2C完成
回讀24C02
位址0x00
I2C中斷
副程式
判斷 I2C
狀態流程
傳送或接收
I2C資料
清除中斷旗標
返回
主程式

14.5. 程式碼

說明 ： 在此僅貼上 I2C_Master.C 程式內容做參考.

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * I2C_Master_Demo.c                                                           *
 * --------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                             *
 * http://www.hycontek.com/                                                    *
 *                                                                             *
 * Program Description:                                                        *
 *                                                                             *
 *            HY17M26                    24C02A                                *
 *---------------------          ---------------                               *
 *                            |          |                                     *
 *                            |          |                                     *
 *                       VDD |-----> | VDD                                     *
 *                     PT3.7 |-----> | SCL                                     *
 *                     PT3.6 |<----> | SDA                                     *
 *                       VSS |-----> | VSS                                     *
 *                            |          |                                     *
 *                            |          |                                     *
 *---------------------          ---------------                               *
 *                                                                             *
 * EEPROM Address                                                              *
 * addr[0x00]=0xaa                                                             *
 * addr[0x01]=0x02                                                             *
 * addr[0x02]=0x03                                                             *
 * addr[0x03]=0x04                                                             *
 * addr[0x04]=0x05                                                             *
 *                                                                             *
 * For External Input                                                          *
 * IC Body: HY17M26                                                            *
 * Project Name : I2C_Master                                                   *
 * Created Date : 2021/6/4 BY Cyril                                            *
 *                                                                             *
 *******************************************************************************/
/*------------------------------------------------------------------------------*/
/* Includes                                                                     */
/*------------------------------------------------------------------------------*/
#include <SFRType.h>
#include <INT.h>
#include <CLK.h>
#include <RST.h>
#include <I2C.h>
#include <GPIO.h>
/*------------------------------------------------------------------------------*/
/* DEFINITIONS                                                                  */
/*------------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
```

```
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

//I2C PORT
//#define SCL_PT31
//#define SCL_PT33
//#define SCL_PT35
#define SCL_PT37

#define EEPROM_ADDR       0xA0    // Device address for slave target

#define I2CBufferSize    10
#define I2C_WRITE    0x00           // I2C WRITE command
#define I2C_READ     0x01           // I2C READ command
#define I2C_Delay     100000    //@2MHz, Delay 5s
/*-------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                     */
/*-------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void I2CInit(void);
void CLOCKInit(void);
void EEPROM_ByteWrite(unsigned int addr, unsigned char dat);
unsigned char EEPROM_ByteRead(unsigned int addr);
void EEPROM_WriteArray(unsigned int dest_addr, unsigned char* src_addr,
                       unsigned int len);
void EEPROM_WriteArray1(unsigned int dest_addr, unsigned char* src_addr,
                       unsigned int len);
void EEPROM_ReadArray(unsigned char* dest_addr, unsigned int src_addr,
                      unsigned int len);
/*-------------------------------------------------------------------------*/
/* Global CONSTANTS                                                        */
/*-------------------------------------------------------------------------*/
unsigned char I2C_Read_Buffer[I2CBufferSize];
unsigned char I2C_RW;
unsigned char I2C_TARGET;          // Target I2C slave address
unsigned char I2C_Sendbuf[I2CBufferSize];
unsigned char I2C_Recbuf[I2CBufferSize];
unsigned char I2C_EndFlag;
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;
unsigned long I2C_Timeout;
unsigned char EEPROM_WriteData[4] = {0x02,0x03,0x04,0x05};
unsigned char i,temp;
/*-------------------------------------------------------------------------*/
/* Main Function                                                           */
/*-------------------------------------------------------------------------*/
void main(void)
{
    CLOCKInit();
    I2CInit();
```

```
    for(i=0;i<I2CBufferSize;i++)
    {
        I2C_Read_Buffer[i]=0x00;      //CLR I2C_Read_Buffer[]
    }

    GIE_Enable();

    // I2C single I2C_WRITE & I2C_READ
    EEPROM_ByteWrite(0x00,0xAA);                    //Single Byte I2C_WRITE word address 0x00, and I2C_WRITE
data 0x01
    Delayms(10);                                    //Delay for EEPROM 24C02 I2C_WRITE Cycle Time
    I2C_Read_Buffer[0]=EEPROM_ByteRead(0x00);   //Single Byte I2C_READ word address 0x00, the I2C_READ
value is 0x01

    // I2C sequential I2C_WRITE & I2C_READ
    EEPROM_WriteArray(0x01,EEPROM_WriteData,4); //I2C_WRITE array data to the word address from 0x01 to
0x04
    Delayms(10);                                    //Delay for EEPROM 24C02 I2C_WRITE Cycle Time
    EEPROM_ReadArray(I2C_Read_Buffer, 0x00, 2); //sequential I2C_READ data from 0x00 to 0x01
    EEPROM_ReadArray(I2C_Read_Buffer, 0x00, 5); //sequential I2C_READ data from 0x00 to 0x04
    EEPROM_ReadArray(I2C_Read_Buffer, 0x00, 6); //sequential I2C_READ data from 0x00 to 0x05

    while(1);
}
/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{
    if(I2CIF_IsFlag() &&I2C_EndFlag==0)   //Get I2C Interrupt Flag
    {
    switch(I2C0_STAState())
    {
        case 0x90:   //MACTFlag+RWFlag
                    //START has been transmitted
            I2C0_SendData(I2C_TARGET);   //Send Slave Address & R/W Bit
            I2C0_Ctrl(0,0,0,0);              //Clear all I2C flag
            break;

        case 0x84:   //MACTFlag+ACKFlag
                    //Slave A + W has been transmitted. ACK has been received.
            I2C0_SendData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            I2C0_Ctrl(0,0,0,0);         //Clear all I2C flag
            break;

        case 0x80: //MACTFlag
                //Slave A + W has been transmitted. ACK has been received.
                I2C0_SendData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            I2C0_Ctrl(0,0,0,0);         //Clear all I2C flag
            break;

        case 0x8C:   //MACTFlag+DFFlag+ACKFlag
                //DATA has been transmitted and ACK has been received
            if(I2C_DataTxIndex<I2C_DataTxLen)
            {
              I2C0_SendData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
              I2C0_Ctrl(0,0,0,0);       // Clear all I2C flag
```

```
            }
            else
            {
                if(I2C_RW == I2C_WRITE)
                {
                  I2C0_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
                  I2C_Timeout=I2C_Delay;    //HAO=2MHz, wait 5s
                  while(I2C_Timeout != 0)
                  {
                      I2C_Timeout--;
                    if(STA0== 0x30)         //A STOP has been transmitted
                    {
                      I2C_EndFlag=1;
                      break;
                     }
                  }
                }
                else if(I2C_RW == I2C_READ)
                  I2C0_Ctrl(1,0,0,0);    //I2C as master sends START signal
                  I2C_DataTxIndex=0;
            }
            break;

case 0x88:    //MACTFlag+DFFlag
                //DATA has been transmitted and NACK has been received
            I2C0_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
            I2C_DataTxIndex=0;
          I2C_Timeout=I2C_Delay;    //HAO=2MHz, wait 5s
          while(I2C_Timeout != 0)
          {
              I2C_Timeout--;
            if(STA0== 0x30)         //A STOP has been transmitted
            {
              I2C_EndFlag=1;
              break;
             }
          }
            break;

case 0xB0:
                //A repeated START has been transmitted.
            I2C0_SendData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
            I2C0_Ctrl(0,0,0,0);       //Clear all I2C flag
            break;

case 0x94:    //MACTFlag+RWFlag+ACKFlag
                //Slave A + R has been transmitted. ACK has been received.
            if(I2C_DataRxLen>1)
            {
              I2C0_Ctrl(0,0,0,1);    //Set ACK bit
            }
          else
          {
              I2C0_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;
```

