



---

RTC 時鐘萬年曆\_應用說明書

HY16F188

RTC Clock Calendar

## 目 錄

1. 內容簡介 .....	4
2. 原理說明 .....	4
2.1 控制原理 .....	4
2.2 控制晶片 .....	5
3. 系統設計 .....	6
3.1 硬體說明 .....	6
3.2 功能說明 .....	8
4. 操作流程 .....	9
4.1 程式流程 (01) .....	9
4.2 程式流程 (02) .....	10
4.3 程式流程 (03) .....	11
5. 技術規格 .....	12
6. 結果總結 .....	12
7. 附加檔案 .....	12
8. 參考文獻 .....	12
9. 修訂紀錄 .....	12
附件: 範例應用程式 .....	13

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。

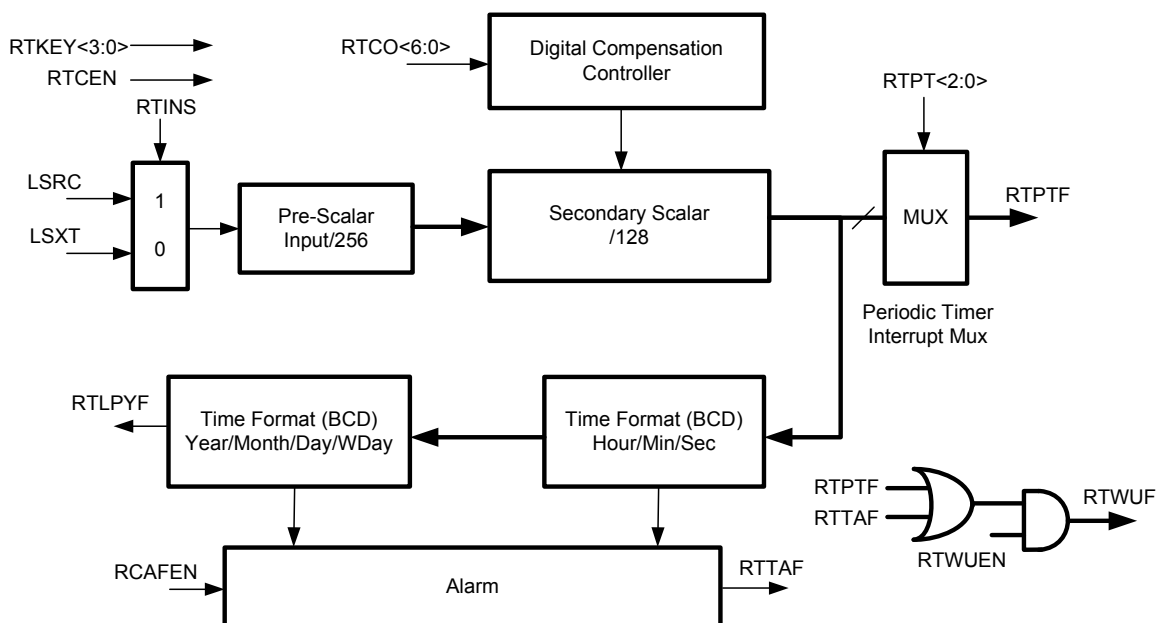
### 1.內容簡介

本文主要是介紹 HYCON HY16F Series 晶片在時鐘及萬年曆的應用。由於 HY16F188 晶片內部包含 Real Time Clock(RTC)，再配合外部 32768Hz 的振盪器及 LCD 驅動 IC HY2613B。使 HY16F188 用於時鐘萬年曆的應用時週邊非常簡潔。HY16F188 相較其他 MCU 能減少電流的消耗。

