HYCON 紘康科技

RTC 時鐘萬年曆應用說明書

HY16F198B

RTC Clock Calendar

目錄

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。

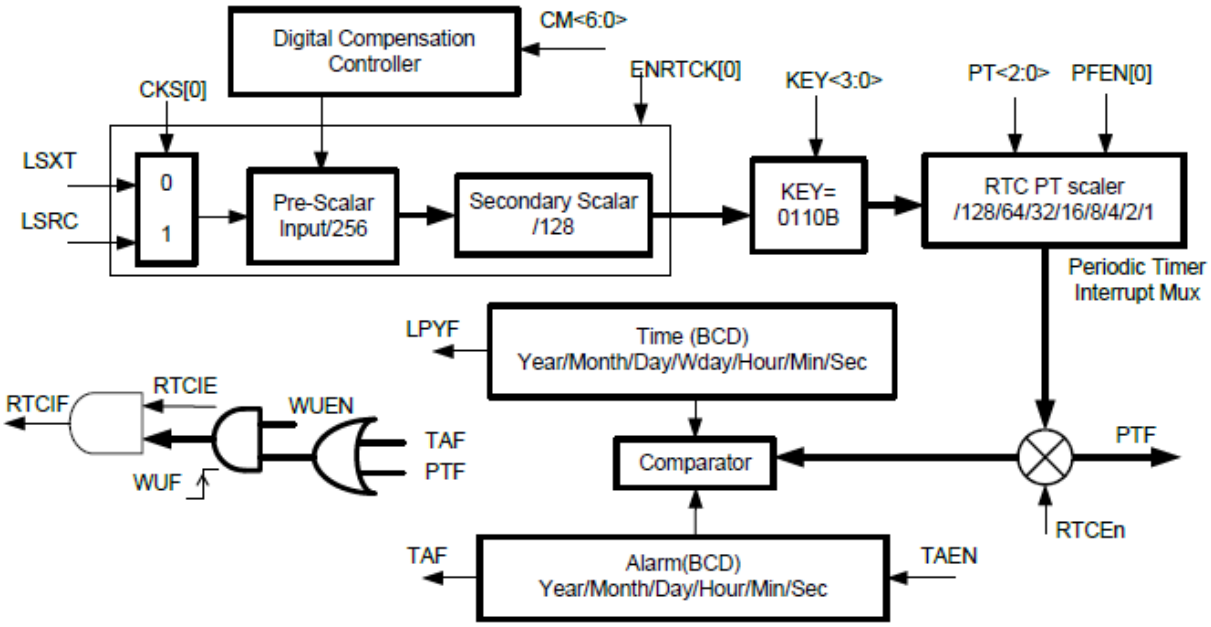
1. 簡介

本文主要是介紹 HYCON HY16F Series 晶片在時鐘及萬年曆的應用。由於 HY16F198B 晶片內部包含 Real Time Clock(RTC)，再配合外部 32768Hz 的振盪器及內部硬體 LCD 驅動. 使 HY16F198B 用於時鐘萬年曆應用,周邊非常簡潔。HY16F198B 相較其他 MCU 能減少電流的消耗.

2. 原理說明

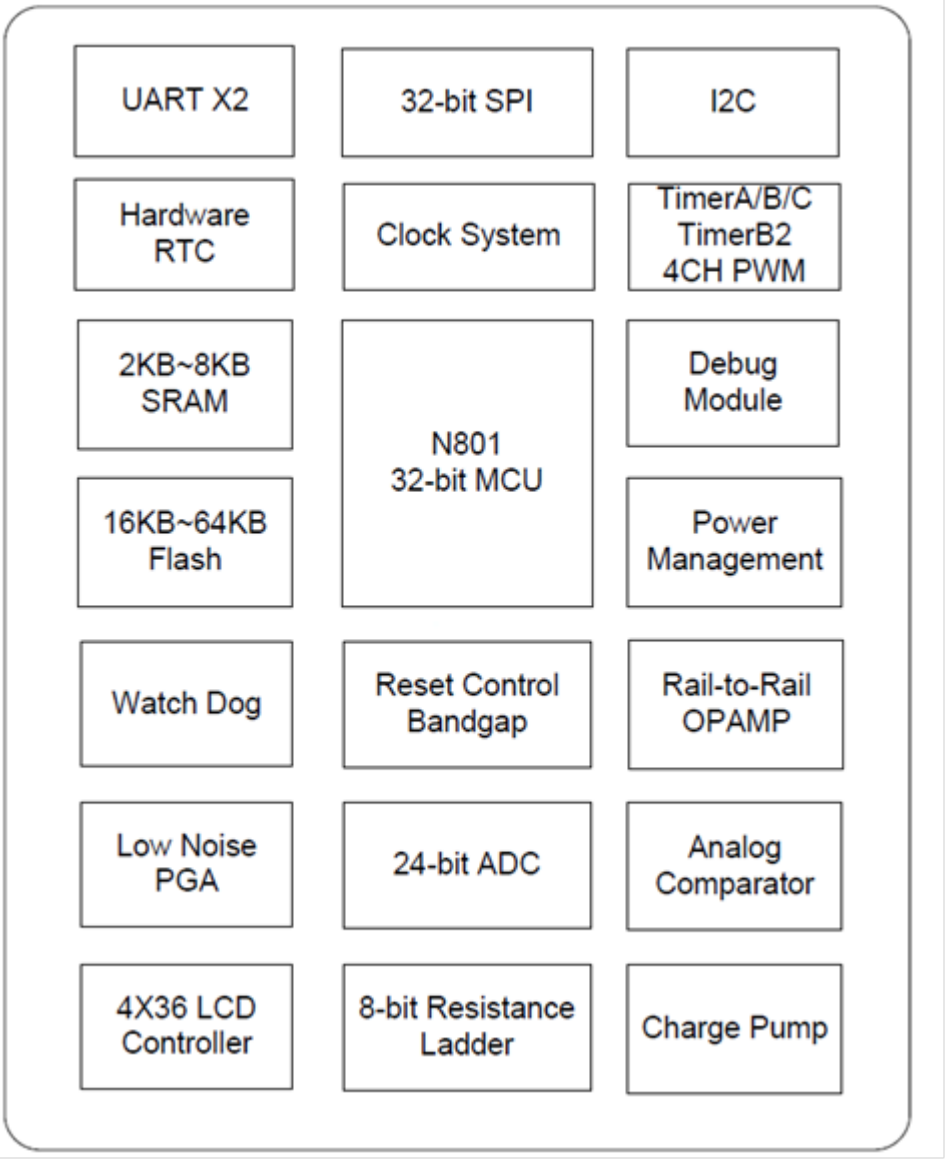
2.1. 控制原理

HY16F198B 應用於時鐘及萬年曆是透過內部 RTC 功能，並結合外部震盪器的應用. RTC 電路選用外部晶體震盪器，震盪頻率為 32768Hz。我們分別進行兩次除頻，第一次除頻為 256、第二次除頻為 128。經過兩次除頻後，可獲得頻率為 1 秒。再將這每秒頻率放置到時間相對應的暫存器內，並判斷是否需要進位。此外選擇 1/128 的週期中斷，以便完成時鐘之碼表功能。下圖為 HY16F198B 內建 RTC 硬體架構:



2.2. 控制晶片

單片機簡介：HY16F 系列 32 位元高性能 Flash 單片機(HY16F198B)



HY16F 系列 32 位元高性能 Flash 單片機(HY16F198B)

特點說明:

(1)採用最新 Andes 32 位元 CPU 核心 N801 處理器。

(2)電壓操作範圍 2.2~3.6V，以及-40℃~85℃工作溫度範圍。

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

(3.1)運行模式 0.6mA@2MHz/2

(3.2)待機模式 5uA@ LSRC=34KHz+IDLE Mode

(3.3)休眠模式 2.5uA

(4)程式記憶體 64KB Flash ROM

(5)資料記憶體 8KB SRAM

(6)擁有 BOR and WDT 功能，可防止 CPU 死機。

(7)24-bit 高精準度ΣΔADC 類比數位轉換器

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

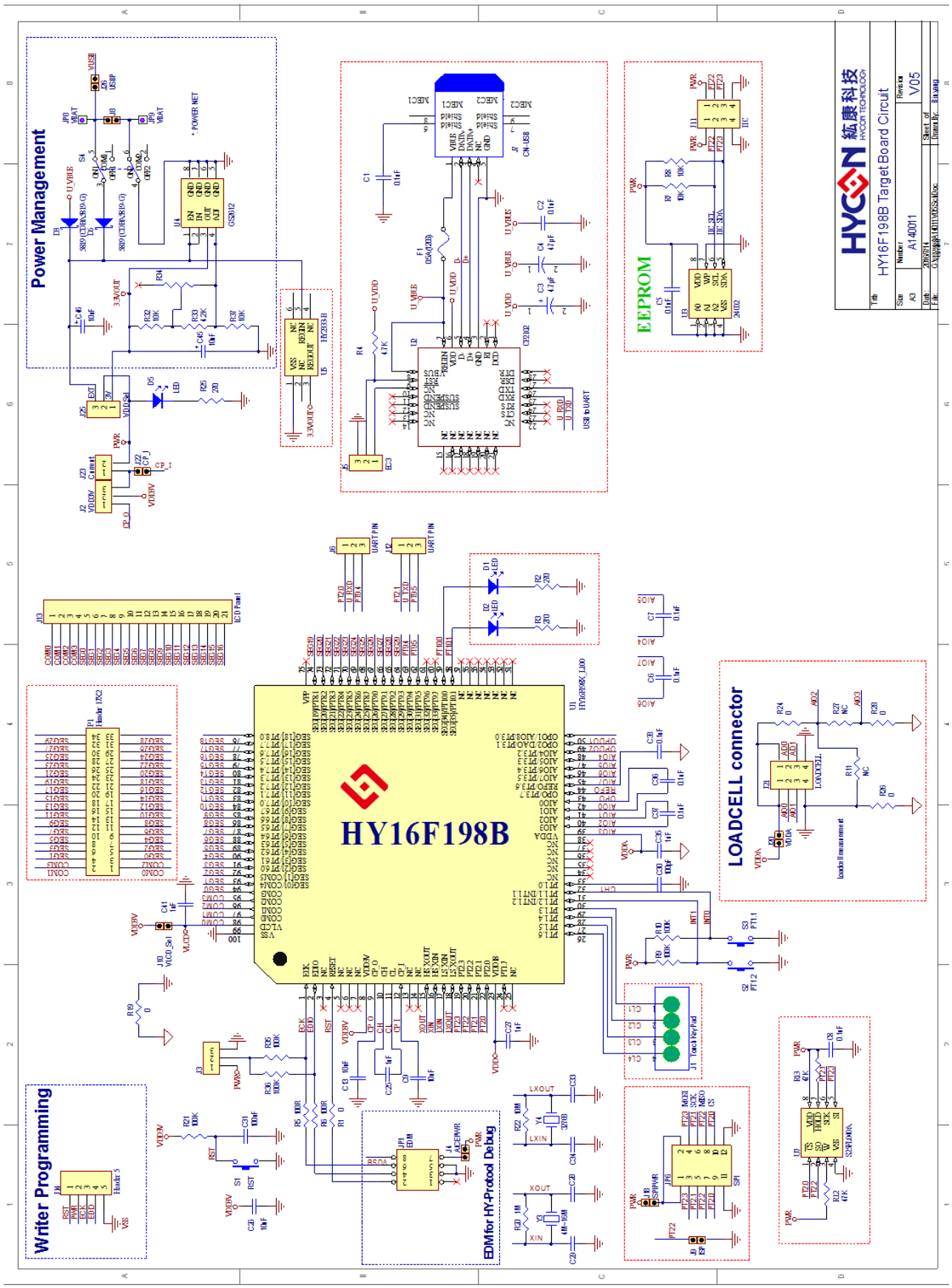
(7.2)內置溫度感測器 TPS。
(8)超低輸入雜訊運算放大器 OPAMP。
(9)16-bit Timer A
(10)16-bit Timer B 模組俱 PWM 波形產生功能
(11)16-bit Timer C 模組俱數位 Capture/Compare 功能
(12)硬體串列通訊 SPI 模組
(13)硬體串列通訊 I2C 模組
(14)硬體串列通訊 UART 模組
(15)硬體 RTC 時鐘功能模組
(16)硬體 Touch KEY 功能模組
(17)硬體 LCD Driver 4x36,6x34

3. 系統設計

3.1. 硬體說明

HY16F198B 對於時鐘及萬年曆應用，分別是(模式/清除)、(設定/上調&開始/停止)兩個按鈕，搭配內部硬體 LCD 驅動顯示目前模式資訊.

(A)MCU：HY16F198B
(B)顯示方式：HY16F198B 內部硬體驅動 4x36 LCD (LCD Driver Segment 4X36)
(C)電源電路：5.0V 轉 3.3V 電源系統
(D)控制按鍵：S2(模式/清除)、S3(設定/上調&開始/停止)
(E)HY16F198B Target Board Circuit 整體電路如下圖:
(紅色虛線區域為此範例未使用的電路)


Power Management
EEPROM
LOADCELL connector
HY16F198B
Writer Programming
EDM for HY-Protod Debug
HYCON TECHNOLOGY
HY16F198B Target Board Circuit

3.2. 功能說明

本程式包含下列功能：時鐘,萬年曆,鬧鈴,碼表.

A. 電源開啓：

初始化 LCD 顯示時間為 00.00.00 (時/分/秒).進入時鐘模式.並且每一秒時間持續往上數.

B. 按鍵功能說明：

- (模式/清除)鍵：可進行模式切換,切換的順序為:
  時鐘模式->萬年曆模式->鬧鐘模式->碼表模式

- (設定/上調&開始/停止)鍵：於任何模式下按下此鍵進行功能設定.
  - 時鐘模式：按下設定鍵，開始進行時間調整狀態.
    - 在時鐘調整狀態,設定鍵每短壓一下數值加 1.
    - 在時鐘調整狀態,長按 1 秒設定鍵進行滾動快速調整.
    - 在時鐘調整狀態,短按(模式/清除)鍵調整順序為時->分->秒->時(循環)
    - 在時鐘調整狀態,長按 2 秒(模式/清除)鍵離開調整模式
    - 在時鐘調整狀態,30 秒內未設定自動離開調整模式.

  - 萬年曆模式：按下設定鍵，開始進行日期調整狀態.
    - 在日期調整狀態,設定鍵每短壓一下數值加 1.
    - 在日期調整狀態,長按 1 秒設定鍵進行滾動快速調整.
    - 在日期調整狀態,短按(模式/清除)鍵調整順序為年->月->日->年(循環)
    - 在日期調整狀態,長按 2 秒(模式/清除)鍵離開調整狀態
    - 在日期調整狀態,30 秒內未設定自動離開調整模式.

  - 鬧鐘模式：按下設定鍵，開始進行鬧鐘時間調整狀態.
    - 在鬧鐘調整狀態,設定鍵每短壓一下數值加 1.
    - 在鬧鐘調整狀態,長按 1 秒設定鍵進行滾動快速調整.
    - 在鬧鐘調整狀態,短按(模式/清除)鍵調整順序為時->分->AL on(oFF)->時(循環)
    - 在鬧鐘調整狀態,長按 2 秒(模式/清除)鍵離開調整狀態
    - 在鬧鐘調整狀態, 30 秒內未設定自動離開調整模式.

  - 碼表模式：按下(開始/停止)鍵，開始進行碼表計數狀態.
    - 在碼表計數狀態,按下(開始/停止)鍵,可停止碼表計數.再按(開始/停止)鍵可繼續計數.
    - 在停止碼表計數狀態,按下(模式/清除)鍵,可將當前計數狀態清除(顯示 000000).
    - 在停止碼表計數,並且顯示 000000 的情況下,可按 (模式/清除)鍵離開碼表模式.

C. 補充說明：

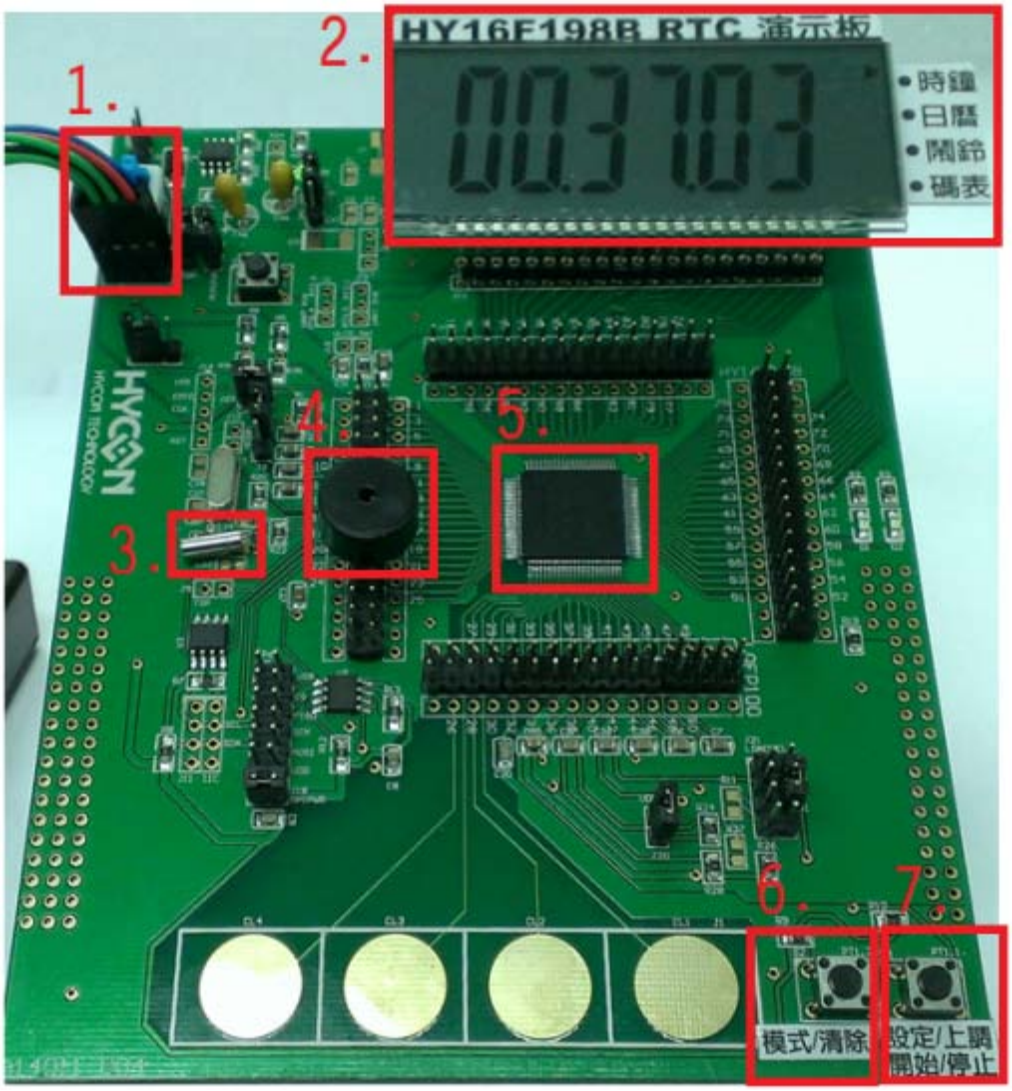
- 當 LCD 顯示 A 字,表示鬧鐘功能開啓,在到達設定時間後 Buzzer 開始鈴響 1 分鐘，鈴響期間可按任何按鍵取消鈴響.
- 關閉鬧鈴功能，需進入鬧鐘時間調整狀態,關閉 AL oFF，LCD 顯示的 A 字會消失.