```
            case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
                        //Data byte has been received. ACK has been transmitted.
                    I2C0_ReceiveDATA(I2C_Recbuf[I2C_DataRxIndex++]);
                    if((I2C_DataRxLen-1)>I2C_DataRxIndex)
                    {
                       I2C0_Ctrl(0,0,0,1);    //Set ACK bit
                    }
                    else
                    {
                       I2C0_Ctrl(0,0,0,0);    //Clear all I2C flag
                    }
                    break;

            case 0x98:    //MACTFlag+RWFlag+DFFlag
                        //Data byte has been received. NACK has been transmitted.
                    I2C0_ReceiveDATA(I2C_Recbuf[I2C_DataRxIndex++]);
                    I2C0_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
                  I2C_Timeout=I2C_Delay;    //HAO=2MHz, wait 5s
                  while(I2C_Timeout != 0)
                  {
                      I2C_Timeout--;
                     if(STA0== 0x30)        //A STOP has been transmitted
                     {
                        I2C_EndFlag=1;
                        break;
                      }
                  }
                  break;

            default:
                    I2C0_Ctrl(0,0,0,0);    //Clear all I2C flag
                    I2C_EndFlag=1;
                    break;
      }
           I2C0_I2CINT_CLEAR();                //CLR I2C INT Flag
           I2C0_I2CER_CLEAR();          //CLR I2C error Flag
           I2CIF_ClearFlag();                          //CLR I2CIF
      }

      if(I2CERIF_IsFlag())    //Get I2C Error Interrupt Flag
      {
          I2C_EndFlag=1;
            I2C0_I2CINT_CLEAR();                //CLR I2C INT Flag
            I2C0_I2CER_CLEAR();          //CLR I2C error Flag
            I2CERIF_ClearFlag();          //CLR I2CERIF
          I2C0_Ctrl(0,0,0,0);               //Clear all I2C flag
      }

}
/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
```

```
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif


}

/*---------------------------------------------------------------------------*/
/* I2C Init Function                                                         */
/*---------------------------------------------------------------------------*/
void I2CInit(void)
{
   //GPIO Init
#if defined(SCL_PT31)
   GPIO_GSCLSet(GSCL_PT31);
#endif
#if defined(SCL_PT33)
   GPIO_GSCLSet(GSCL_PT33);
#endif
#if defined(SCL_PT35)
   GPIO_GSCLSet(GSCL_PT35);
#endif
#if defined(SCL_PT37)
   GPIO_GSCLSet(GSCL_PT37);
#endif

#if defined(HAO_2MHZ)
   I2C0_Open(40);    //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
#endif
#if defined(HAO_4MHZ)
   I2C0_Open(9);    //Default CPU clock is 1MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 4000000/[4*(9+1)]=100kHz
#endif
#if defined(HAO_8MHZ)
   I2C0_Open(19);    //Default CPU clock is 8MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 8000000/[4*(19+1)]=100kHz
#endif
#if defined(HAO_17MHZ)
   I2C0_Open(39);    //Default CPU clock is 16MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]=
16000000/[4*(39+1)]=100kHz
#endif

}
```

```
/*---------------------------------------------------------------------------*/
/* EEPROM_ByteWrite ()                                                       */
/*                                                                           */
/* Return Value : None                                                       */
/* Parameters     :                                                          */
/* 1) unsigned int addr - address to I2C_WRITE in the EEPROM                 */
/*                                                                           */
/* 2) unsigned char dat-data to I2C_WRITE to the address <addr> in the EEPROM */
/*                                range is full range of character: 0 to 255 */
/*                                                                           */
/* This function writes the value in <dat> to location <addr> in the EEPROM  */
/* then polls the EEPROM until the I2C_WRITE is complete.                    */
/*                                                                           */
/*---------------------------------------------------------------------------*/
void EEPROM_ByteWrite(unsigned int addr, unsigned char dat)
{
   I2C_TARGET    = EEPROM_ADDR;     //Device_ID_Addr=0XA0

   I2C_Sendbuf[0]   = (addr&0xFF);   //24Cxx Memory Address
   I2C_Sendbuf[1]   = dat;           //24Cxx Memory Data
   I2C_DataTxLen=2;
   I2C_DataTxIndex=0;
   I2C_EndFlag=0;
   I2C_RW = I2C_WRITE;

   I2C0_Ctrl(1,0,0,0);   // I2C as master sends START signal
   while(I2C_EndFlag==0);

}

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

   I2C_TARGET    = EEPROM_ADDR;   //Device_ID_Addr=0XA0

   I2C_Sendbuf[0]   = (addr&0xFF);   //24Cxx Memory Address
   I2C_DataTxLen=1;
   I2C_DataTxIndex=0;
   I2C_DataRxLen=1;
   I2C_DataRxIndex=0;
   I2C_EndFlag=0;
   I2C_RW = I2C_READ;

   I2C0_Ctrl(1,0,0,0);   // I2C as master sends START signal
   while(I2C_EndFlag==0);

   retval=I2C_Recbuf[0];
   return retval;
}

/*---------------------------------------------------------------------------*/
/* EEPROM_WriteArray()                                                       */
```

```
/*---------------------------------------------------------------------------*/
/* Return Value : None                                                       */
/* Parameters     :                                                          */
/* 1) unsigned int dest_addr-beginning address to I2C_WRITE to in the EEPROM  */
/* 2) unsigned char* src_addr - pointer to the array of data to be written    */
/*                                range is full range of character: 0 to 255 */
/* 3) unsigned int len - length of the array to be written to the EEPROM      */
/* Writes <len> data bytes to the EEPROM slave specified by the <EEPROM_ADDR> */
/* constant.                                                                 */
/*---------------------------------------------------------------------------*/
void EEPROM_WriteArray(unsigned int dest_addr, unsigned char* src_addr, unsigned int len)
{
   unsigned char* pData = (unsigned char*) src_addr;
   unsigned int WCount;

   I2C_TARGET   = EEPROM_ADDR;
   I2C_Sendbuf[0]   = (dest_addr&0xFF);
   I2C_DataTxLen=1;

   for(WCount=0 ; WCount<len; WCount++)
   {
      I2C_Sendbuf[I2C_DataTxLen++]=*pData;
      pData++;
   }
   I2C_DataTxIndex=0;
   I2C_EndFlag=0;
   I2C_RW = I2C_WRITE;

   I2C0_Ctrl(1,0,0,0);   // I2C as master sends START signal
   while(I2C_EndFlag==0);

}

/*---------------------------------------------------------------------------*/
/* EEPROM_ReadArray ()                                                       */
/*---------------------------------------------------------------------------*/
/* Return Value : None                                                       */
/* Parameters     :                                                          */
/*     1) unsigned char* dest_addr - pointer to the array that will be filled */
/*                                   with the data from the EEPROM           */
/*                                range is full range of character: 0 to 255 */
/* 2)unsigned int src_addr-beginning address to I2C_READ data from the EEPROM */
/* 3)unsigned int len - length of the array to be I2C_READ from the EEPROM    */
/* Reads up to 256 data bytes from the EEPROM slave specified by the         */
/* <EEPROM_ADDR> constant.                                                   */
/*---------------------------------------------------------------------------*/
void EEPROM_ReadArray (unsigned char* dest_addr, unsigned int src_addr,
                       unsigned int len)
{
   unsigned char* pData = (unsigned char*) dest_addr;
   unsigned int RCount;

   I2C_TARGET   = EEPROM_ADDR;   //Device_ID_Addr=0XA0

   I2C_Sendbuf[0]   = (src_addr&0xFF);   //24Cxx Memory Address
   I2C_DataTxLen=1;
   I2C_DataTxIndex=0;
```

```
    I2C_DataRxLen=len;
    I2C_DataRxIndex=0;
    I2C_EndFlag=0;
    I2C_RW = I2C_READ;

    I2C0_Ctrl(1,0,0,0);    // I2C as master sends START signal
    while(I2C_EndFlag==0);

    for(RCount=0 ; RCount < len ; RCount++ )
    {
       *pData=I2C_Recbuf[RCount];
       pData++;
    }
}
/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
    unsigned char t;
    for(;ms>0;ms--)              //@HAO=1.843MHz
       for(t=114;t>0;t--)
       { __asm__("NOP"); }
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

15. 通訊 IP(I2C2)

15.1. 範例名稱

I2C2_Master

15.2. 範例說明

(6) HY17M26 工作在 I2C2 Master Mode, 對 I2C EEPROM 24C02 做寫入與讀取控制範例.