### 2.原理說明

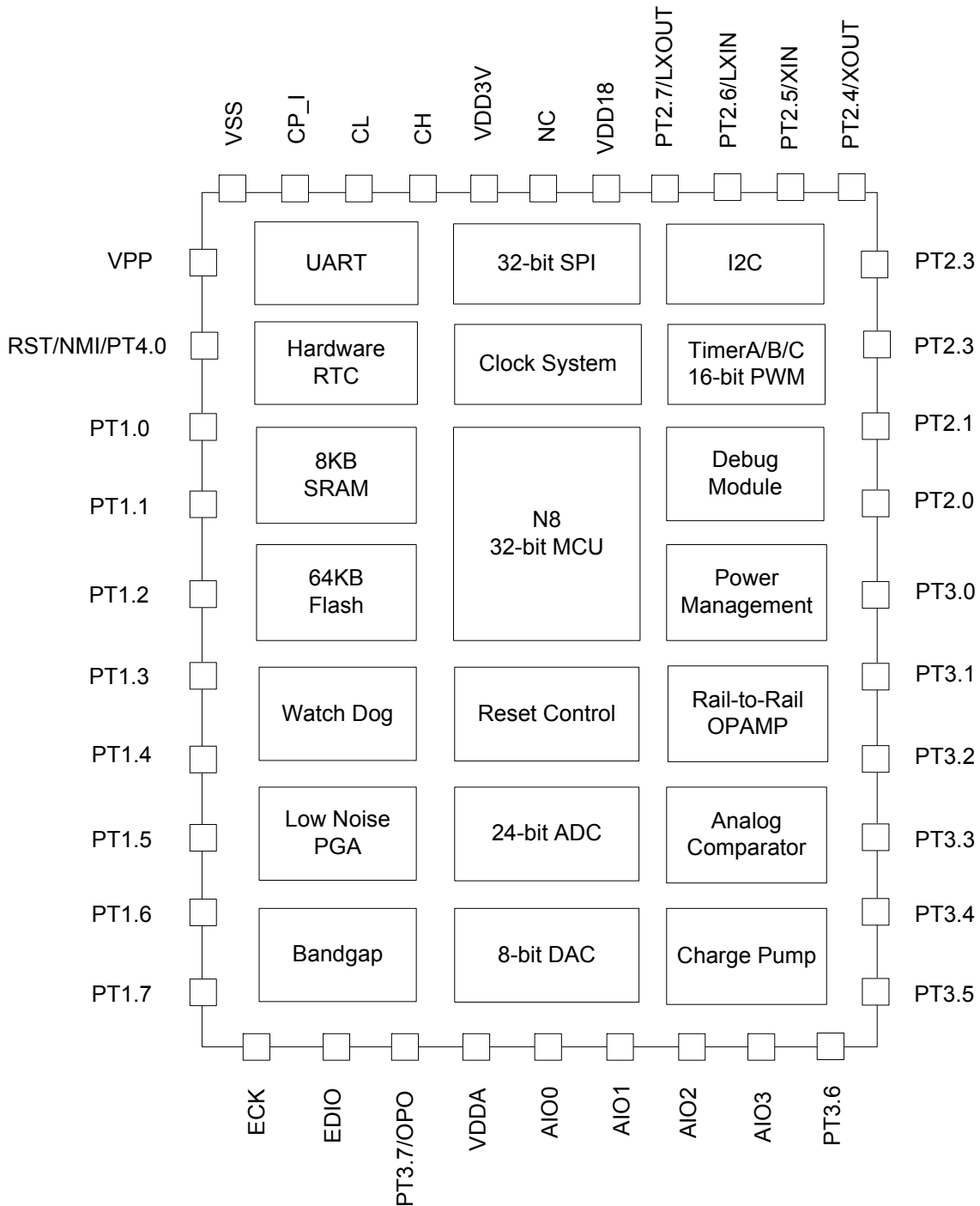
#### 2.1 控制原理

HY16F188 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 HY16F188 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(HY16F188)



紘康 HY16F 系列 32 位元高性能 Flash 單片機(HY16F188)

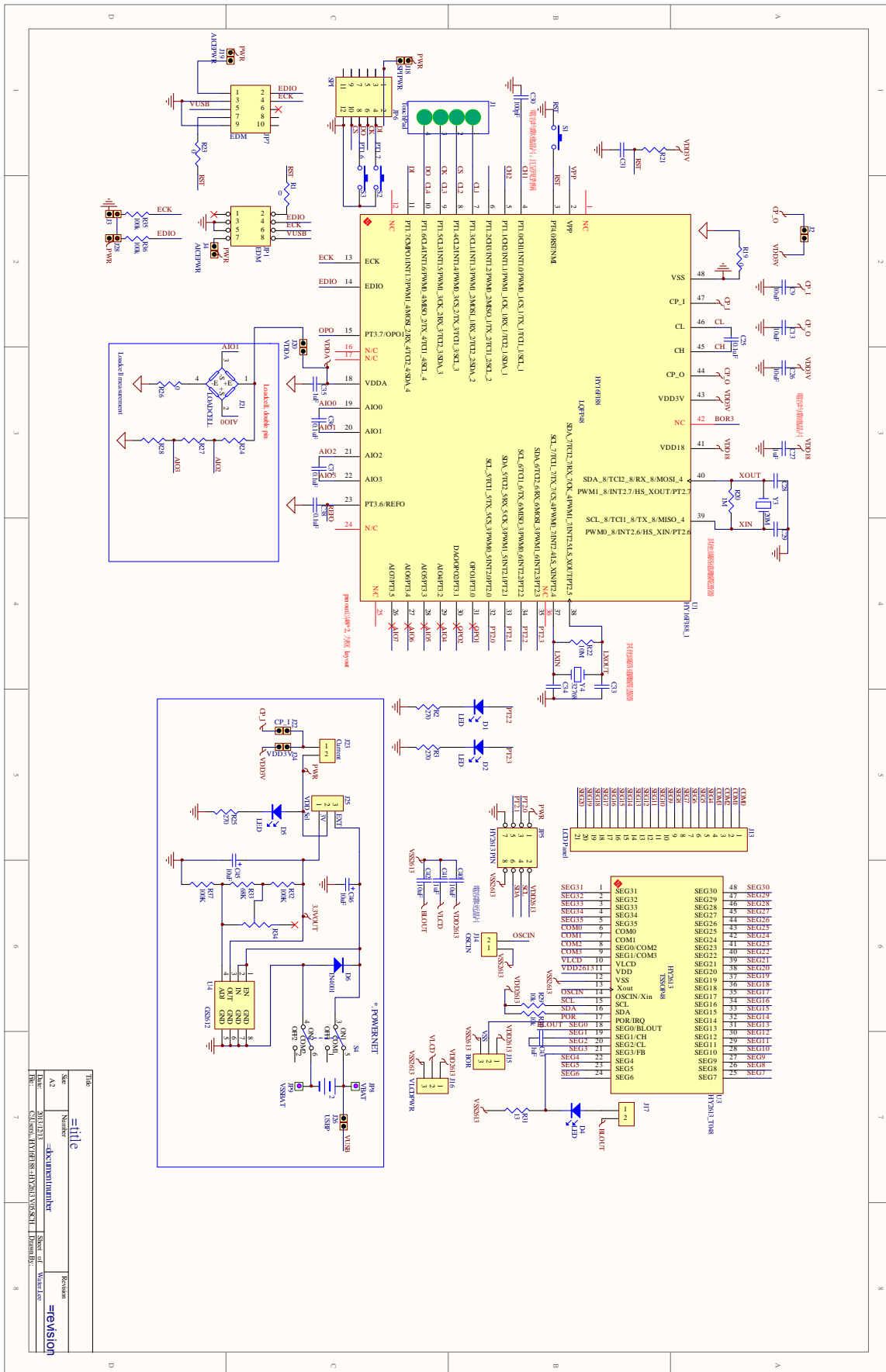
- (1)採用最新 Andes 32 位元 CPU 核心 N801 處理器。
- (2)電壓操作範圍 2.4~3.6V，以及-40°C~85°C工作溫度範圍。
- (3)支援外部 20MHz 石英震盪器或內部 20MHz 高精度 RC 震盪器，  
擁有多種 CPU 工作時脈切換選擇，可讓使用者達到最佳省電規劃。
- (3.1)運行模式 350uA@2MHz/2(3.2)待機模式 10uA@32KHz/2(3.3)休眠模式 2.5uA
- (4)程式記憶體 64KBytes Flash ROM
- (5)資料記憶體 08KBytes SRAM。
- (6)擁有 BOR and WDT 功能，可防止 CPU 死機。
- (7)24-bit 高精準度  $\Sigma\Delta$ ADC 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 模組具 PWM 波形產生功能
- (11)16-bit Timer C 模組具數位 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