4. 操作流程

4.1. 操作方式

初始化 LCD 顯示時間為 00.00.00 (時/分/秒).進入時鐘模式.並且每一秒時間持續往上數.

4.1.1. Target Board 功能簡介



圖片說明:

1. 5V 電源輸入端.
2. LCD 顯示.
3. 外掛 32768 晶振
4. Buzzer 輸出
5. MCU(HY16F198B)
6. (模式/清除)按鍵
7. (設定/上調&開始/停止)按鍵

4.1.2. LCD 顯示說明：

■ 時鐘模式 LCD 顯示方式：01 時 56 分 55 秒（時鐘模式固定使用 24 小制）

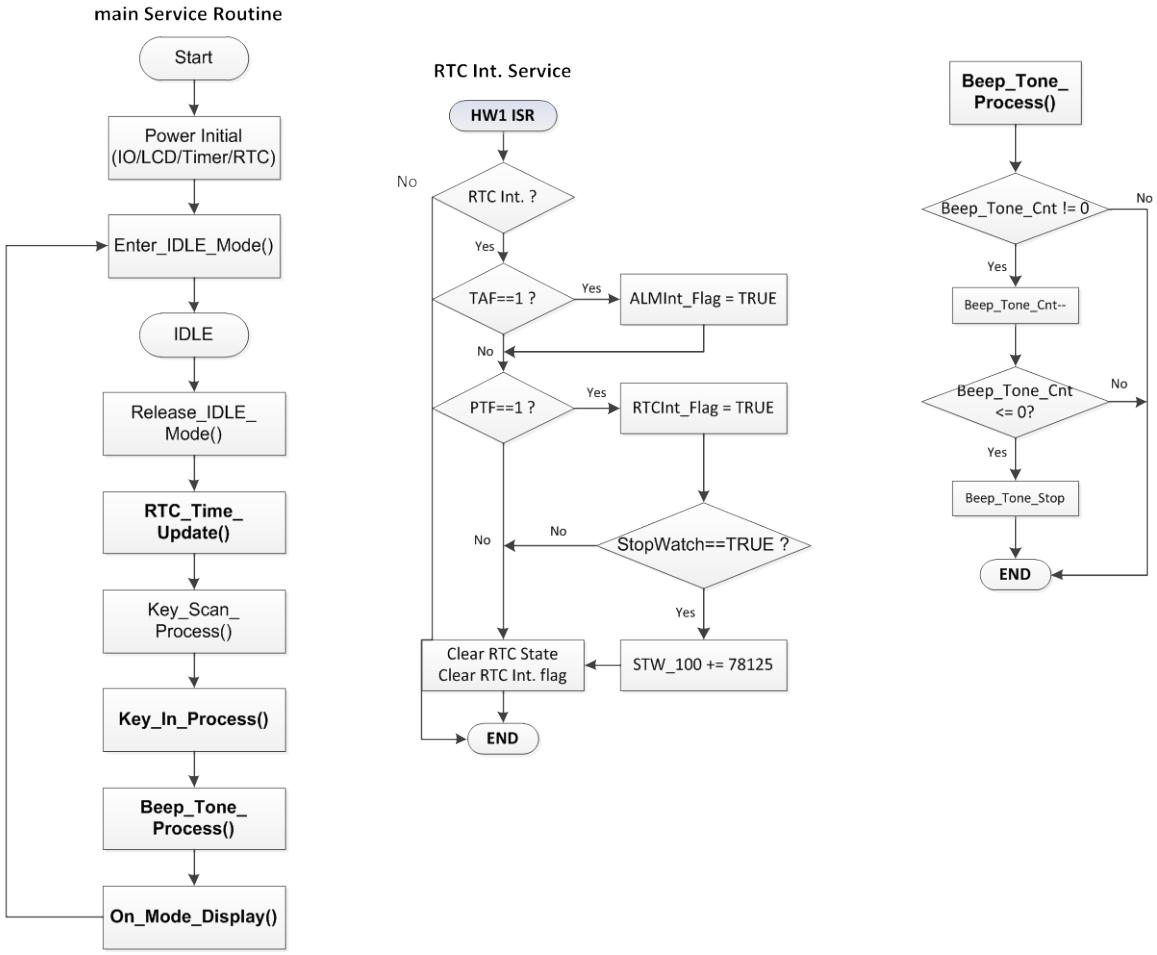
■ 萬年曆模式 LCD 顯示方式：2016 年 1 月 1 日

■ 鬧鐘模式 LCD 顯示方式：設定鬧鐘時間 01 時 00 分

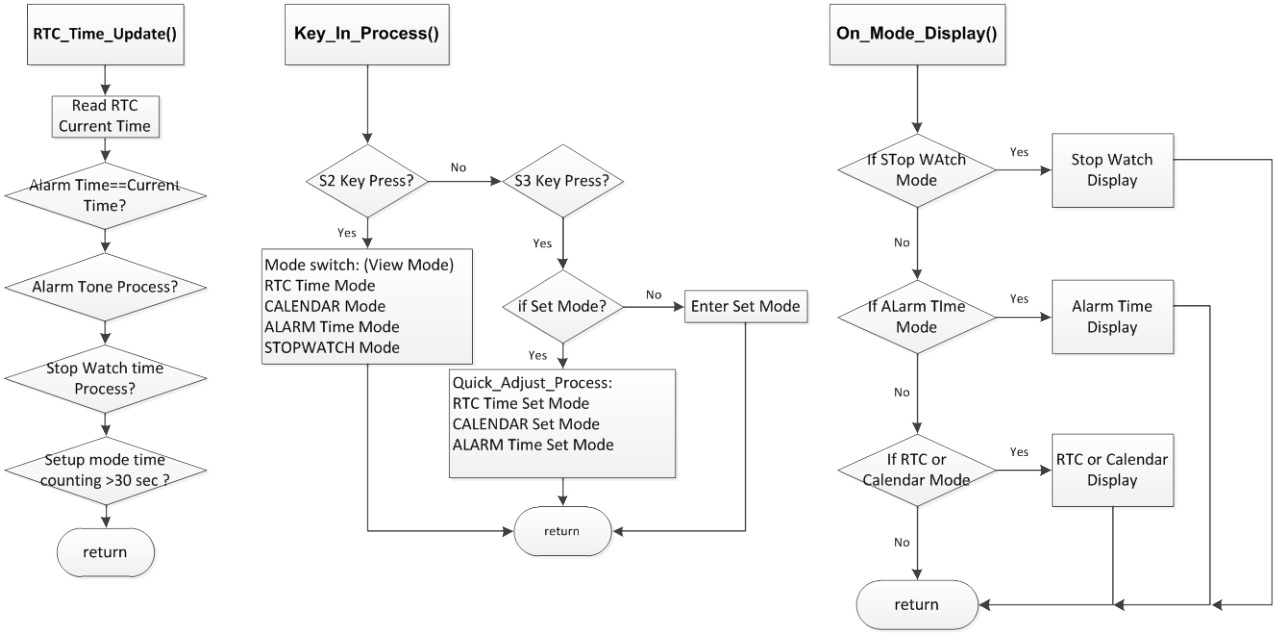
■ 碼表模式 LCD 顯示方式：開始進行碼表計數

4.2. 程式流程

4.2.1. Main Service & RTC Int. & Beep Tone Process



4.2.2. RTC Time Update & Key Process & On Mode Display Process


RTC_Time_Update()
Read RTC Current Time
Alarm Time==Current Time?
Alarm Tone Process?
Stop Watch time Process?
Setup mode time counting >30 sec ?
return
Key_In_Process()
S2 Key Press?
No
S3 Key Press?
Yes
Yes
Mode switch: (View Mode)
RTC Time Mode
CALENDAR Mode
ALARM Time Mode
STOPWATCH Mode
if Set Mode?
No
Enter Set Mode
Yes
Quick_Adjust_Process:
RTC Time Set Mode
CALENDAR Set Mode
ALARM Time Set Mode
return
On_Mode_Display()
If STop WAtch Mode
Yes
Stop Watch Display
No
If ALarm TIme Mode
Yes
Alarm Time Display
No
If RTC or Calendar Mode
Yes
RTC or Calendar Display
No
return