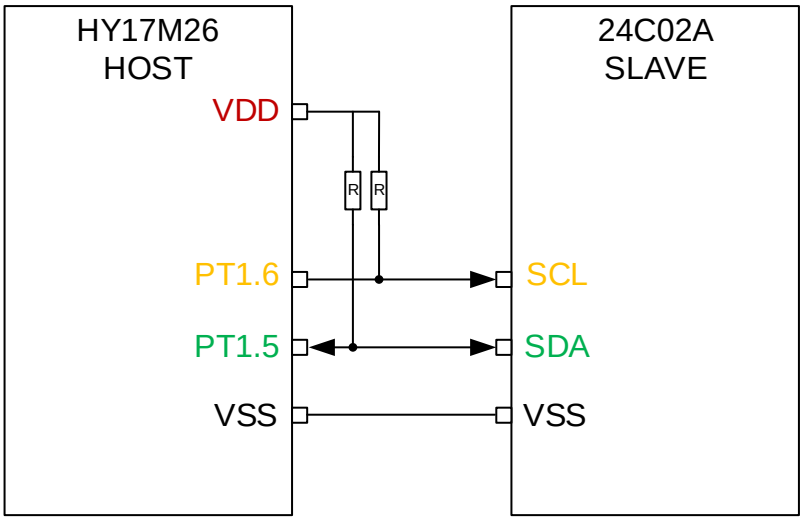
(7) 範例程式內容包含 HY17M26 之 I2C2 Master 設置初始化過程，並且對一個 I2C EEPROM(24C02)裝置做單次的資料寫入和讀取與連續的資料寫入讀取範例

(8) 程式第一段功能為做單次性的資料寫入和讀取在 EEPROM 裝置，首先由 I2C2 Master 寫入資料 0xAA 在 EEPROM 的 WORD ADDRESS 0x00，然後再下 Read 指令讀回 EEPROM 的 WORD ADDRESS 0x00 位置。

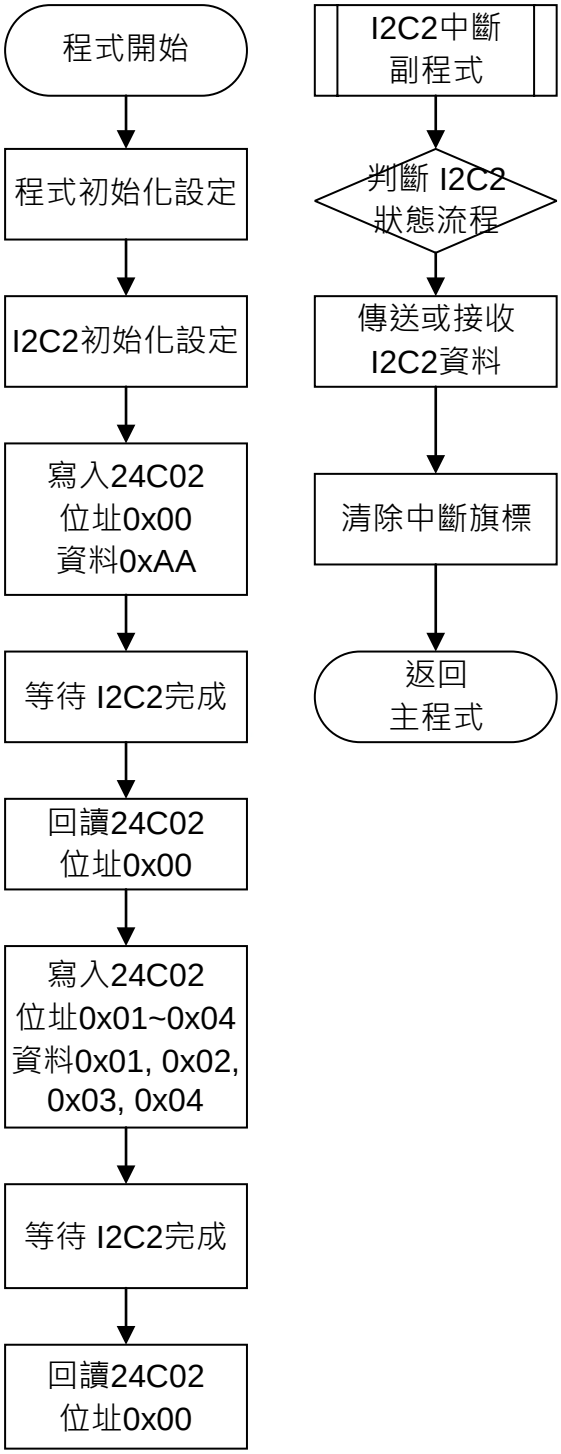
(9) 程式第二段功能為做連續性的資料寫入和讀取在 EEPROM 裝置，從 EEPROM 的 WORD ADDRESS 0x01 開始寫入 4bytes 資料直到 WORD ADDRESS 0x04，再依序從 EEPROM 的 WORD ADDRESS 0x00 連續讀取，分別為 2、5、6 筆資料，觀察資料是否已經正確被寫入。

(10) 在本文範例程式裡面， 當執行完程式第一段和第二段的正確的執行結果為 EEPROM Address 0x00 資料等於 0xAA，Address 0x01 資料等於 0x02，Address 0x02 資料等於 0x03，Address 0x03 資料等於 0x04，Address 0x04 資料等於 0x05。

15.3. 範例配置



15.4. 軟體流程


程式開始
程式初始化設定
I2C2初始化設定
寫入24C02
位址0x00
資料0xAA
等待 I2C2完成
回讀24C02
位址0x00
寫入24C02
位址0x01~0x04
資料0x01, 0x02,
0x03, 0x04
等待 I2C2完成
回讀24C02
位址0x00
I2C2中斷
副程式
判斷 I2C2
狀態流程
傳送或接收
I2C2資料
清除中斷旗標
返回
主程式

15.5. 程式碼

說明 ： 在此僅貼上 I2C2_Master.C 程式內容做參考.