### 3.系統設計

#### 3.1 硬體說明

HY16F188 對於時鐘及萬年曆應用，整體電路包含外部震盪器，S2、S3、LED 按鈕部分及 LCD 顯示，整體電路如下圖所示。

# HY16F188 RTC 時鐘萬年曆應用說明書



RTC 時鐘及萬年曆整體電路

### 3.2 功能說明

本程式包含了：

時鐘、萬年曆、碼錶、鬧鐘等功能，配合外部 LCD 面板顯示及兩個按鈕進行控制。

電源開啓後，所顯示的直接為內部所設定的時間依序為時/分/秒。

在時間顯示時，按下 S3 後進入碼表模式；

在碼錶模式下。S3 為開始計時/停止計時；S2 為歸零/離開碼表模式。

在時間顯示時，按下 S2 後進入萬年曆模式；

萬年曆模所顯示的為內部所設定之日期，首先顯示為年。

在萬年曆模式下 S2 為顯示月/日/星期的切換或者離開萬年曆模式；S3 則進入設定模式。

預設定時間、日期等於，萬年曆模式顯示”月/日/星期”時按下 S3

預設定鬧鐘時間等於，萬年曆模式顯示”年”時按下 S3

在設定模式下，S3 代表數值+1、S2 代表確定。

設定時間：依序給予修改的內容為年/月/日/星期/時/分/秒。

設定鬧鐘：依序給予修改的內容為時/分/秒。

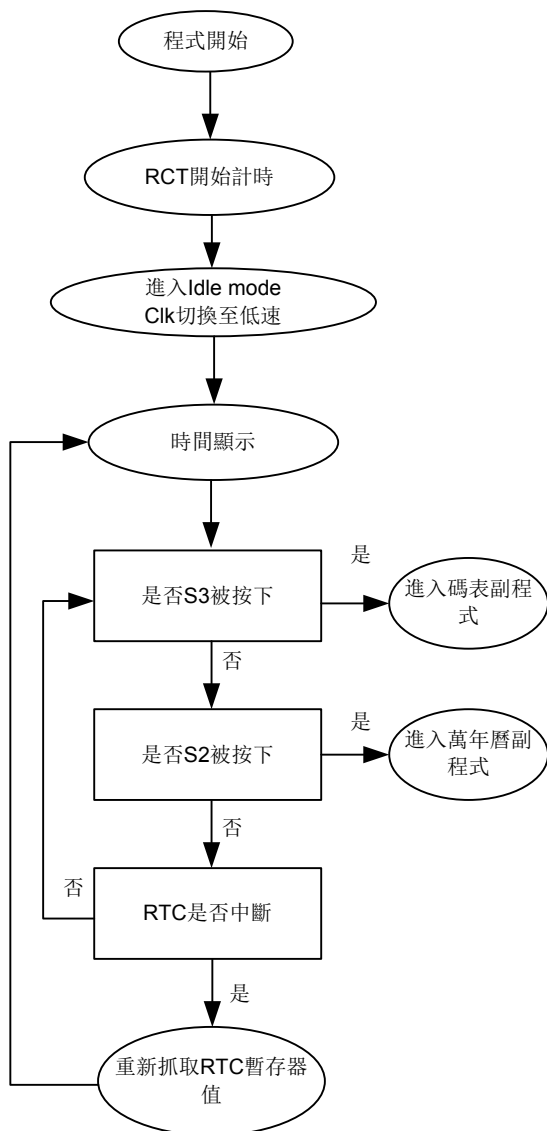
當修改為秒時，在次按下 S2 則會離開設定模式，顯示剛剛所設定之時間。



## 4. 操作流程

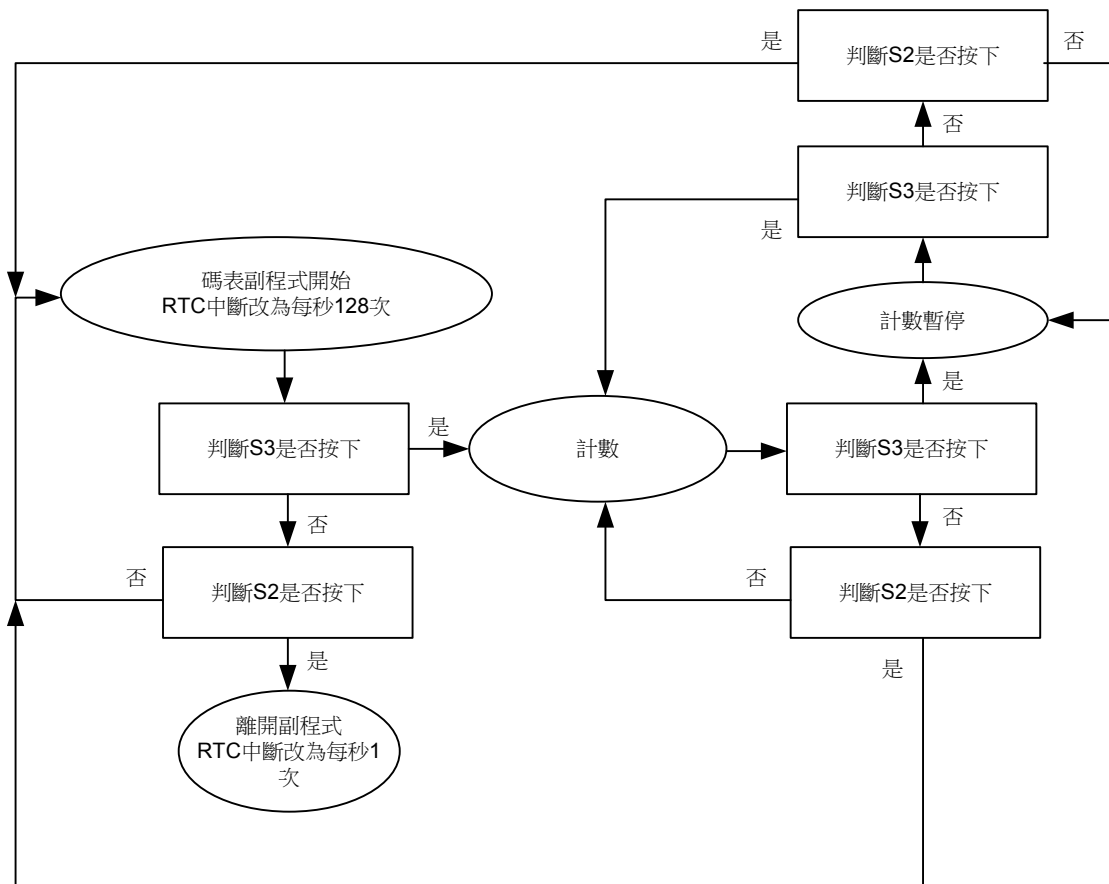
### 4.1 程式流程(01)

#### 主程式流程



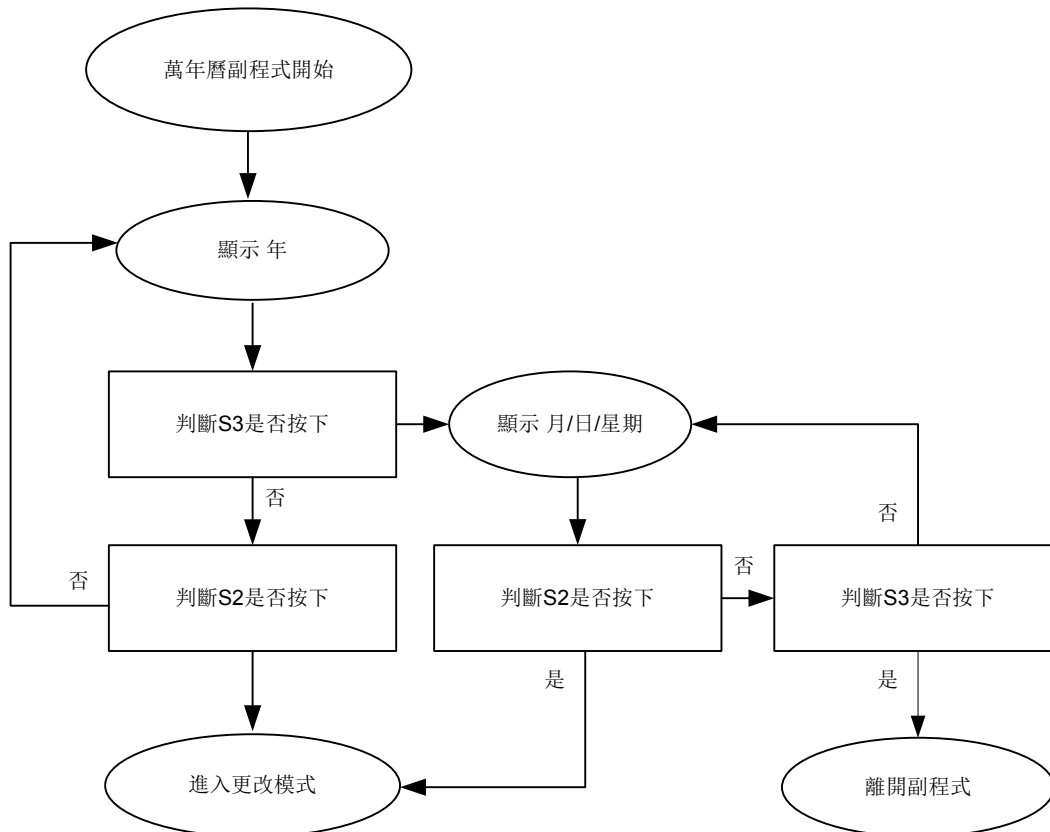
4.2 程式流程(02)

碼表副程式



4.3 程式流程(03)