5. 技術規格

(1) Operation voltage : 2.4~3.6V
(2) Operation current : (HSRC=2MHz@CPU_CK:2MHz/2 Typ.0.6mA)
(3) IDLE Mode current :(LSRC=34KHz Typ. 5uA)
(4) RTC Int. Mode: (Mode 1 -> 1sec) or (Mode 2 -> 1/128 sec)
(5) Application Areas: Calendar、Stop Watch、Alarm、RTC.
(6) Operating Temperature:-40℃~ +85℃
(7) LCD Load 13.5uA
(8) Power consumption:

MCU Status / Mode	MCU Process Time	Operation current (HSRC=2MHz)	IDLE Time	IDLE Mode current (LSRC=34KHz)	Average current (Sec)
Mode 1: RTC Display Calendar Display Alarm Display Stop Watch Display (1sec)	13.3mS	13.9uA (Note1)	986.7mS	4.934uA (Note3)	18.834uA (Note5)
Mode 2: Stop Watch counter Set Mode (1/128 sec)	0.9425mS	0.566uA (Note2)	6.87mS	0.0344uA (Note4)	76.8512uA (Note6)

■ Operation current

Note1: Mode 1 -> 13.3mS*0.6mA=13.9uA @ 1sec
Note2: Mode 2 -> 0.9425mS*0.6mA=0.566uA @1/128sec

■ IDLE current

Note3: Mode 1 -> 986.7mS*5uA=4.934uA @1sec
Note4: Mode 2 -> 6.87mS*5uA=0.0344uA @1/128sec

■ Average current (Sec)= Operation current + IDLE current

Note5: Mode 1 -> 1*(13.9uA+4.934uA)=18.834uA
Note6: Mode 2 -> 128*(0.566uA+0.0344uA)= 76.8512uA

6. 成果總結

以 HY16F198B 為主控結合內部高精度 RTC 模式.搭配外部 32768 晶體震盪器，可以達到高準度低耗電的時鐘及萬年曆.不需要額外增加 IC 即可用 MCU 本身完成高準度計時.相較於吳內建 RTC 之 MCU,HY16F198B 不論在時間計算的精準度、整體電路的大小、功耗方面皆有相當出色的表現.