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * I2C2_Master_Demo.c                                                          *
 * ---------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                             *
 * http://www.hycontek.com/                                                   *
 *                                                                             *
 * Program Description:                                                       *
 *                                                                             *
 *           HY17M26                        24C02A                            *
 *---------------------              ------------------                      *
 *                            |            |                                  *
 *                            |            |                                  *
 *                       VDD |----->    | VDD                                *
 *                     PT1.6 |----->    | SCL                               *
 *                     PT1.5 |<---->    | SDA                               *
 *                       VSS |----->    | VSS                               *
 *                            |            |                                  *
 *                            |            |                                  *
 *---------------------              ----------------                       *
 *                                                                             *
 * EEPROM Address                                                              *
 * addr[0x00]=0xaa                                                             *
 * addr[0x01]=0x02                                                             *
 * addr[0x02]=0x03                                                             *
 * addr[0x03]=0x04                                                             *
 * addr[0x04]=0x05                                                             *
 *                                                                             *
 * For External Input                                                         *
 * IC Body: HY17M26                                                            *
 * Project Name : I2C2_Master                                                 *
 * Created Date : 2021/6/4 BY Cyril                                           *
 *                                                                             *
 *******************************************************************************/
/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <INT.h>
#include <CLK.h>
#include <RST.h>
#include <I2C.h>
#include <GPIO.h>
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
```

```
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

//I2C PORT
//#define SCL2_PT12
#define SCL2_PT16
//#define SCL2_PT23
//#define SCL2_PT27

#define EEPROM_ADDR       0xA0    // Device address for slave target

#define I2CBufferSize    10
#define I2C_WRITE    0x00           // I2C WRITE command
#define I2C_READ     0x01           // I2C READ command
#define I2C_Delay     100000    //@2MHz, Delay 5s
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void I2CInit(void);
void CLOCKInit(void);
void EEPROM_ByteWrite(unsigned int addr, unsigned char dat);
unsigned char EEPROM_ByteRead(unsigned int addr);
void EEPROM_WriteArray(unsigned int dest_addr, unsigned char* src_addr,
                       unsigned int len);
void EEPROM_WriteArray1(unsigned int dest_addr, unsigned char* src_addr,
                       unsigned int len);
void EEPROM_ReadArray(unsigned char* dest_addr, unsigned int src_addr,
                      unsigned int len);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
unsigned char I2C_Read_Buffer[I2CBufferSize];
unsigned char I2C_RW;
unsigned char I2C_TARGET;          // Target I2C slave address
unsigned char I2C_Sendbuf[I2CBufferSize];
unsigned char I2C_Recbuf[I2CBufferSize];
unsigned char I2C_EndFlag;
unsigned int I2C_DataTxLen,I2C_DataTxIndex,I2C_DataRxLen,I2C_DataRxIndex;
unsigned long I2C_Timeout;
unsigned char EEPROM_WriteData[4] = {0x02,0x03,0x04,0x05};
unsigned char i,temp;
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
    CLOCKInit();
    I2CInit();

      for(i=0;i<I2CBufferSize;i++)
```

```
    {
        I2C_Read_Buffer[i]=0x00;      //CLR I2C_Read_Buffer[]
    }

    GIE_Enable();

    // I2C single I2C_WRITE & I2C_READ
    EEPROM_ByteWrite(0x00,0xAA);                    //Single Byte I2C_WRITE word address 0x00, and I2C_WRITE
data 0x01
    Delay(100);                                     //Delay for EEPROM 24C02 I2C_WRITE Cycle Time
    I2C_Read_Buffer[0]=EEPROM_ByteRead(0x00);   //Single Byte I2C_READ word address 0x00, the I2C_READ
value is 0x01

    // I2C sequential I2C_WRITE & I2C_READ
    EEPROM_WriteArray(0x01,EEPROM_WriteData,4); //I2C_WRITE array data to the word address from 0x01 to
0x04
    Delay(100);                                     //Delay for EEPROM 24C02 I2C_WRITE Cycle Time
    EEPROM_ReadArray(I2C_Read_Buffer, 0x00, 2); //sequential I2C_READ data from 0x00 to 0x01
    EEPROM_ReadArray(I2C_Read_Buffer, 0x00, 5); //sequential I2C_READ data from 0x00 to 0x04
    EEPROM_ReadArray(I2C_Read_Buffer, 0x00, 6); //sequential I2C_READ data from 0x00 to 0x05

    while(1);
}
/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{
    if(I2C2IF_IsFlag() &&I2C_EndFlag==0)   //Get I2C Interrupt Flag
    {
    switch(I2C2_STAState())
    {
        case 0x90:   //MACTFlag+RWFlag
                     //START has been transmitted
            I2C2_SendData(I2C_TARGET);   //Send Slave Address & R/W Bit
            I2C2_Ctrl(0,0,0,0);              //Clear all I2C flag
            break;

        case 0x84:   //MACTFlag+ACKFlag
                     //Slave A + W has been transmitted. ACK has been received.
            I2C2_SendData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            I2C2_Ctrl(0,0,0,0);         //Clear all I2C flag
            break;

        case 0x80: //MACTFlag
                 //Slave A + W has been transmitted. ACK has been received.
                 I2C2_SendData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
            I2C2_Ctrl(0,0,0,0);         //Clear all I2C flag
            break;

        case 0x8C:   //MACTFlag+DFFlag+ACKFlag
                 //DATA has been transmitted and ACK has been received
            if(I2C_DataTxIndex<I2C_DataTxLen)
            {
              I2C2_SendData(I2C_Sendbuf[I2C_DataTxIndex++]);   //Send Data to Slave
              I2C2_Ctrl(0,0,0,0);       // Clear all I2C flag
            }
```

```
            else
            {
                if(I2C_RW == I2C_WRITE)
                {
                   I2C2_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
                   I2C_Timeout=I2C_Delay;    //HAO=2MHz, wait 5s
                   while(I2C_Timeout != 0)
                   {
                       I2C_Timeout--;
                     if(STA2== 0x30)         //A STOP has been transmitted
                     {
                        I2C_EndFlag=1;
                        break;
                      }
                   }
                }
                else if(I2C_RW == I2C_READ)
                   I2C2_Ctrl(1,0,0,0);    //I2C as master sends START signal
                   I2C_DataTxIndex=0;
            }
            break;

case 0x88:    //MACTFlag+DFFlag
                  //DATA has been transmitted and NACK has been received
            I2C2_Ctrl(0,1,0,0);       //I2C as master sends STOP signal
            I2C_DataTxIndex=0;
          I2C_Timeout=I2C_Delay;    //HAO=2MHz, wait 5s
          while(I2C_Timeout != 0)
          {
              I2C_Timeout--;
             if(STA2== 0x30)         //A STOP has been transmitted
             {
                I2C_EndFlag=1;
                break;
              }
          }
            break;

case 0xB0:
                  //A repeated START has been transmitted.
            I2C2_SendData(I2C_TARGET | I2C_READ);    //Send Slave Address & R/W Bit
            I2C2_Ctrl(0,0,0,0);       //Clear all I2C flag
            break;

case 0x94:    //MACTFlag+RWFlag+ACKFlag
                  //Slave A + R has been transmitted. ACK has been received.
            if(I2C_DataRxLen>1)
            {
               I2C2_Ctrl(0,0,0,1);    //Set ACK bit
            }
          else
          {
              I2C2_Ctrl(0,0,0,0);    //Clear all I2C flag
            }
            break;

case 0x9C:    //MACTFlag+RWFlag+DFFlag+ACKFlag
```

```
                    //Data byte has been received. ACK has been transmitted.
                  I2C2_ReceiveDATA(I2C_Recbuf[I2C_DataRxIndex++]);
                  if((I2C_DataRxLen-1)>I2C_DataRxIndex)
                  {
                     I2C2_Ctrl(0,0,0,1);    //Set ACK bit
                  }
                  else
                  {
                     I2C2_Ctrl(0,0,0,0);    //Clear all I2C flag
                  }
                  break;

            case 0x98:    //MACTFlag+RWFlag+DFFlag
                    //Data byte has been received. NACK has been transmitted.
                  I2C2_ReceiveDATA(I2C_Recbuf[I2C_DataRxIndex++]);
                  I2C2_Ctrl(0,1,0,0);    //I2C as master sends STOP signal
                 I2C_Timeout=I2C_Delay;    //HAO=2MHz, wait 5s
                 while(I2C_Timeout != 0)
                 {
                     I2C_Timeout--;
                   if(STA2== 0x30)        //A STOP has been transmitted
                   {
                      I2C_EndFlag=1;
                      break;
                    }
                 }
                 break;

            default:
                    I2C2_Ctrl(0,0,0,0);    //Clear all I2C flag
                    I2C_EndFlag=1;
                    break;
      }
            I2C2_I2CINT_CLEAR();              //CLR I2C INT Flag
            I2C2_I2CER_CLEAR();         //CLR I2C error Flag
            I2C2IF_ClearFlag();                     //CLR I2CIF
      }

      if(I2CERIF_IsFlag())    //Get I2C Error Interrupt Flag
      {
          I2C_EndFlag=1;
            I2C2_I2CINT_CLEAR();              //CLR I2C INT Flag
            I2C2_I2CER_CLEAR();         //CLR I2C error Flag
            I2CER2IF_ClearFlag();         //CLR I2CERIF
          I2C2_Ctrl(0,0,0,0);              //Clear all I2C flag
      }