萬年曆副程式



### 5.技術規格

- (1)電源接點：3.6V
- (2)功耗：省電模式:MCU 8uA      LCD13.5uA ；  
          操作模式:MCU 670uA    LCD280.4uA ；
- (3)適用範圍：各種萬年曆、碼表、鬧鐘；
- (4)工作溫度：-40℃ ~ +85℃ ；
- (5)存貯溫度：-55℃ ~ +125℃ ；
- (6)相對濕度：<95%（20±5℃條件）

### 6.結果總結

以 HY16F188 為主控結合內部高精度 RTC 模式。配合外部 32768 晶體震盪器，可以達到高準度低耗電的時鐘及萬年曆。不需外不添加 IC 即可用 MCU 本身完成高準度的計時。相較於無內建 RTC 之 MCU，HY16F188 不論在時間計算的精確度、整體電路的大小及耗電流的多寡，都有相當出色的表現。

### 7.附加檔案



HY16F009V03.rar

### 8.參考文獻

- (1)HYCON HY16F188 Series Data Sheet
- (2)HYCON HY16F188 Series User's Guide

### 9.修訂紀錄

以下描述本檔差異較大的地方，而標點符號與字形的改變不在此描述範圍。

版本	頁次	變更摘要	日期
V1.0	ALL	初版發行	2013/10/30
V2.0	ALL	更改腳位名稱 VDD->VDD18	2013/12/13
V3.0	ALL	更新程序，修改按鍵名稱	2014/12/17

附件:範例應用程式

```
/*-----*/
*HY16F188
* main.c
* Created on:2013/12/08
* Maintenance:2014/10/01
* -----
* Release 1.0.0
* Program Description:
* -----
*
* -----
*
* | -----
* PT2.0 | SDA ---> SDA | LCD Drive HY2613 |
* PT2.1 | SCK ---> SCK -----
* GND |
* |
* -----
*-----*/
/*-----*/
/* Includes */
/*-----*/
#include "HY16F188.h"
#include "System.h"
#include "DrvGPIO.h"
#include "DrvI2C.h"
#include "DrvHWI2C.h"
#include "DrvCLOCK.h"
#include "HY2613.h"
#include "my define.h"
#include "DrvREG32.h"
#include "DrvTimer.h"
#include "DrvRTC.h"

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

```
typedef union _MCUSTATUS
```

```
{  
    char _byte;  
    struct  
    {  
        unsigned b_ADCdone:1;  
        unsigned b_TMAdone:1;  
        unsigned b_TMBdone:1;  
        unsigned b_TMC0done:1;  
        unsigned b_TMC1done:1;  
        unsigned b_RTCdone:1;  
        unsigned b_UART_TxDone:1;  
        unsigned b_UART_RxDone:1;  
    };  
} MCUSTATUS;
```

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

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

```
MCUSTATUS MCUSTATUSbits;  
extern unsigned char seg[16];  
#define Disable 0  
#define Enable 1  
unsigned int sec,min,hour,week,day,month,year; // 丁ノ  
unsigned int b,c,pb6,pb7; // 齡ノ  
unsigned int mm,ss,sss,stopsss,stopss,stopmm; // 纒ノ  
unsigned int setsec,setmin,sethour,setweek,setday,setmonth,setyear,setnum; //  
unsigned int act,acs,acm,ach,ac;  
unsigned int first;
```

```
/*-----*/  
/* Function PROTOTYPES */  
/*-----*/
```

```
void ClearLCDframe(void);  
void LCD_DATA_DISPLAY_1(unsigned int LcdBuffer);
```

```
void LCD_DATA_DISPLAY_2(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY_3(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY_4(unsigned int LcdBuffer);
void LCD_DATA_DISPLAY_5(unsigned int LcdBuffer);
void Delay(unsigned int num);
void LCD_DISPLAY_HYCON(void);
void LCD_DISPLAY_PASS01(void);
void Initial_Array(void);
void RTC_Initial(void);
void Hc(void);
void loopday(void);
void Pb_7();
void Pb_6();
void timeloop(void);
void lookac(void);
void settime(void);
void setac(void);

/*-----*/
/* Main Function */
/*-----*/
int main(void)
{
    //unsigned int m;

    DrvCLOCK_EnableHighOSC(E_INTERNAL,50);    // Select HAO 4MHz
    InitalI2C();
    SYS_EnableGIE(7,0x3F); // Enable GIE(Global Interrupt)

    Ini_Display();
    ClearLCDframe();
    Delay(10000);
    DisplayHYcon();
    Delay(50000);
    MCUSTATUSbits._byte = 0;
    LCD_DATA_DISPLAY(123456);
    int_04=0xff20ff00;
    RTC_Initial();
}
```

```
pio_00=0xff00ff00;
pio_04=0xff00ff00;
pio_08=0x00;
pio_10=0xff00ff03;
pio_14=0xff00ff03;
```

```
DrvGPIO_Open(E_PT1,0xc0,E_IO_INPUT);
DrvGPIO_Open(E_PT1,0xc0,E_IO_PullHigh); //enable PT1_5-7 pull high R
DrvGPIO_Open(E_PT1,0x10,E_IO_OUTPUT);
```

```
SYS_EnableGIE(7,0X3F); //Enable GIE (Global Interrupt Enable)
DrvGPIO_ClearIntFlag(E_PT1,0xc0); //clear PT1.4~7 interrupt flag
DrvCLOCK_EnableHighOSC(E_INTERNAL,50); //select hsrc 2MHz
DrvGPIO_ClkGenerator(E_HS_CK,1); //GPIO CLK enable
DrvGPIO_Open(E_PT1,0xc0,E_IO_IntEnable); //PT1.4~7 interrupt enable
DrvGPIO_IntTrigger(E_PT1,0xc0,E_N_Edge); //PT1.4~7 interrupt trigger method is negative edge
```