7. 附件

7.1. 範例程式

```
/*---------------------------------------------------------------------------*/
/* Header file include                                                       */
/*---------------------------------------------------------------------------*/
#include "HY16F198.h"
#include "System.h"
#include "ModuleID.h"
#include "SpecialMacro.h"
#include "Sysinfra.h"
#include "DrvClock.h"
#include "DrvGPIO.h"
#include "DrvLCD.h"
#include "DrvREG32.h"
#include "DrvADC.h"
#include "DrvTimer.h"
#include "DrvUART.h"
#include "user_DEF.h"
#include "DrvRTC.h"

/*---------------------------------------------------------------------------*/
/* MAIN  function                                                            */
/*---------------------------------------------------------------------------*/
int main(void)
{
    User_Sample_Init();

//------Enter Main program loop-----------------
    while(1)
    {
        //SYS enter IDLE mode !
        Enter_IDLE_Mode();
        SYS_LowPower(1);                    //待機模式（Idle  mode）
        Release_IDLE_Mode();

        //SYS wakeup going !
        RTC_Time_Update();
        Key_Scan_Process();
        Key_In_Process();                   //Key in processing.
        Beep_Tone_Process();
        On_Mode_Display();
    }
    return 0;
}

/*---------------------------------------------------------------------------*/
/* Exception Service Routines                                                */
/*---------------------------------------------------------------------------*/
void tlb_exception_handler()
{
    //procedure define by customer.
    asm("nop");
    asm("nop");
}


/*---------------------------------------------------------------------------*/
/* Function Name: HW1_ISR()                                                  */
/* Description  : WDT & RTC & Timer A/B/C interrupt Service Routine (HW1).   */
/*---------------------------------------------------------------------------*/
void HW1_ISR(void)
{
    if( DrvRTC_ReadState() & 0x05 )
    {
        //TAF == 1 ?
        if( DrvRTC_ReadState() & 0x01 )
            UserFlag0.b_ALMInt_Flag = TRUE;
        //PTF == 1 ?
        if( DrvRTC_ReadState() & 0x04 )
        {
            UserFlag0.b_RTCInt_Flag = TRUE;
            if ( UserFlag1.b_StpWGo_Flag == TRUE )
```

```
            STW_100 += 78125;                   //Stop Watch 1/100Sec time base ++ ( 1 count= 1/128Sec ).
        }
        DrvRTC_ClearState( E_DRVRTC_CLEAR_ALL );
        DrvRTC_ClearIntFlag();
    }
}


* ---------------------------------------------------------------------------
 * File Name    : user_sample.c
 * IDE tooling  : AndeSight C/C++ IDE, version: 2.0.1 Build ID : 201303201736
 * Device Ver.  : V0.9 crt0.o for HY16F19xx & HY16F18xx MCU.
 * Library Ver. : 1.1
 * MCU Device   : HY16F198B-L100 @3.0V
 * Description  : This file implements the customer's function.
 * Created Date : 2016/06/29
 *
 ****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Header file include                                                       */
/*---------------------------------------------------------------------------*/
#include "HY16F198.h"
#include "System.h"
#include "DrvClock.h"
#include "DrvPMU.h"
#include "DrvGPIO.h"
#include "DrvLCD.h"
#include "DrvREG32.h"
#include "DrvRTC.h"
#include "DrvTimer.h"
#include "DrvUART.h"
#include "DrvFlash.h"
#include "ModuleID.h"
#include "SpecialMacro.h"
#include "Sysinfra.h"
#include "user_DEF.h"
#include "user_LCD.h"
#include "DrvRTC.h"

/*---------------------------------------------------------------------------*/
/* Structure definition                                                      */
/*---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------*/
/* Global variable definition                                                */
/*---------------------------------------------------------------------------*/
S_DRVRTC_TIME_DATA_T sCurTime;               //Declare a RTC structure for current RTC time.
S_DRVRTC_TIME_DATA_T sCurAlarm;              //Declare a RTC structure for current Alarm time.
unsigned char DisplayBuffer[18];
unsigned char Beep_Tone_Cnt;
unsigned char Key_Code_Buf;
unsigned char Key_Code_No;
unsigned char Key_DB_Cnt;
unsigned char Mode_Index;
unsigned int  SetItem_Index;
unsigned int  STW_100, STW_Sec, STW_Min;
unsigned int  Tick500mS_Cnt;
unsigned int  AlmTone_Cnt;
unsigned int  SetMode_Cnt;
unsigned int  QAdjTime_Cnt;


/*---------------------------------------------------------------------------*/
/* Constant value definition                                                 */
/*---------------------------------------------------------------------------*/
const unsigned char code_seg[] =
{
    seg_a+seg_b+seg_c+seg_d+seg_e+seg_f,       //00, char "0"
    seg_b+seg_c,                             //01, char "1"
    seg_a+seg_b+seg_d+seg_e+seg_g,             //02, char "2"
    seg_a+seg_b+seg_c+seg_d+seg_g,             //03, char "3"
    seg_b+seg_c+seg_f+seg_g,                  //04, char "4"
    seg_a+seg_c+seg_d+seg_f+seg_g,             //05, char "5"
    seg_a+seg_c+seg_d+seg_e+seg_f+seg_g,       //06, char "6"
    seg_a+seg_b+seg_c,                       //07, char "7"
    seg_a+seg_b+seg_c+seg_d+seg_e+seg_f+seg_g,  //08, char "8"
```

```
    seg_a+seg_b+seg_c+seg_d+seg_f+seg_g,       //09, char "9"
    seg_a+seg_b+seg_c+seg_e+seg_f+seg_g,       //0A, char "A"
    seg_c+seg_d+seg_e+seg_f+seg_g,            //0B, char "b"
    seg_a+seg_d+seg_e+seg_f,                  //0C, char "C"
    seg_b+seg_c+seg_d+seg_e+seg_g,            //0D, char "d"
    seg_a+seg_d+seg_e+seg_f+seg_g,            //0E, char "E"
    seg_a+seg_e+seg_f+seg_g                   //0F, char "F"
};

/*---------------------------------------------------------------------------*/
/* Function prototype declaration                                            */
/*---------------------------------------------------------------------------*/

//------MISC Function declaration---------------
void User_Sample_Init(void);
void Delay(unsigned int num);
void Beep_Tone_Start(void);
void Beep_Tone_Stop(void);
void Key_Scan_Process(void);
void Key_In_Process(void);
void Beep_Tone_Process(void);
void Enter_IDLE_Mode(void);
void Release_IDLE_Mode(void);

//------RTC Function declaration----------------
void RTC_Init(void);
void RTC_Time_Update(void);
void Quick_Adjust_Process(void);
int ComputeWeek(int TempYear, int TempMonth, int TempDay);

//------GPIO Function declaration---------------
void GPIO_Init(void);

//------TIMER Function declaration--------------
void TIMER_Init(void);

//------LCD Function declaration----------------
void LCD_Init(void);
void LCD_RAM_Clear(void);
void LCD_Display_HYcon(void);
void On_Mode_Display(void);
void RAM2LCD( unsigned char *Buffer_Adr );

/*---------------------------------------------------------------------------*/
/* Function Name: User_Sample_Init                                           */
/* Description  : Sample code Initialization Subroutines                     */
/*---------------------------------------------------------------------------*/
void User_Sample_Init(void)
{
    unsigned int InitGo_cnt;
    unsigned char step;

    SYS_DisableGIE();                     //Disable Global Interrupt.

//------Sys. clock setting----------------------
    DrvCLOCK_SelectIHOSC( TRIM_HAO2MHZ );      //Select HAO= 2MHz.
    DrvCLOCK_EnableHighOSC(E_INTERNAL, 60);    //Enable internal HAO.
    DrvCLOCK_EnableLowOSC(E_EXTERNAL, 130000); //Enable external LSXT 32768Hz.
    DrvCLOCK_SelectMCUClock(0, 0);            //Select MCUCKS= HS_CK/1= 2MHz.
//  DrvCLOCK_SelectMCUClock(1, 0);            //Select MCUCKS= LS_CK/1= 32768Hz.

//------MISC. system initial--------------------
    GPIO_Init();
    LCD_Init();
    RTC_Init();
    TIMER_Init();

    UserFlag0._byte = 0;
    UserFlag1._byte = 0;
    Mode_Index = RTC_TIME_MODE;              //power on's default mdoe is RTC time mode.
    SYS_EnableGIE( 4, 0x02 );                //Enable HW1 only.

//------Power on initial going------------------
//Step1: Beep tone & turn all LCD display on about 1Sec.
//Step2: LCD display "HYcon" about 1Sec.
//Step3: Normally system display.
```

```
//
    for ( step=1; step <3; step++ )
    {
        if ( step == 1 )
        {
            DrvLCD_DisplayMode( E_LCD_PIXELON );       //all LCD display on.
            Beep_Tone_Start();
        }
        if ( step == 2 )
        {
            DrvLCD_DisplayMode( E_LCD_NORMAL );        //LCD at normal mode.
            Beep_Tone_Stop();
            LCD_Display_HYcon();
        }

        for ( InitGo_cnt= 0; InitGo_cnt < INIT_GO_TIME; )
        {
            if ( UserFlag0.b_RTCInt_Flag == TRUE )
            {
                InitGo_cnt++;
                UserFlag0.b_RTCInt_Flag = FALSE;
            }
        }
    }
    LCD_RAM_Clear();
    UserFlag0.b_DspGo_Flag = TRUE;
    On_Mode_Display();
}

/*-------------------------------------------------------------------------------*/
/* Function Name: Key_Scan_Process                                               */
/* Description  : S2 & S3 key scanning, scan period is 7.8125mS.                 */
/*-------------------------------------------------------------------------------*/
void Key_Scan_Process(void)
{
    unsigned char key_code;

    if ( UserFlag1.b_KScan_Flag == FALSE )
        return;

//------Key scanning go !-----------------------
    key_code = NULL_KEY_NO;

    if ( DrvGPIO_GetBit( E_PT1, 2 ) == 0 )            // Check S2
    {
          key_code = S2_KEY_NO;
    }
    if ( DrvGPIO_GetBit( E_PT1, 1 ) == 0 )            // Check S3
    {
          key_code = S3_KEY_NO;
    }

//------Key in debounce-------------------------
    Key_DB_Cnt++;
    if ( key_code != Key_Code_Buf )
    {