}
/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
```

```
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* I2C Init Function                                                         */
/*---------------------------------------------------------------------------*/
void I2CInit(void)
{
   //GPIO Init
#if defined(SCL2_PT12)
   GPIO_GSCL2Set(GSCL2_PT12);
#endif
#if defined(SCL2_PT16)
   GPIO_GSCL2Set(GSCL2_PT16);
#endif
#if defined(SCL2_PT23)
   GPIO_GSCL2Set(GSCL2_PT23);
#endif
#if defined(SCL2_PT27)
   GPIO_GSCL2Set(GSCL2_PT27);
#endif

#if defined(HAO_2MHZ)
   I2C2_Open(4);    //Default CPU clock is 2MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 2000000/[4*(4+1)]=100kHz
#endif
#if defined(HAO_4MHZ)
   I2C2_Open(9);    //Default CPU clock is 1MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 4000000/[4*(9+1)]=100kHz
#endif
#if defined(HAO_8MHZ)
   I2C2_Open(19);    //Default CPU clock is 8MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]= 8000000/[4*(19+1)]=100kHz
#endif
#if defined(HAO_17MHZ)
   I2C2_Open(39);    //Default CPU clock is 16MHz, Data Baud Rate : (I2CLK)/[4x(CRG+1)]=
16000000/[4*(39+1)]=100kHz
#endif

}

/*---------------------------------------------------------------------------*/
```

```
/* EEPROM_ByteWrite ()                                                    */
/*                                                                        */
/* Return Value : None                                                    */
/* Parameters     :                                                       */
/* 1) unsigned int addr - address to I2C_WRITE in the EEPROM              */
/*                                                                        */
/* 2) unsigned char dat-data to I2C_WRITE to the address <addr> in the EEPROM */
/*                              range is full range of character: 0 to 255 */
/*                                                                        */
/* This function writes the value in <dat> to location <addr> in the EEPROM */
/* then polls the EEPROM until the I2C_WRITE is complete.                 */
/*                                                                        */
/*------------------------------------------------------------------------*/
void EEPROM_ByteWrite(unsigned int addr, unsigned char dat)
{
   I2C_TARGET    = EEPROM_ADDR;     //Device_ID_Addr=0XA0

   I2C_Sendbuf[0]   = (addr&0xFF);   //24Cxx Memory Address
   I2C_Sendbuf[1]   = dat;           //24Cxx Memory Data
   I2C_DataTxLen=2;
   I2C_DataTxIndex=0;
   I2C_EndFlag=0;
   I2C_RW = I2C_WRITE;

   I2C2_Ctrl(1,0,0,0);   // I2C as master sends START signal
   while(I2C_EndFlag==0);

}

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

   I2C_TARGET    = EEPROM_ADDR;   //Device_ID_Addr=0XA0

   I2C_Sendbuf[0]   = (addr&0xFF);   //24Cxx Memory Address
   I2C_DataTxLen=1;
   I2C_DataTxIndex=0;
   I2C_DataRxLen=1;
   I2C_DataRxIndex=0;
   I2C_EndFlag=0;
   I2C_RW = I2C_READ;

   I2C2_Ctrl(1,0,0,0);   // I2C as master sends START signal
   while(I2C_EndFlag==0);

   retval=I2C_Recbuf[0];
   return retval;
}

/*------------------------------------------------------------------------*/
/* EEPROM_WriteArray()                                                    */
/*------------------------------------------------------------------------*/
/* Return Value : None                                                    */
```

```
/* Parameters     :                                                              */
/* 1) unsigned int dest_addr-beginning address to I2C_WRITE to in the EEPROM    */
/* 2) unsigned char* src_addr - pointer to the array of data to be written       */
/*                                  range is full range of character: 0 to 255   */
/* 3) unsigned int len - length of the array to be written to the EEPROM         */
/* Writes <len> data bytes to the EEPROM slave specified by the <EEPROM_ADDR> */
/* constant.                                                                     */
/*-------------------------------------------------------------------------*/
void EEPROM_WriteArray(unsigned int dest_addr, unsigned char* src_addr, unsigned int len)
{
   unsigned char* pData = (unsigned char*) src_addr;
   unsigned int WCount;

   I2C_TARGET    = EEPROM_ADDR;
   I2C_Sendbuf[0]    = (dest_addr&0xFF);
   I2C_DataTxLen=1;

   for(WCount=0 ; WCount<len; WCount++)
   {
      I2C_Sendbuf[I2C_DataTxLen++]=*pData;
      pData++;
   }
   I2C_DataTxIndex=0;
   I2C_EndFlag=0;
   I2C_RW = I2C_WRITE;

   I2C2_Ctrl(1,0,0,0);    // I2C as master sends START signal
   while(I2C_EndFlag==0);

}

/*-------------------------------------------------------------------------*/
/* EEPROM_ReadArray ()                                                       */
/*-------------------------------------------------------------------------*/
/* Return Value : None                                                     */
/* Parameters     :                                                          */
/*     1) unsigned char* dest_addr - pointer to the array that will be filled     */
/*                                   with the data from the EEPROM                */
/*                                  range is full range of character: 0 to 255   */
/* 2)unsigned int src_addr-beginning address to I2C_READ data from the EEPROM */
/* 3)unsigned int len - length of the array to be I2C_READ from the EEPROM       */
/* Reads up to 256 data bytes from the EEPROM slave specified by the             */
/* <EEPROM_ADDR> constant.                                                       */
/*-------------------------------------------------------------------------*/
void EEPROM_ReadArray (unsigned char* dest_addr, unsigned int src_addr,
                       unsigned int len)
{
   unsigned char* pData = (unsigned char*) dest_addr;
   unsigned int RCount;

   I2C_TARGET    = EEPROM_ADDR;    //Device_ID_Addr=0XA0

   I2C_Sendbuf[0]    = (src_addr&0xFF);    //24Cxx Memory Address
   I2C_DataTxLen=1;
   I2C_DataTxIndex=0;
   I2C_DataRxLen=len;
   I2C_DataRxIndex=0;
```

```
    I2C_EndFlag=0;
    I2C_RW = I2C_READ;

    I2C2_Ctrl(1,0,0,0);    // I2C as master sends START signal
    while(I2C_EndFlag==0);

    for(RCount=0 ; RCount < len ; RCount++ )
    {
       *pData=I2C_Recbuf[RCount];
       pData++;
    }
}
/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
    unsigned char t;
    for(;ms>0;ms--)             //@HAO=1.843MHz
       for(t=114;t>0;t--)
       { __asm__("NOP"); }
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

16. 通訊 IP(SPI)