```
DrvGPIO_Open(E_PT1,0x10,E_IO_OUTPUT);
```

```
S_DRVRTC_TIME_DATA_T sCurTime;//read
DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
sec=sCurTime.u32cSecond;
min=sCurTime.u32cMinute;
hour=sCurTime.u32cHour;
week=sCurTime.u32cDayOfWeek;
day=sCurTime.u32cDay;
month=sCurTime.u32cMonth;
year=sCurTime.u32Year;
asm("NOP");
```

```
LCD_DATA_DISPLAY_1(hour);
LCD_DATA_DISPLAY_2(min);
LCD_DATA_DISPLAY_3(sec);
```

```
ac=0;
acs=00;
```



ach=16;

acm=05;

while(1)

{

asm volatile("sethi \$r0, 0xc0000");

asm volatile("ori \$r0, \$r0, 0x003f");

asm volatile("mtr \$r0, \$INT\_MASK");

asm volatile("movi \$r0, 0x70009");

asm volatile("mtr \$r0, \$PSW");

clk\_08=0x0101;

clk\_00=0x0100;

sys\_04=0x1010;

asm("standby 1");

lookac();

if(b==0) Pb\_6();

if(pb6==1)

{

pb6=0;

clk\_00=0x0f05;

clk\_08=0x0f00;

DrvRTC\_PeriodicTimeEnable(0);//set 1/128

Hc();

DrvRTC\_PeriodicTimeEnable(7);//set 1/128

clk\_08=0x0101;

clk\_00=0x0100;

}

if(c==0) Pb\_7();

if(pb7==1)

{

pb7=0;

clk\_00=0x0f05;

clk\_08=0x0f00;

loopday();

clk\_08=0x0101;

clk\_00=0x0100;

}

if(sss>=1)

```
    {
        sss=0;
        timeloop();
    }
}

return 0;
}

/*-----*/
/* Hardware Communication Interrupt */
/* UART/SPI/I2C Interrupt Service Routines */
/*-----*/
void HW0_ISR(void)
{
    unsigned char I2C_Status;

    if(DrvI2C_ReadIntFlag()==E_DRVI2C_INT) // Get I2C Interrupt Flag
    {
        I2C_Status=DrvI2C_GetStatusFlag(); // Get I2C Status Flag
        switch(I2C_Status)
        {
            case 0x90: //MACTFlag+RWFlag
                { /* START has been transmitted */
                    DrvI2C_WriteData(I2C_TARGET); //Send Slave Address & R/W Bit
                    DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag
                    break;
                };
            case 0x84: //MACTFlag+ACKFlag
                { /* Slave A + W has been transmitted. ACK has been received. */
                    DrvI2C_WriteData(Sendbuf[DataTxIndex++]); //Send Data to Slave
                    DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag
                    break;
                };
            case 0x80: //MACTFlag
                { /* Slave A + W has been transmitted. ACK has been received. */
                    DrvI2C_WriteData(Sendbuf[DataTxIndex++]); //Send Data to Slave
                    DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag
                }
            }
        }
    }
}
```

```
        break;
    };
case 0x30:
    {
        DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag
        EndFlag=1;
        break;
    };
case 0x8C: // MACTFlag+DFFlag+ACKFlag
    { /* DATA has been transmitted and ACK has been received */
        if(DataTxIndex<DataTxLen)
        {
            DrvI2C_WriteData(Sendbuf[DataTxIndex++]); //Send Data to Slave
            DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag
        }
        else
        {
            if(I2C_RW == WRITE)
            {
                DrvI2C_Ctrl(0,1,0,0); // I2C as master sends STOP signal
                EndFlag=1;
            }
            else if(I2C_RW == READ)
                DrvI2C_Ctrl(1,0,0,0); // I2C as master sends START signal
            DataTxIndex=0;
        }
        break;
    };
case 0x88: //MACTFlag+DFFlag
    { /* DATA has been transmitted and NACK has been received */
        DrvI2C_Ctrl(0,1,0,0); // I2C as master sends STOP signal
        DataTxIndex=0;
        EndFlag=1;
        break;
    };
case 0xB0:
    { /* A repeated START has been transmitted. */
        DrvI2C_WriteData(I2C_TARGET | READ); //Send Slave Address & R/W Bit
        DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag
```

```
        break;
    }
case 0x94: //MACTFlag+RWFlag
    { /* Slave A + R has been transmitted. ACK has been received. */
        DrvI2C_Ctrl(0,0,0,1); // Set ACK bit
        break;
    };
case 0x9C: //MACTFlag+RWFlag+DFFlag+ACKFlag
    { /* Data byte has been received. ACK has been transmitted. */
        if(DataRxLen>DataRxIndex)
        {
            Recbuf[DataRxIndex++]=DrvI2C_ReadData();
            DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag
        }
        else
        {
            Recbuf[DataRxIndex++]=DrvI2C_ReadData();
            DrvI2C_Ctrl(0,1,0,0); // I2C as master sends STOP signal
            EndFlag=1;
        }
        break;
    };
case 0x98: //MACTFlag+RWFlag+DFFlag
    { /* Data byte has been received. NACK has been transmitted. */
        DrvI2C_Ctrl(0,1,0,0); // I2C as master sends STOP signal
        EndFlag=1;
        break;
    };
default:
    {
        DrvI2C_Ctrl(0,0,0,0); // Clear all I2C flag
        EndFlag=1;
        break;
    };
}
DrvI2C_ClearEIRQ();
DrvI2C_ClearIntFlag(2); // Clear I2C Interrupt Flag(I2CIF)
SYS_EnableGIE(7,0x3F); // Enable GIE(Global Interrupt)
}
```