        Key_Code_Buf = key_code;
        Key_DB_Cnt = 0;
    }
    else
    {
        if (Key_DB_Cnt >= KEY_DB_TIME)
        {
            Key_DB_Cnt = 0;
            if ( Key_Code_Buf == NULL_KEY_NO )
            {
                UserFlag1.b_KeyOn_Flag = FALSE;
                UserFlag1.b_KScan_Flag = FALSE;
                QAdjTime_Cnt = 0;
            }
            else
            {
                if (UserFlag1.b_KeyOn_Flag == FALSE)
                {
                    UserFlag1.b_KeyOn_Flag = TRUE;
                    UserFlag0.b_KInPro_Flag = TRUE;
```

```
                Key_Code_No = Key_Code_Buf;
            }
        }
    }
}

/*-------------------------------------------------------------------------------*/
/* Function Name: Key_In_Process                                                 */
/* Description  : Key in tasks processing.                                       */
/*-------------------------------------------------------------------------------*/
void Key_In_Process(void)
{

    //Key processing for 1st key in
    if( UserFlag0.b_KInPro_Flag == TRUE )
    {
        UserFlag0.b_KInPro_Flag = FALSE;
        UserFlag0.b_DspGo_Flag = TRUE;
        SetMode_Cnt = 0;

        //Alarm Tone is going ?
        if ( UserFlag1.b_AlmGo_Flag == TRUE )
        {
            UserFlag1.b_AlmGo_Flag = FALSE;
            Beep_Tone_Start();
            return;
        }

        switch( Key_Code_No )
        {
            //S2 key go !
            case S2_KEY_NO:
            {
                //on View mode !
                if ( UserFlag1.b_SetEI_Flag == FALSE )
                {
                    if ( ( Mode_Index == STOP_WATCH_MODE ) && ( (STW_Min+STW_Sec+STW_100) > 0 ) )
                    {
                        if ( UserFlag1.b_StpWGo_Flag == FALSE )
                        {
                            STW_Min = 0;
                            STW_Sec = 0;
                            STW_100 = 0;
                        }
                    }
                    //to next mode
                    else
                    {
                        Mode_Index = Mode_Index << 1;
                        if ( Mode_Index == 0 ) { Mode_Index = RTC_TIME_MODE; }
                    }
                }
                //on setup mode.
                else
                {
                    SetItem_Index++;
                    QAdjTime_Cnt = KEY_SET_OFF_TIME;
                    if ( SetItem_Index >= 3 ) { SetItem_Index = 0; }
                }
                break;
            }
            //S3 key go !
            case S3_KEY_NO:
            {
                //enter setup mode.
                if ( UserFlag1.b_SetEI_Flag == FALSE )
                {
                    if ( Mode_Index == STOP_WATCH_MODE )
                    {
                        UserFlag1.b_StpWGo_Flag ^= 1;
                    }
                    else
                    {
                        UserFlag1.b_SetEI_Flag = TRUE;
                        SetItem_Index = 0;
```

```
                }
            }
            //current adust go !
            else
            {
                QAdjTime_Cnt = QADJ_TIME_1HZ;
                Quick_Adjust_Process();
            }
            break;
        }
        default:
            break;
    }
    Beep_Tone_Start();
}
//Key processing for repeat key
else if ( (UserFlag1.b_KeyOn_Flag == TRUE) && (UserFlag1.b_SetEI_Flag == TRUE) && (QAdjTime_Cnt > 0) )
{
    SetMode_Cnt = 0;
    QAdjTime_Cnt--;
    if ( QAdjTime_Cnt > 0 )
        return;
    UserFlag0.b_DspGo_Flag = TRUE;

    switch( Key_Code_No )
    {
        //S2 key repeated going
        case S2_KEY_NO:
        {
            UserFlag1.b_SetEI_Flag = FALSE;
            Beep_Tone_Start();
            break;
        }
        //S3 key repeated going
        case S3_KEY_NO:
        {
            QAdjTime_Cnt = QADJ_TIME_8HZ;
            Quick_Adjust_Process();
            break;
        }
        default:
            break;
    }
}
}

/*-----------------------------------------------------------------------------*/
/* Function Name: Enter_IDLE_Mode                                              */
/* Description  : Prepare to enter IDLE mode.                                  */
/*-----------------------------------------------------------------------------*/
void Enter_IDLE_Mode(void)
{
    //Stop HAO ?
    if ( ( Beep_Tone_Cnt == 0 ) && ( UserFlag1._byte == 0 ) )
    {
        Tick500mS_Cnt = 0;
        UserFlag0.b_IRQ128_Flag = FALSE;
        UserFlag0.b_HAO_EI_Flag = FALSE;
        DrvRTC_PeriodicTimeEnable( E_DRVRTC_1_SEC );       //Set RTC int. period time= 1/1Sec= 1Sec.
        DrvCLOCK_SelectMCUClock(1, 0);                    //Select MCUCKS= LS_CK/1= 32768Hz.
        Delay( 5 );
        DrvCLOCK_CloseIHOSC();                           //Stop 2MHz HAO.
        Delay( 5 );
    }
    else
    {
        UserFlag0.b_HAO_EI_Flag = TRUE;
        UserFlag0.b_IRQ128_Flag = TRUE;
        DrvRTC_PeriodicTimeEnable( E_DRVRTC_1_128_SEC );    //Set RTC int. period time= 1/128Sec= 7.8125mS.
    }
}

/*-----------------------------------------------------------------------------*/
/* Function Name: Release_IDLE_Mode                                            */
/* Description  : Prepare to release from IDLE mode.                           */
/*-----------------------------------------------------------------------------*/
```

```
void Release_IDLE_Mode(void)
{
    //Release by KEY ?
    if ( DrvGPIO_PT1_GetIntFlag() )
    {
        DrvGPIO_PT1_ClearIntFlag( 0x06 );      //clear PT1.1 & PT1.2 int pending flag.
        if ( UserFlag1.b_KScan_Flag == FALSE )
            UserFlag0.b_HAO_EI_Flag = FALSE;
        UserFlag1.b_KScan_Flag = TRUE;
    }
    //Enable HAO
    if ( UserFlag0.b_HAO_EI_Flag == FALSE )
    {
        clk_00 = 0x4545;                    //Enable internal 2MHz HAO & enter 32768Hz.
        Delay( 10 );                        //delay 1.5mS for HAO stability when MCUCK= 32768Hz.
        DrvCLOCK_SelectMCUClock(0, 0);        //Select MCUCKS= HS_CK/1= 2MHz.
        Delay( 10 );                        //delay 1.5mS for HAO stability when MCUCK= 32768Hz.
    }
}

/*-------------------------------------------------------------------------*/
/* Function Name: Beep_Tone_Process                                        */
/* Description  : Beep tone time processing.                               */
/*-------------------------------------------------------------------------*/
void Beep_Tone_Process(void)
{
    if (Beep_Tone_Cnt != 0)
    {
        Beep_Tone_Cnt--;
        if (Beep_Tone_Cnt <= 0)
            Beep_Tone_Stop();
    }
}

/*-------------------------------------------------------------------------*/
/* Function Name: Beep_Tone_Start                                          */
/* Description  : Start PWM beep tone.                                     */
/*-------------------------------------------------------------------------*/
void Beep_Tone_Start(void)
{
    Beep_Tone_Cnt = BEEP_TONE_TIME;
    DrvTMB_Open( E_TMB_MODE0, E_TMB_NORMAL, PWM_4KHZ_TIME );       //set TMB working on normal mode & TMB
clock int period= PWM_4KHZ_TIME/HS_CK= 250uS.
    DrvPWM0_Open( 0, 1, 2 );                                    //Set PT2.0= PWMO0, PT2.1= PWMO1 and PWMO0
working on PWMA mdoe with normal pulse output.
    DrvPWM1_Open( 0, 0, 2 );                                    //Set PT2.0= PWMO0, PT2.1= PWMO1 and PWMO1
working on PWMA mdoe with reverse pulse output.
}

/*-------------------------------------------------------------------------*/
/* Function Name: Beep_Tone_Stop                                           */
/* Description  : Stop PWM beep tone.                                      */
/*-------------------------------------------------------------------------*/
void Beep_Tone_Stop(void)
{
    Beep_Tone_Cnt = 0;
    DrvTMB_Close();
    DrvPWM0_Close();
    DrvPWM1_Close();
    DrvGPIO_ClrPortBits( E_PT2, 0x03 );       //set PT2[1:0] as output low.
}

/*-------------------------------------------------------------------------*/
/* Function Name: Delay()                                                  */
/* Description  : Software delay subroutines.                              */
/*-------------------------------------------------------------------------*/
void Delay(unsigned int num)
{
    for( ; num >0; num-- )
        asm( "NOP" );
}

/*************************************************************************/
/* File Name    : user_GPIO.c                                            */
/* Description  : This file implements the customer's GPIO function.     */
/*************************************************************************/
```

```
/*-------------------------------------------------------------------------*/
/* Function Name: GPIO_Init                                                */
/* Description  : GPIO Initialization Subroutines                          */
/*-------------------------------------------------------------------------*/
void GPIO_Init(void)
{
    DrvGPIO_ClkGenerator( E_LS_CK, 1 );                    //Set IO sampling clock input source is LS_CK,
and IOCLK is LS_CK/1.
    DrvGPIO_PT1_EnablePullHigh( 0x06 );                     //enable PT1.1 & PT1.2 pull high 75K ohm.
    DrvGPIO_PT1_EnableINPUT( 0x06 );                       //set PT1.1 & PT1.2  as input port.
    DrvGPIO_PT1_IntTriggerPorts( 0x06, E_N_Edge );          //set PT1.1 & PT1.2 as interrupt trigger
method is negative edge.
    DrvGPIO_PT1_DisableOUTPUT( 0xFF );                      //close Output PT1[7:0] OE.
    DrvGPIO_PT1_EnableINT( 0x06 );                         //PT1.1 & PT1.2 interrupt enable.
    DrvGPIO_PT1_ClearIntFlag( 0xFF );                       //clear PT1[7:0] interrupt flag.