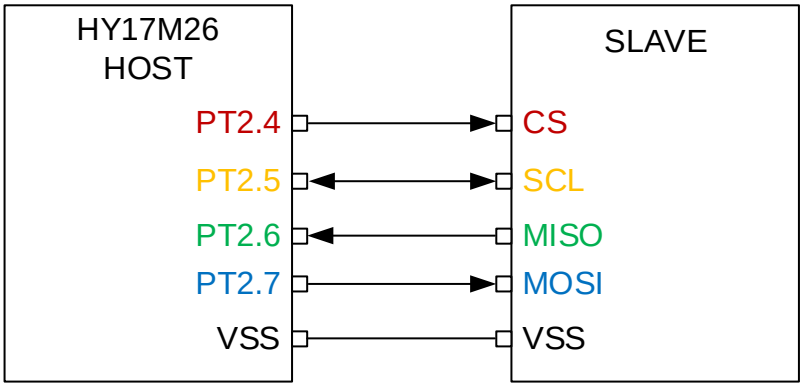
16.1. 範例名稱

SPI_Master

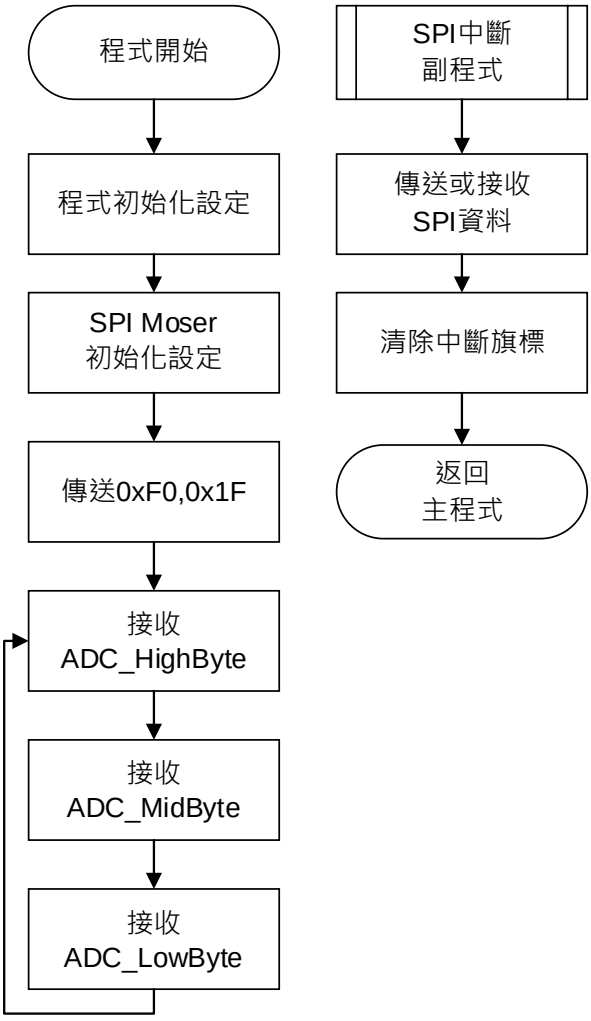
16.2. 範例說明

(1) 範例程式內容包含 HY17M26 之 SPI Master 設置初始化過程，並且對 ADC Device 做資料發送和接收範例
(2) 設置系統工作頻率
(3) SPI Master 初始化設置。
(4) 程式第一段由 HY17M26 發送開始信號 0xF0,0x1F。
(5) 程式第二段當 ADC Device 接收到開始信號後，會不斷發送 ADC 資料。此時 HY17M26 依序接收 ADC_HighByte,ADC_MidByte,ADC_LowByte。

16.3. 範例配置



16.4. 軟體流程



16.5. 程式碼

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * SPI_Host_Demo.c                                                             *
 * --------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                             *
 * http://www.hycontek.com/                                                    *
 *                                                                             *
 * Program Description:                                                        *
 *                                                                             *
 *            HY17M26(host)                  slave                             *
 *---------------------              ----------------                         *
 *                            |            |                                   *
 *                            |            |                                   *
 *                       VDD | -----> | VDD                                   *
 *                     PT2.4 | -----> | CS                                    *
 *                     PT2.5 | -----> | SCL                                   *
 *                     PT2.6 | <----- | MISO                                  *
 *                     PT2.7 | -----> | MOSI                                  *
 *                       VSS | -----> | VSS                                   *
 *                            |            |                                   *
 *                            |            |                                   *
 *---------------------              ----------------                         *
 *                                                                             *
 *                                                                             *
 * For External Input                                                          *
 * IC Body: HY17M26                                                            *
 * Project Name : SPI_Host                                                     *
 * Created Date : 2021/6/4 By Cyril                                            *
 *                                                                             *
 *******************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <GPIO.h>
#include <CLK.h>
#include <SPI.h>
#include <RST.h>
#include <INT.h>
/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif

//SPI PORT
//#define SPI_PT11       //PT10 CS, PT11 CK, PT12 MISO, PT13 MOSI
#define SPI_PT25    //PT24 CS, PT25 CK, PT26 MISO, PT27 MOSI
//#define SPI_PT31    //PT30 CS, PT31 CK, PT32 MISO, PT33 MOSI
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
```

```
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void CLOCKInit(void);
void GPIOInit(void);
void SPIInit(void);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
volatile typedef union _Flag
{
  unsigned int    _byte;
    struct
    {
       unsigned b_ADCdone:1;
       unsigned b_SPIDone:1;
       unsigned Reserved:14;
    };
} Flag;
Flag     Flagbits;
unsigned int SPI_READ_Buffer[8];
unsigned int SPI_write_buffer[8];
unsigned int SPI_Test_counter;
unsigned char ADC_Start;
signed long    ADCData;
signed long    ADC_LowByte;
signed long ADC_MiddleByte;
signed long ADC_HighByte;
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
    //HY17M26 SPI Host sends 0xF0, 0x1F. SPI Slave returns 0x12, 0x34 and set to transmit ADC Data at this time.
    //Next, the SPI Host sends 24 bits (8bit+8bit+8bit) at will.
    //at this time the SPI Slave will return ADC_HighByte, ADC_MiddleByte, ADC_LowByte in sequence

    PT10_HWResetEnable();

    CLOCKInit();
    GPIOInit();
    SPIInit();

    SPI_write_buffer[0]=0xF0;
    SPI_write_buffer[1]=0x1F;
    SPI_write_buffer[2]=0xA0;
    SPI_write_buffer[3]=0xA1;
    SPI_write_buffer[4]=0xA2;
    SPI_Test_counter=0;
    Flagbits.b_SPIDone= 0;
    ADC_Start= 0;
    ADCData=0;
    ADC_LowByte= 0;
    ADC_MiddleByte= 0;
    ADC_HighByte= 0;

    //Delay(100);

    GIE_Enable();


    GPIO_PT2OutputLow(PT24); //PT2.4 output low
    Delay(20);          //have to add, this delay time is wait for HY17M26 SPI slave to prepare the data that needs to be
transmitted
    SSPBUF0 = SPI_write_buffer[0];
    while(Flagbits.b_SPIDone==1);
```

```
    SPI_READ_Buffer[0]=SSPBUF0;
    GPIO_PT2OutputHigh(PT24); //PT2.4 output high
    Flagbits.b_SPIDone= 0;
    Delay(200); //have to add, otherwise, write to HY17M26 SPI slave fail

    GPIO_PT2OutputLow(PT24); //PT2.4 output low
    Delay(20);          //have to add, this delay time is wait for HY17M26 SPI slave to prepare the data that needs to be
transmitted
    SSPBUF0 = SPI_write_buffer[1];
    while(Flagbits.b_SPIDone==1);
    SPI_READ_Buffer[1]=SSPBUF0;
    GPIO_PT2OutputHigh(PT24); //PT2.4 output high
    Flagbits.b_SPIDone= 0;
    Delay(200); //have to add, otherwise, write to HY17M26 SPI slave fail

    while(1)
    {
      if(GPIO_PT2GET(PT20)==0)
      {
        Delay(2000);
        while(GPIO_PT2GET(PT20)==0);