```
if(DrvI2C_ReadIntFlag()==E_DRVI2C_ERROR_INT) //Get I2C Error Interrupt Flag
{
    EndFlag=1;
    DrvI2C_ClearIRQ();
    DrvI2C_ClearIntFlag(2); // Clear I2C Interrupt Flag(I2CEIF)
    DrvI2C_Ctrl(0,1,0,0); // I2C as master sends STOP signal
    SYS_EnableGIE(7,0x3F); // Enable GIE(Global Interrupt)
}
}

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

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

void RTC_Initial(void)
{
    clk_00=0x0404;
    //RTC CLK;
    clk_08=0x8080ff00;
    //RTC KEY;
    rtc_00=0xff60ff00;

    DrvRTC_WriteEnable();
    Delay(0x01);
    //DrvRTC_ClockSource(E_INTERNAL_CLOCK);//35KHz
    DrvRTC_ClockSource(E_EXTERNAL_CLOCK);//32768Hz
}
```

```
DrvRTC_PeriodicTimeEnable(7);//set 1/1
DrvRTC_Enable();
DrvRTC_HourFormat(0);

Delay(0x01);
S_DRVRTC_TIME_DATA_T sCurTime;//setting start
DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
sCurTime.u8cClockDisplay=0;
sCurTime.u8cAmPm=0;
sCurTime.u32cSecond=0;
sCurTime.u32cMinute=15;
sCurTime.u32cHour=18;
sCurTime.u32cDayOfWeek=1;
sCurTime.u32cDay=4;
sCurTime.u32cMonth=11;
sCurTime.u32Year=2013;
sCurTime.u8IsEnableWakeUp=0;
DrvRTC_Write(DRVRTC_CURRENT_TIME,&sCurTime);
Delay(0x01);

asm volatile("sethi $r0, 0xc0000");
asm volatile("ori  $r0, $r0, 0x003f");
asm volatile("mtr $r0, $INT_MASK");
asm volatile("movi $r0, 0x70009");
asm volatile("mtr $r0, $PSW");

}

/*-----*/
void HW1_ISR(void)
{
    int_04=0xff20ff00;
    rtc_00=0xff60ff31;
    sss++;
    asm volatile("sethi $r0, 0xc0000");
    asm volatile("ori  $r0, $r0, 0x003f");
    asm volatile("mtr $r0, $INT_MASK");
    asm volatile("movi $r0, 0x70009");
}
```

```
asm volatile("mstr $r0, $PSW");
}
/*-----*/
void HW4_ISR(void)
{
    DrvGPIO_ClearIntFlag(E_PT1,0xc0);           //Clear PT1 interrupt flag
    b=DrvGPIO_GetBit(E_PT1,6);
    c=DrvGPIO_GetBit(E_PT1,7);

    asm volatile("sethi $r0, 0xc0000");
    asm volatile("ori  $r0, $r0, 0x003f");
    asm volatile("mstr $r0, $INT_MASK");
    asm volatile("movi $r0, 0x70009");
    asm volatile("mstr $r0, $PSW");
}
/*-----*/
void timeloop(void)
{
    S_DRVRTC_TIME_DATA_T sCurTime;//read
    DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
    sec=sCurTime.u32cSecond;
    min=sCurTime.u32cMinute;
    hour=sCurTime.u32cHour;
    week=sCurTime.u32cDayOfWeek;
    day=sCurTime.u32cDay;
    month=sCurTime.u32cMonth;
    year=sCurTime.u32Year;
    clk_00=0x0f05;
    clk_08=0x0f00;
    LCD_DATA_DISPLAY_1(hour);
    LCD_DATA_DISPLAY_2(min);
    LCD_DATA_DISPLAY_3(sec);
    clk_08=0x0303;
    clk_00=0x0100;
    asm volatile("sethi $r0, 0xc0000");
    asm volatile("ori  $r0, $r0, 0x003f");
    asm volatile("mstr $r0, $INT_MASK");
    asm volatile("movi $r0, 0x70009");
    asm volatile("mstr $r0, $PSW");
}
```

```
}
/*-----*/
void Hc(void)
{
    while(1)
    {
zero:  sss=0;
        ss=0;
        mm=0;
        LCD_DATA_DISPLAY_3(sss);
        LCD_DATA_DISPLAY_2(ss);
        LCD_DATA_DISPLAY_1(mm);
        //b=DrvGPIO_GetBit(E_PT1,6);
        //c=DrvGPIO_GetBit(E_PT1,7);
        if(c==0) Pb_7();
        if(pb7==1)
        {
            pb7=0;
            break;
        }
        if(b==0) Pb_6();
        if(pb6==1)
        {
            pb6=0;
            sss=0;
            while(1)
            {
                LCD_DATA_DISPLAY_3(sss);
                LCD_DATA_DISPLAY_2(ss);
                LCD_DATA_DISPLAY_1(mm);
                // b=DrvGPIO_GetBit(E_PT1,6);
                // c=DrvGPIO_GetBit(E_PT1,7);
                if(sss>=128)
                {
                    sss=0;
                    ss++;
                    if(ss>=60)
                    {
                        ss=0;

```