//------PWM function initial for Buzzer---------
    DrvGPIO_Open( E_PT2, 0x03, E_IO_OUTPUT );               //set PT2[1:0] as output port.
    DrvPWM_CountCondition( PWM_4KHZ_TIME/2, PWM_4KHZ_TIME/2 );  //Set TBC1 as 50% duty, TBC2 as 50% duty.
    DrvGPIO_ClrPortBits( E_PT2, 0x03 );                     //set PT2[1:0] as output low.
}


/*************************************************************************/
/* File Name    : user_RTC.c                                             */
/* Description  : This file implements the customer's RTC function.      */
/*************************************************************************/
/*-------------------------------------------------------------------------*/
/* Function Name: RTC_Init                                                 */
/* Description  : RTC Initialization Subroutines                           */
/*-------------------------------------------------------------------------*/
void RTC_Init(void)
{
    DrvRTC_ClkConfig( 1 );                          //Enable RTC clock, ENRTCK = 1.
    DrvRTC_WriteEnable();                           //Unlock RTC IP writing function.
    DrvRTC_ClockSource( 0 );                        //Select RTC's clock source from 32768Hz LSXT.
    DrvRTC_Enable();                               //Enable RTC IP.

//------initial current RTC time----------------
//default RTC time is 00:00 24Hr Jan/01/2016.
//
    sCurTime.u8cClockDisplay = E_DRVRTC_HOUR_24;       //set RTC Hour format is 24H.
    sCurTime.u8cAmPm = 0;                           //set RTC time is AM.
    sCurTime.u32cSecond = 0;
    sCurTime.u32cMinute = 0;
    sCurTime.u32cHour = 0;
    sCurTime.u32cDay =  1;
    sCurTime.u32cMonth = 1;
    sCurTime.u32Year = 2016;
    sCurTime.u32cDayOfWeek = ComputeWeek(sCurTime.u32Year, sCurTime.u32cMonth, sCurTime.u32cDay);
    DrvRTC_Write( DRVRTC_CURRENT_TIME, &sCurTime);

//------initial current Alarm time--------------
//default Alarm Time is 00:00 24Hr Jan/01/2016.
//
    sCurAlarm.u8cClockDisplay = E_DRVRTC_HOUR_24;      //set alarm Hour format is 24H.
    sCurAlarm.u8cAmPm = 0;                          //set alarm time is AM.
    sCurAlarm.u32cSecond = 0;
    sCurAlarm.u32cMinute = 0;
    sCurAlarm.u32cHour = 0;
    sCurAlarm.u32cDay = sCurTime.u32cDay;
    sCurAlarm.u32cMonth = sCurTime.u32cMonth;
    sCurAlarm.u32Year = sCurTime.u32Year = 2016;
    sCurAlarm.u32cDayOfWeek = sCurTime.u32cDayOfWeek;
    DrvRTC_Write( DRVRTC_ALARM_TIME, &sCurTime);

//------RTC time running !----------------------
    DrvRTC_PeriodicTimeEnable(E_DRVRTC_1_128_SEC);     //Set RTC int. period time= 1/128Sec= 7.8125mS.
    DrvRTC_ClearIntFlag();                          //Clear RTC int. request flag.
    DrvRTC_EnableInt();                            //Enable RTC interrupt !
}

/*-------------------------------------------------------------------------*/
/* Function Name: RTC_Time_Update                                          */
/* Description  : RTC display                                              */
```

```
/*-------------------------------------------------------------------------------*/
void RTC_Time_Update(void)
{

//------ALARM compare with TIME-----------------
    UserFlag0.b_ALMInt_Flag = FALSE;
    DrvRTC_Read( DRVRTC_CURRENT_TIME, &sCurTime );
    if ( (UserFlag0.b_AlmEI_Flag == TRUE)
    && (sCurTime.u32cHour == sCurAlarm.u32cHour)
    && (sCurTime.u32cMinute == sCurAlarm.u32cMinute)
    && (sCurTime.u32cSecond == sCurAlarm.u32cSecond) )
    {
        UserFlag1.b_AlmGo_Flag = TRUE;
        AlmTone_Cnt = 0;
    }

//------RTC IRQ processing----------------------
    if ( UserFlag0.b_RTCInt_Flag == FALSE )
        return;
    UserFlag0.b_RTCInt_Flag = FALSE;

    //Tick time update !
    if ( UserFlag0.b_IRQ128_Flag == TRUE )
    {
        Tick500mS_Cnt++;
        if ( Tick500mS_Cnt > (10000000u/78125u) ) { Tick500mS_Cnt = 0; }
    }

    //Alarm Tone process !
    if ( UserFlag1.b_AlmGo_Flag == TRUE )
    {
        //halt tick process !
        if ( Tick500mS_Cnt < ( 5000000u/78125u) )
        {
            if ( (Tick500mS_Cnt & 0x0F) == 0x08 )
            {
                AlmTone_Cnt++;
                if ( AlmTone_Cnt >= ALM_TONE_TIME )
                {
                    UserFlag1.b_AlmGo_Flag = FALSE;
                    Beep_Tone_Stop();
                }
                else
                    Beep_Tone_Start();
            }
            else if ( (Tick500mS_Cnt & 0x0F) == 0 )
                Beep_Tone_Stop();
        }
        UserFlag0.b_DspGo_Flag = TRUE;
    }

    //Stop Watch time update !
    if ( UserFlag1.b_StpWGo_Flag == TRUE )
    {
        //update if over 1Sec.
        if ( STW_100 >= (128*78125) )
        {
            STW_100 = 0;
            STW_Sec++;
            if ( STW_Sec >= 60 )
            {
                STW_Sec = 0;
                STW_Min++;
                if ( STW_Min >= 100 ) { STW_Min = 0; }
            }
        }
        UserFlag0.b_DspGo_Flag = TRUE;
    }

    //Setup mode time counting !
    if ( UserFlag1.b_SetEI_Flag == TRUE )
    {
        SetMode_Cnt++;
        if ( SetMode_Cnt >= SET_RELEASE_TIME )
            UserFlag1.b_SetEI_Flag = FALSE;
        UserFlag0.b_DspGo_Flag = TRUE;
```

```
    }

    //don't update display if mode is ALARM and no any events.
    if ( !((Mode_Index == ALARM_TIME_MODE) && (Beep_Tone_Cnt == 0) && (UserFlag1._byte == 0)) )
        UserFlag0.b_DspGo_Flag = TRUE;
}

/*-------------------------------------------------------------------------------*/
/* Function Name: Quick_Adjust_Process                                           */
/* Description  : Current time quickly adjust increase.                          */
/*-------------------------------------------------------------------------------*/
void Quick_Adjust_Process(void)
{
//------Adjust go for Alarm set mode------------
    if ( Mode_Index == ALARM_TIME_MODE )
    {
        DrvRTC_Read( DRVRTC_ALARM_TIME, &sCurTime);

        if ( SetItem_Index == 0 )
        {
            sCurAlarm.u32cHour++;
            if ( sCurAlarm.u32cHour >= 24 )
                sCurAlarm.u32cHour = 0;
        }
        if ( SetItem_Index == 1 )
        {
            sCurAlarm.u32cMinute++;
            if ( sCurAlarm.u32cMinute >= 60 )
                sCurAlarm.u32cMinute = 0;
        }
        if ( SetItem_Index == 2 )
        {
            QAdjTime_Cnt = 0;
            if ( UserFlag0.b_AlmEI_Flag == TRUE )
            {
              //DrvRTC_AlarmDisable();
                UserFlag0.b_AlmEI_Flag = FALSE;
                UserFlag0.b_ALMInt_Flag = FALSE;
            }
            else
            {
                UserFlag0.b_AlmEI_Flag = TRUE;
              //DrvRTC_AlarmEnable();
            }
        }
        DrvRTC_Write( DRVRTC_ALARM_TIME, &sCurTime);
    }

//------Adjust go for RTC set mode--------------
    if ( (Mode_Index == RTC_TIME_MODE) || (Mode_Index == CALENDAR_MODE) )
    {
        DrvRTC_Read( DRVRTC_CURRENT_TIME, &sCurTime );

        //current time adjust
        if ( Mode_Index == RTC_TIME_MODE )
        {
            if ( SetItem_Index == 0 )
            {
                sCurTime.u32cHour++;
                if ( sCurTime.u32cHour >= 24 )
                    sCurTime.u32cHour = 0;
            }
            if ( SetItem_Index == 1 )
            {
                sCurTime.u32cMinute++;
                if ( sCurTime.u32cMinute >= 60 )
                    sCurTime.u32cMinute = 0;
            }
            if ( SetItem_Index == 2 )
            {
                QAdjTime_Cnt = 0;
                sCurTime.u32cSecond = 0;
            }
            DrvRTC_Write( DRVRTC_CURRENT_TIME, &sCurTime );
        }
        //calendar adjust
```

```
        else
        {
            if ( SetItem_Index == 0 )
            {
                sCurTime.u32Year++;
                if ( sCurTime.u32Year >= 2100 )
                    sCurTime.u32Year = 2016;
            }
            if ( SetItem_Index == 1 )
            {
                sCurTime.u32cMonth++;
                if ( sCurTime.u32cMonth >= 13 )
                    sCurTime.u32cMonth = 1;
            }
            if ( SetItem_Index == 2 )
            {
                sCurTime.u32cDay++;
                if ( (sCurTime.u32cMonth == 2) && (sCurTime.u32cDay >= 30) )
                    sCurTime.u32cDay = 1;
                else if ( (sCurTime.u32cDay >= 31) && ((sCurTime.u32cMonth == 4) || (sCurTime.u32cMonth ==
6) || (sCurTime.u32cMonth == 9) || (sCurTime.u32cMonth == 11)) )
                    sCurTime.u32cDay = 1;
                else if ( sCurTime.u32cDay >= 32 )
                    sCurTime.u32cDay = 1;
            }
            DrvRTC_Write( DRVRTC_CURRENT_TIME, &sCurTime );

            //Leap year checking
            DrvRTC_Read( DRVRTC_CURRENT_TIME, &sCurTime );
            if ( ((rtc_00 & 0x00080000) == 0) && (sCurTime.u32cMonth == 2) && (sCurTime.u32cDay > 28) )
                sCurTime.u32cDay = 1;
            DrvRTC_Write( DRVRTC_CURRENT_TIME, &sCurTime );

            sCurTime.u32cDayOfWeek = ComputeWeek(sCurTime.u32Year, sCurTime.u32cMonth,
sCurTime.u32cDay);
        }
    }
}

/*---------------------------------------------------------------------------*/
/* Function Name: ComputeWeek                                                */
/* Description  : Compute Week Subroutines.                                  */
/*---------------------------------------------------------------------------*/
int ComputeWeek(int TempYear, int TempMonth, int TempDay)
{
    int TempWeek;
    if (TempMonth >= 3)
    {
        TempMonth = TempMonth - 2;
    }
    else
    {
        TempMonth = TempMonth + 10;
        TempYear--;
    }

    TempWeek = TempYear + (int)(TempYear / 4) - (int)(TempYear / 100) + (int)(TempYear / 400) + (int)(2.6