        GPIO_PT2OutputLow(PT24); //PT2.4 output low
        Delay(20);          //have to add, this delay time is wait for HY17M26 SPI slave to prepare the data that needs to be
transmitted
        SSPBUF0 = SPI_write_buffer[2];
        while(Flagbits.b_SPIDone==1);
        SPI_READ_Buffer[2]=SSPBUF0;
        ADC_HighByte=SPI_READ_Buffer[2];
        GPIO_PT2OutputHigh(PT24); //PT2.4 output high
        Flagbits.b_SPIDone= 0;
        Delay(200); //have to add, otherwise, write to HY17M26 SPI slave fail

        GPIO_PT2OutputLow(PT24); //PT2.4 output low
        Delay(20);          //have to add, this delay time is wait for HY17M26 SPI slave to prepare the data that needs to be
transmitted
        SSPBUF0 = SPI_write_buffer[3];
        while(Flagbits.b_SPIDone==1);
        SPI_READ_Buffer[3]=SSPBUF0;
        ADC_MiddleByte=SPI_READ_Buffer[3];
        GPIO_PT2OutputHigh(PT24); //PT2.4 output high
        Flagbits.b_SPIDone= 0;
        Delay(200); //have to add, otherwise, write to HY17M26 SPI slave fail

        GPIO_PT2OutputLow(PT24); //PT2.4 output low
        Delay(20);          //have to add, this delay time is wait for HY17M26 SPI slave to prepare the data that needs to be
transmitted
        SSPBUF0 = SPI_write_buffer[4];
        while(Flagbits.b_SPIDone==1);
        SPI_READ_Buffer[4]=SSPBUF0;
        ADC_LowByte=SPI_READ_Buffer[4];
        GPIO_PT2OutputHigh(PT24); //PT2.4 output high
        Flagbits.b_SPIDone= 0;
        Delay(200); //have to add, otherwise, write to HY17M26 SPI slave fail

        ADCData=(ADC_HighByte<<24)+(ADC_MiddleByte<<16)+(ADC_LowByte<<8);    //first, get temp 32bit ADC
        //ADCData=ADCData>>8;    //second, get 24bit ADC
      }

    }

}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
```

```
void ISR(void) __interrupt
{
   NOP();
   //SPI Event
   if(SPIIF_IsFlag())
   {
      SPI_Test_counter++;

      SPIIF_ClearFlag();     //Clear SPIIF
      Flagbits.b_SPIDone= 1;
   }

}

/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
        for(;num>0;num--)
      __asm__("NOP");
}


void Delayms(unsigned int ms)
{
   unsigned char t;
   for(;ms>0;ms--)              //@HAO=1.843MHz
      for(t=114;t>0;t--)
      { __asm__("NOP"); }
}

/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif
}

/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void GPIOInit(void)
{
```

```
    //PT2.0 Input mode
    GPIO_PT2InputMode(PT20);
    GPIO_PT2DigitalEnable(PT20);

}

/*---------------------------------------------------------------------------*/
/* SPI Init Function                                                         */
/*---------------------------------------------------------------------------*/
void SPIInit(void)
{

#if defined(SPI_PT11)
    GPIO_GSPISet(GSPI_PT11);
    GPIO_PT1OutputMode(PT10); //PT1.0 output mode
    GPIO_PT1DigitalEnable(PT10);
    GPIO_PT1OutputHigh(PT10); //PT1.0 output high
#endif
#if defined(SPI_PT25)
    GPIO_GSPISet(GSPI_PT25);
    GPIO_PT2OutputMode(PT24); //PT2.4 output mode
    GPIO_PT2DigitalEnable(PT24);
    GPIO_PT2OutputHigh(PT24); //PT2.4 output high
#endif
#if defined(SPI_PT31)
    GPIO_GSPISet(GSPI_PT31);
    GPIO_PT3OutputMode(PT30); //PT3.0 output mode
    GPIO_PT3DigitalEnable(PT30);
    GPIO_PT3OutputHigh(PT30); //PT3.0 output high
#endif

    SPI_Open(SSPM_MCPUCK,CKP_HI,CKE_IDLE);    //master, 3-wire, SPI_CK=CPU_CK
                                              //CKP=1b
                                              //CKE=1b


}
/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

17. 其他 IP(Power)

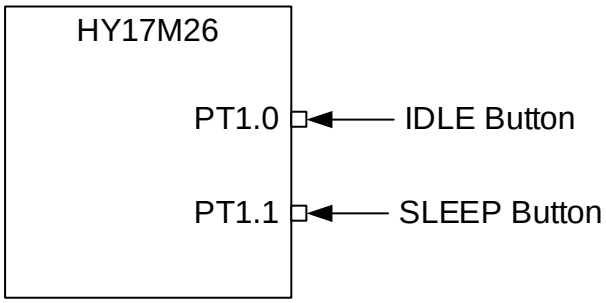
17.1. 範例名稱

Power

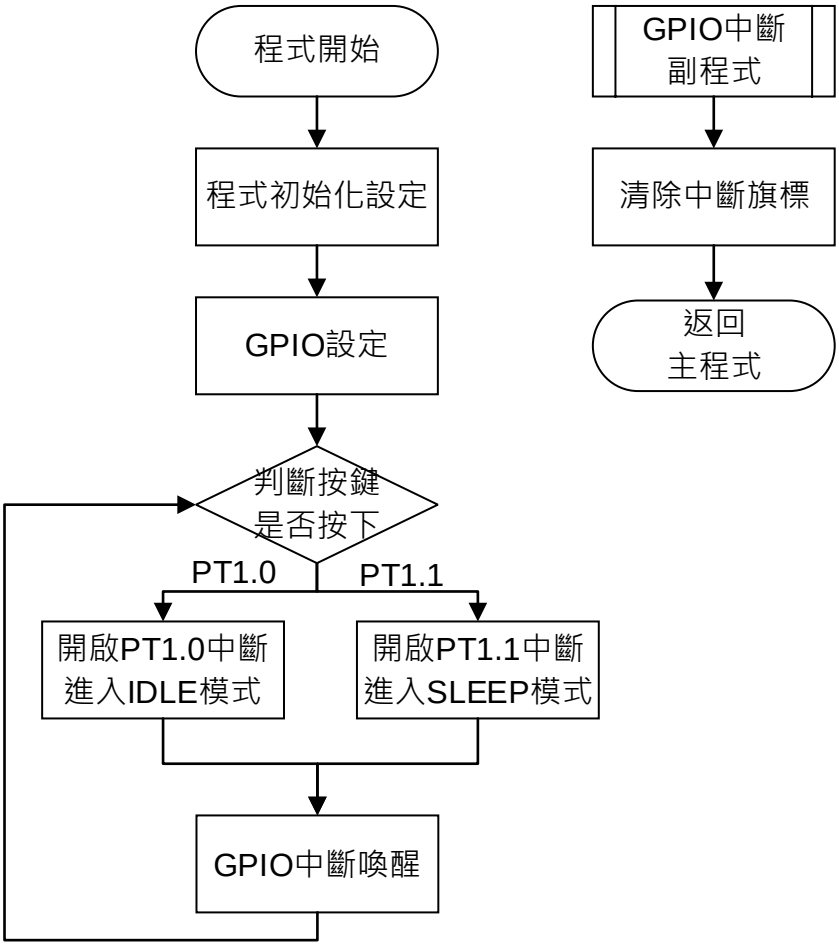
17.2. 範例說明

(1) HY17M26 進入 Sleep 與 Idle Mode 範例程式
(2) 設置系統工作頻率。
(3) 關閉 BOR2 與 VDDA 電壓
(4) 電後按下 PT1.1 進入 Sleep Mode，再按下 PT1.1 喚醒。
(5) 電後按下 PT1.0 進入 Idle Mode，再按下 PT1.0 喚醒。