```
        mm++;
    }
}
if(c==0) Pb_7();
if(pb7==1)
{
    pb7=0;
    goto zero;
}
if(b==0) Pb_6();
if(pb6==1)
{
    pb6=0;
    stopsss=sss;
    stopss=ss;
    stopmm=mm;
    Delay(0x300);
    while(1)
    {
        LCD_DATA_DISPLAY_3(stopsss);
        LCD_DATA_DISPLAY_2(stopss);
        LCD_DATA_DISPLAY_1(stopmm);
        // b=DrvGPIO_GetBit(E_PT1,6);
        // c=DrvGPIO_GetBit(E_PT1,7);
        if(c==0) Pb_7();
        if(pb7==1)
        {
            pb7=0;
            goto zero;
        }
        if(b==0) Pb_6();
        if(pb6==1)
        {
            pb6=0;
            sss=stopsss;
            ss=stopss;
            mm=stopmm;
            break;
        }
    }
}
```



```
        settime();
    }
}
}
if(b==0) Pb_6();
if(pb6==1)
{
    pb6=0;
    setac();
}
}
bb:  asm("NOP");
}
void setac(void)
{
    act=0;
    while(1)
    {
        b=DrvGPIO_GetBit(E_PT1,6);
        c=DrvGPIO_GetBit(E_PT1,7);
        LCD_DATA_DISPLAY_3(acs);
        LCD_DATA_DISPLAY_2(acm);
        LCD_DATA_DISPLAY_1(ach);
        if(c==0) Pb_7();
        if(pb7==1)
        {
            pb7=0;
            act++;
            if(act>=3) break;
        }
        if(b==0)
        {
            if(act==0)
            {
                acs++;
                if(acs>=60) acs=0;
            }
            if(act==1)
            {
```

```
        acm++;
        if(acm>=60) acm=0;
    }
    if(act==2)
    {
        ach++;
        if(ach>=24) ach=0;
    }
}
}
}
void settime(void)
{
    setyear=year;
    setmonth=month;
    setday=day;
    setweek=week;
    sethour=hour;
    setmin=min;
    setsec=sec;
    setnum=0;
    LCD_DATA_DISPLAY_4(setyear);
    while(1)
    {
        if(setnum==0) LCD_DATA_DISPLAY_4(setyear);
        else
        {
            if(setnum>=4)
            {
                LCD_DATA_DISPLAY_1(sethour);
                LCD_DATA_DISPLAY_2(setmin);
                LCD_DATA_DISPLAY_3(setsec);
            }
            if(setnum<=3)
            {
                LCD_DATA_DISPLAY_1(setmonth);
                LCD_DATA_DISPLAY_2(setday);
                LCD_DATA_DISPLAY_3(setweek);
            }
        }
    }
}
```

```
    }  
}  
b=DrvGPIO_GetBit(E_PT1,6);  
c=DrvGPIO_GetBit(E_PT1,7);  
if(b==0) Pb_6();  
if(pb6==1)  
{  
    pb6=0;  
    if(setnum==0)  
    {  
        setyear++;  
        if(setyear>=2100) setyear=2012;  
    }  
    if(setnum==1)  
    {  
        setmonth++;  
        if(setmonth>=13) setmonth=1;  
    }  
    if(setnum==2)  
    {  
        setday++;  
        if(setday>=32) setday=1;  
    }  
    if(setnum==3)  
    {  
        setweek++;  
        if(setweek>=8) setweek=1;  
    }  
    if(setnum==4)  
    {  
        sethour++;  
        if(sethour>=24) sethour=0;  
    }  
    if(setnum==5)  
    {  
        setmin++;  
        if(setmin>=60) setmin=0;  
    }  
    if(setnum==6)
```

```
        {
            setsec++;
            if(setsec>=60) setsec=0;
        }
    }
    if(c==0) Pb_7();
    if(pb7==1)
    {
        pb7=0;
        setnum++;
        if(setnum==7)
        {
            setnum=0;
            S_DRVRTC_TIME_DATA_T sCurTime;//setting start
            DrvRTC_Read(DRVRTC_CURRENT_TIME,&sCurTime);
            sCurTime.u32cSecond=setsec;
            sCurTime.u32cMinute=setmin;
            sCurTime.u32cHour=sethour;
            sCurTime.u32cDayOfWeek=setweek;
            sCurTime.u32cDay=setday;
            sCurTime.u32cMonth=setmonth;
            sCurTime.u32Year=setyear;
            DrvRTC_Write(DRVRTC_CURRENT_TIME,&sCurTime);
            break;
        }
    }
}

/*-----*/

/*-----*/
void Pb_7()
{
    if(c==0)
    {
        pb7=1;
        c=1;
    }
}
```

```
        while(1)
        {
            c=DrvGPIO_GetBit(E_PT1,7);
            if(c==1)break;
        }

    }
}
/*-----*/
void Pb_6()
{
    if(b==0)
    {
        pb6=1;
        b=1;
        while(1)
        {
            b=DrvGPIO_GetBit(E_PT1,6);
            if(b==1)break;
        }
    }
}
/*-----*/
void lookac(void)
{
    DrvGPIO_SetPortBits(E_PT1,0x00);
    if(hour==ach)
        if(min==acm)
            if(sec==acs) ac=1;
    if(ac==1)
    {
        DrvGPIO_SetPortBits(E_PT1,0x10);
        b=DrvGPIO_GetBit(E_PT1,6);
        c=DrvGPIO_GetBit(E_PT1,7);
        if(c==0) Pb_7();
        if(pb7==1)
        {
            pb7=0;

```

```
        ac=0;
    }
    if(b==0) Pb_6();
    if(pb6==1)
    {
        pb6=0;
        ac=0;
    }
}
}
/*-----*/
/* End Of File*/
/*-----*/
```