* TempMonth - 0.2) + TempDay;
    TempWeek = TempWeek - 7*(int)(TempWeek / 7);
    return (TempWeek);
}


/*****************************************************************************/
/* File Name    : user_TIMER.c                                               */
/* Description  : This file implements the customer's TIMER unction.         */
/*****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Function Name: TIMER_Init                                                 */
/* Description  : TIMER Initialization Subroutines.                          */
/*---------------------------------------------------------------------------*/
void TIMER_Init(void)
{
//------Initial TIMER B for PWM-----------------
    DrvTMBC_Clk_Source( E_HS_CK, 0 );                       //set TMB Clock from HS_CK/1.
```

```
    DrvTMB_Open( E_TMB_MODE0, E_TMB_NORMAL, PWM_4KHZ_TIME );    //set TMB working on normal mode & TMB
clock int period= PWM_4KHZ_TIME/HS_CK= 250uS.
}


/*****************************************************************************/
/* File Name    : user_LCD.c                                                 */
/* Description  : This file implements the customer's LCD function.          */
/*****************************************************************************/

/*---------------------------------------------------------------------------*/
/* Function Name: LCD_Init                                                   */
/* Description  : LCD Initialization Subroutines                             */
/*---------------------------------------------------------------------------*/
void LCD_Init(void)
{
    clk_10 = 0xFF00FF42;                    //Set LCK= (LS_CK/8)/9/1= (32768/8)/9/1= 455Hz & charge pump
clock = LS_CK/1.
    DrvLCD_VLCDMode( E_VLCD27 );
    DrvLCD_VLCDTrim( 3 );                    //Trim VLCD as 2.93V.
    DrvLCD_LcdDuty ( E_LCD_DUTY4 );           //enable LCD@1/4 duty.

    DrvLCD_IOMode( 0, 0xFF );                //Set PT6(SEG2 ~SEG9 ) as LCD mode.
    DrvLCD_IOMode( 1, 0x7F );                //Set PT7(SEG10~SEG16) as LCD mode.
    DrvLCD_IOMode( 5, 0xFF );                //Set SEG1/0 as LCD mode.
    DrvLCD_LCDBuffer( ENABLE );              //enable LCD working.
    DrvLCD_DisplayMode( E_LCD_NORMAL );       //LCD at normal mode.
    LCD_RAM_Clear();
}

/*---------------------------------------------------------------------------*/
/* Function Name: LCD_RAM_Clear                                              */
/* Description  : LCD display RAM clear.                                     */
/*---------------------------------------------------------------------------*/
void LCD_RAM_Clear(void)
{
    int i=0;
    for(i=0; i< SEG_BUF_SIZE; i++)
    {
        DisplayBuffer[i] = 0;
    }
}

/*---------------------------------------------------------------------------*/
/* Function Name: On_Mode_Display                                            */
/* Description  : Current Mode display updated                               */
/*---------------------------------------------------------------------------*/
void On_Mode_Display(void)
{
    unsigned char Dig5_buf, Dig4_buf, Dig3_buf, Dig2_buf, Dig1_buf, Dig0_buf;

    if ( UserFlag0.b_DspGo_Flag == FALSE )
        return;
    UserFlag0.b_DspGo_Flag = FALSE;

//------Stop Watch display----------------------
    if ( Mode_Index == STOP_WATCH_MODE )
    {
        Dig5_buf = code_seg[ (STW_Min/10) % 10 ];
        Dig4_buf = code_seg[ (STW_Min) % 10 ];
        Dig3_buf = code_seg[ (STW_Sec/10) % 10 ];
        Dig2_buf = code_seg[ (STW_Sec) % 10 ];
        Dig1_buf = code_seg[ (STW_100/1000000) % 10 ];
        Dig0_buf = code_seg[ (STW_100/100000) % 10 ];
    }

//------Alarm Time display----------------------
    if ( Mode_Index == ALARM_TIME_MODE )
    {
        DrvRTC_Read( DRVRTC_ALARM_TIME, &sCurTime);
        Dig5_buf = 0;
        Dig4_buf = 0;
        Dig3_buf = code_seg[ (sCurAlarm.u32cHour/10) % 10 ];
        Dig2_buf = code_seg[ (sCurAlarm.u32cHour) % 10 ];
        Dig1_buf = code_seg[ (sCurAlarm.u32cMinute/10) % 10 ];
        Dig0_buf = code_seg[ (sCurAlarm.u32cMinute) % 10 ];
```

```
        if( UserFlag1.b_SetEI_Flag == TRUE )
        {
            if ( (SetItem_Index == 0) && (Tick500mS_Cnt > (5000000u/78125u)) )
            {
                if ( (QAdjTime_Cnt == 0) || ((QAdjTime_Cnt != 0) && (Key_Code_No == S2_KEY_NO)) )
                {
                    Dig3_buf = 0;
                    Dig2_buf = 0;
                }
            }
            if ( (SetItem_Index == 1)  && (Tick500mS_Cnt > (5000000u/78125u)) )
            {
                if ( (QAdjTime_Cnt == 0) || ((QAdjTime_Cnt != 0) && (Key_Code_No == S2_KEY_NO)) )
                {
                    Dig1_buf = 0;
                    Dig0_buf = 0;
                }
            }
            if ( SetItem_Index == 2 )
            {
                Dig5_buf = Char_A;
                Dig4_buf = Char_L;
                Dig3_buf = 0;
                Dig2_buf = 0;
                Dig1_buf = 0;
                Dig0_buf = 0;
                if ( (Tick500mS_Cnt < (5000000u/78125u)) && (UserFlag0.b_AlmEI_Flag == TRUE) )
                {
                    Dig2_buf = Char_o;
                    Dig1_buf = Char_n;
                }
                else if ( (Tick500mS_Cnt < (5000000u/78125u)) && (UserFlag0.b_AlmEI_Flag == FALSE) )
                {
                    Dig2_buf = Char_o;
                    Dig1_buf = Char_F;
                    Dig0_buf = Char_F;
                }
            }
        }
    }

//------Time & Calendar Display-----------------
    if ( (Mode_Index == RTC_TIME_MODE) || (Mode_Index == CALENDAR_MODE) )
    {
        DrvRTC_Read( DRVRTC_CURRENT_TIME, &sCurTime );

        if ( Mode_Index == RTC_TIME_MODE )
        {
            Dig5_buf = code_seg[ (sCurTime.u32cHour/10) % 10 ];
            Dig4_buf = code_seg[ (sCurTime.u32cHour) % 10 ];
            Dig3_buf = code_seg[ (sCurTime.u32cMinute/10) % 10 ];
            Dig2_buf = code_seg[ (sCurTime.u32cMinute) % 10 ];
            Dig1_buf = code_seg[ (sCurTime.u32cSecond/10) % 10 ];
            Dig0_buf = code_seg[ (sCurTime.u32cSecond) % 10 ];
        }
        else
        {
            Dig5_buf = code_seg[ (sCurTime.u32Year/10) % 10 ];
            Dig4_buf = code_seg[ (sCurTime.u32Year) % 10 ];
            Dig3_buf = code_seg[ (sCurTime.u32cMonth/10) % 10 ];
            Dig2_buf = code_seg[ (sCurTime.u32cMonth) % 10 ];
            Dig1_buf = code_seg[ (sCurTime.u32cDay/10) % 10 ];
            Dig0_buf = code_seg[ (sCurTime.u32cDay) % 10 ];
        }

        if( (UserFlag1.b_SetEI_Flag == TRUE) && (Tick500mS_Cnt > (5000000u/78125u)) )
        {
            if ( (SetItem_Index == 0) && ( (QAdjTime_Cnt == 0) || ((QAdjTime_Cnt != 0) && (Key_Code_No ==
S2_KEY_NO)) ) )
            {
                Dig5_buf = 0;
                Dig4_buf = 0;
            }
            if ( (SetItem_Index == 1) && ( (QAdjTime_Cnt == 0) || ((QAdjTime_Cnt != 0) && (Key_Code_No ==
S2_KEY_NO)) ) )
```

```
            {
                Dig3_buf = 0;
                Dig2_buf = 0;
            }
            if ( (SetItem_Index == 2) && ( (QAdjTime_Cnt == 0) || ((QAdjTime_Cnt != 0) && (Key_Code_No ==
S2_KEY_NO)) ) )
            {
                Dig1_buf = 0;
                Dig0_buf = 0;
            }
        }
    }

    //Alarm icon display
    DisplayBuffer[7] = 0;
    if ( UserFlag0.b_AlmEI_Flag == TRUE )
    {
        if ( ! ((UserFlag1.b_AlmGo_Flag == TRUE) && (Tick500mS_Cnt < (5000000u/78125u) )) )
            DisplayBuffer[7] = ALARM_LCD_SIGN;
    }

    DisplayBuffer[0] = Dig5_buf;
    DisplayBuffer[1] = Dig4_buf;
    if ( !(Mode_Index == ALARM_TIME_MODE) )
        DisplayBuffer[1] |= seg_h;                              //point display.
    DisplayBuffer[2] = Dig3_buf;
    DisplayBuffer[3] = Dig2_buf;
    if ( !( (Mode_Index == ALARM_TIME_MODE) && (UserFlag1.b_SetEI_Flag == TRUE) && (SetItem_Index == 2) ) )
        DisplayBuffer[3] |= seg_h;                              //point display.
    DisplayBuffer[4] = Dig1_buf;
    DisplayBuffer[5] = Dig0_buf;
    DisplayBuffer[8] = Mode_Index;
    RAM2LCD( DisplayBuffer );                                   //Update LCD display RAM.
}

/*-------------------------------------------------------------------------------*/
/* Function Name: LCD_Display_HYcon                                              */
/*-------------------------------------------------------------------------------*/
void LCD_Display_HYcon(void)
{
    DisplayBuffer[1] = Char_H;
    DisplayBuffer[2] = Char_Y;
    DisplayBuffer[3] = Char_c;
    DisplayBuffer[4] = Char_o;
    DisplayBuffer[5] = Char_n;
    RAM2LCD( DisplayBuffer );                                   //Update LCD display RAM.
}

/*-------------------------------------------------------------------------------*/
/* Function Name: RAM2LCD                                                        */
/* Description  : RAM buffer data transfer to LCD RAM                            */
/*-------------------------------------------------------------------------------*/
void RAM2LCD( unsigned char *Buffer_Adr )
{
    unsigned char buf_idx;

    for(buf_idx=0; buf_idx<SEG_BUF_SIZE; buf_idx++)
    {
        DrvLCD_WriteData ( buf_idx, *Buffer_Adr );
        Buffer_Adr++;
    }
}

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

7.2. 附加檔案

APD-HY16F023_DemoCode_V01.zip

8. 參考文獻

[1] http://www.hycontek.com/attachments/MSP/APD-HY16F009_TC.pdf
紘康科技HY16F188 RTC時鐘萬年曆
[2] http://www.hycontek.com/attachments/MSP/DS-HY16F198_TC.pdf
紘康科技HY16F198 Datasheet.
[3] http://www.hycontek.com/attachments/MSP/UG-HY16F198_TC.pdf
紘康科技 HY16F198 User Guide.

9. 修訂記錄

以下描述本檔差異較大的地方，而標點符號與字形的改變不在此描述範圍。

日期	文件版次	頁次	摘要
2016/07/06	V1.0	ALL	初版發行