17.3. 範例配置



17.4. 軟體流程


程式開始
程式初始化設定
GPIO設定
判斷按鍵
是否按下
PT1.0
PT1.1
開啟PT1.0中斷
進入IDLE模式
開啟PT1.1中斷
進入SLEEP模式
GPIO中斷喚醒
GPIO中斷
副程式
清除中斷旗標
返回
主程式

17.5. 程式碼

```
#define   USE_HY17M26_2M
/*******************************************************************************
 * POWER_Demo.c                                                                *
 * ---------------------------------------------------------------------------- *
 * Copyright 2021 HYCON Technology                                              *
 * http://www.hycontek.com/                                                     *
 *                                                                              *
 * Program Description:                                                         *
 *                                                                              *
 *            HY17M26                                                           *
 *--------------------                                                          *
 *                          |                                                   *
 *                    PT1.0|<-button (IDLE)                                    *
 *                          |                                                   *
 *                    PT1.1|<-button (SLEEP)                                   *
 *                          |                                                   *
 *--------------------                                                          *
 *                                                                              *
 *                                                                              *
 * For External Input                                                           *
 * IC Body: HY17M26                                                             *
 * Project Name : Power                                                         *
 * Created Date : 2021/6/4 BY Cyril                                             *
 *                                                                              *
 *******************************************************************************/

/*---------------------------------------------------------------------------*/
/* Includes                                                                  */
/*---------------------------------------------------------------------------*/
#include <SFRType.h>
#include <PWR.h>
#include <GPIO.h>
#include <RST.h>
#include <INT.h>
#include <CLK.h>

/*---------------------------------------------------------------------------*/
/* DEFINITIONS                                                               */
/*---------------------------------------------------------------------------*/
#ifdef USE_HY17M26_2M
#define HAO_2MHZ
#endif
#ifdef USE_HY17M26_4M
#define HAO_4MHZ
#endif
#ifdef USE_HY17M26_8M
#define HAO_8MHZ
```

```
#endif
#ifdef USE_HY17M26_16M
#define HAO_17MHZ
#endif
/*---------------------------------------------------------------------------*/
/* Function PROTOTYPES                                                       */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num);
void Delayms(unsigned int ms);
void GPIOInit(void);
void CLOCKInit(void);
/*---------------------------------------------------------------------------*/
/* Global CONSTANTS                                                          */
/*---------------------------------------------------------------------------*/
volatile typedef union _Flag
{
   unsigned int    _byte;
   struct
   {
      unsigned b_PT1INT0done:1;
      unsigned b_PT1INT1done:1;
      unsigned b_PT1INT2done:1;
      unsigned b_PT1INT3done:1;
      unsigned b_PT1INT4done:1;
      unsigned b_PT1INT5done:1;
      unsigned b_PT1INT6done:1;
   };
} Flag;
Flag     Flagbits;
unsigned int counterA;
/*---------------------------------------------------------------------------*/
/* Main Function                                                             */
/*---------------------------------------------------------------------------*/
void main(void)
{
   counterA=0;
   CLOCKInit();
   GPIOInit();
   Flagbits.b_PT1INT0done=0;
   Flagbits.b_PT1INT1done=0;

   GIE_Enable();

   while(1)
   {

      if(Flagbits.b_PT1INT0done==1)    //PT1.0
      {
         Flagbits.b_PT1INT0done=0;
         CSFON_Enable();
         ENBOR2_Disable();
         PWR_BGRDisable();
```

```
        CLK_IDLE_HAOSet(IDLEM_HAOAuto);    //Disable HAO. When wake up from GPIO Interrupt, auto set
OSCS[1:0]=00b=OSC_HAO
        Idle();                            //IDLE mode
        counterA++;
      }

      if(Flagbits.b_PT1INT1done==1)    //PT1.1
      {
        Flagbits.b_PT1INT1done=0;
        CSFON_Enable();
        ENBOR2_Disable();
        PWR_BGRDisable();
        CLK_OSCSelect(OSCS_LPO);
        CLK_HAODisable();
        Sleep();                         //Sleep mode
        counterA--;
      }

    }

}

/*---------------------------------------------------------------------------*/
/* Interrupt Service Routines                                                */
/*---------------------------------------------------------------------------*/
void ISR(void) __interrupt
{
   NOP();
   //GPIO Event

   if(E0IF_IsFlag())    //if E0IF=1b
   {
      Flagbits.b_PT1INT0done=1;
      E0IF_ClearFlag();    //PT1.0 E0IF=0b, clear PT1.0 Interrupt Flag
   }

   if(E1IF_IsFlag())    //if E1IF=1b
   {
      Flagbits.b_PT1INT1done=1;
      E1IF_ClearFlag();    //PT1.1 E1IF=0b, clear PT1.1 Interrupt Flag
   }
}

/*---------------------------------------------------------------------------*/
/* CLOCK Init Function                                                       */
/*---------------------------------------------------------------------------*/
void CLOCKInit(void)
{
#if defined(HAO_2MHZ)
   //HAO=1.843MHz
   CLK_CPUCKOpen(HAOM_1843KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif
```

```
#if defined(HAO_4MHZ)
   //HAO=4.147MHz
   CLK_CPUCKOpen(HAOM_4147KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_8MHZ)
   //HAO=8.755MHz
   CLK_CPUCKOpen(HAOM_8755KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

#if defined(HAO_17MHZ)
   //HAO=17.51MHz
   CLK_CPUCKOpen(HAOM_17510KHZ ,OSCS_HAO ,DHS_HSCKDIV1 ,CPUS_HSCK );
#endif

}

/*---------------------------------------------------------------------------*/
/* GPIO Init Function                                                        */
/*---------------------------------------------------------------------------*/
void GPIOInit(void)
{
   //GPIO initial on PT1.0 & PT1.1 (INPUT)
   GPIO_PT1InputMode(PT10);
   GPIO_PT1InputMode(PT11);
   GPIO_PT1DigitalEnable(PT10);
   GPIO_PT1DigitalEnable(PT11);
   GPIO_PT1SETPU(PT10);
   GPIO_PT1SETPU(PT11);
   INTEG0Sel(INTEG0_EDGEFALL);
   INTEG1Sel(INTEG1_EDGEFALL);
   E0IE_Enable();
   E1IE_Enable();
   E0IF_ClearFlag();
   E1IF_ClearFlag();


}

/*---------------------------------------------------------------------------*/
/* Subroutine Function                                                       */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Delay Function                                                            */
/*---------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
       for(;num>0;num--)
     __asm__("NOP");
}
```

```
void Delayms(unsigned int ms)
{
   unsigned char t;
   for(;ms>0;ms--)              //@HAO=1.843MHz
      for(t=114;t>0;t--)
      { __asm__("NOP"); }
}

/*---------------------------------------------------------------------------*/
/* End Of File                                                               */
/*---------------------------------------------------------------------------*/
```

18. 修訂記錄

以下描述本文件差異較大的地方，而標點符號與字形的改變不在此描述範圍。

文件版次	頁次	日期	摘要
V01	All	2021/06/10	初版發行
V02	All	2021/09/16	修正數位電路最低操作電壓 更改 HAOM 參數名稱 HAOM_1843KHZ, HAOM_4147KHZ, HAOM_8755KHZ, HAOM_17510KHZ
V03	8, 9	2023/03/02	1. 電壓操作範圍 1.8V~5.5V 改為 1.9V~5.5V 2. 修改 MTP/EPROM 的燒錄次數 修改前 MTP 燒錄次數 1K 次 修改後 MTP 燒錄次數 100 次 修改前 EPROM 燒錄次數 1K 次 修改後 EPROM 燒錄次數 100 次