MSP430电子万年历

目　录

1. 系统总体设计 0
1.1 功能说明 0
1.2 任务分配情况 0
1.3 系统工作流程 0
2.硬件设计 1
2.1 MSP430F5438A芯片简介 1
2.2 矩阵键盘模块 1
2.2.1 矩阵键盘介绍 1
2.2.2 矩阵键盘实物图 1
2.2.3 矩阵键盘与 MSP430F5438A接口电路 2
2.3 液晶 12864 模块 2
2.3.1 液晶介绍 2
2.3.2 液晶与 MSP430F5438A接口电路 2
2.4 DS1302 实时时钟芯片模块 2
2.4.1 DS1302 实时时钟芯片简介 2
2.4.2 DS1302 实时时钟芯片实物图 3
2.4.3 DS1302 实时时钟芯片与 MSP430F5438A接口电路 3
2.4.4 SPI 协议简介 3
2.5 DS18B20 温度芯片模块 3
2.5.1 DS18B20 温度芯片简介 3
2.5.2 DS18B20 温度芯片实物图 4
2.5.3 DS18B20 温度芯片与 MSP430F5438A接口电路 4
2.5.4 单总线协议简介 4
3. 软件设计 5
3.1 系统总体设计 5
3.1.1 系统流程图 5
3.2 矩阵键盘模块 6
3.2.1 按键进入修改界面 6
3.2.2 按键选择修改内容 6
3.2.3 按键修改时间 7

3.3 液晶模块 8
3.4 DS1302 实时时钟芯片模块 9
3.4.1 DS1302 的初始化 9
3.4.2 DS1302 的读写 10
3.5 DS18B20 温度芯片模块 10
3.5.1 DS18B20 初始化 10
3.5.2 DS18B20 写操作 10
3.5.3 DS18B20 读操作 10
3.6 芯片值转化为显示值模块 11
4. 实验结果 12
4.1 整体图 12
4.2 运行过程 12
5. 缺陷与调试 12
5.1 调试过程 12
5.2 程序的缺陷 13
6. 实验心得 13
7. 附录 13

1.系统总体设计

1.1功能说明

本次课程设计的要求是制作一个电子万年历，要求在显示屏上显示年、月、日、时、分、秒、周、温度等信息，并且能够自行修改相关信息，且在修改信息时时钟停振。根据要求所设计的系统的总体框架如图 1 所示：

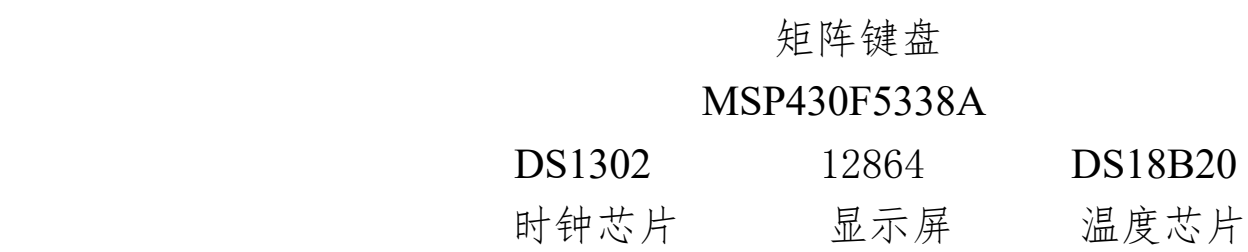

图 1 系统总体框架图

1、单片机最小系统

要求系统设计使用 Texas Instrument 公司的 MSP430F5438A 单片机作为系统的核心控制器。

2、时钟芯片 DS1302

DS1302 是 dallas 公司推出的一种高性能、定功耗的实时时钟芯片，内含一个实时时钟和 31 字节的静态 RAM，采用 SPI 三线接口与 CPU 通信。

3、温度芯片 DS18B20

DS18B20的数字温度计提供 9 位到 12 位的摄氏温度测量，并具有报警功能，DS18B20通过定义仅需要一个数据线（和地面）与中央微处理器通信。它的工作范围为-55°C至+ 125°C，在-10°C至+ 85°C的范围内精确到±0.5℃。

4、矩阵键盘

使用 4*3 的矩阵键盘，可以通过矩阵键盘得到 0~9 的数字也可以生成“*”和“#”，用户可通过显示屏与矩阵键盘对 DS1302 的时间进行自定义修改。

5、液晶屏 12864

液晶屏为 128*64 点阵屏幕，在液晶上显示时间和温度信息，用户可以通过显示屏查看时间和温度，并根据提示对 DS1302 内时间进行修改。

1.2任务分配情况

1.3系统工作流程

单片机开机后（第一次开机会对 DS1302 初始化赋初值），会在显示屏上显示从 DS1302 中读取的时间信息和从 DS18B20 中读出的温度信息，在不对矩阵键盘进行操作的情况下，系统正常读取时间和温度显示。当需要修改时间信息时，按对应键进入修改界面，进入修改后，DS1302 停止工作，通过矩阵键盘选择所要

修改的内容，修改完成后将修改值存入 DS1302，修改完成，使 DS1302 继续工作，退回显示时间和温度的界面。

2.硬件设计

此次课设课题中硬件部分的选取与设计主要包含以下模块：MSP430F5438A 单片机，液晶屏 12864，4*3 矩阵键盘，DS1302 时钟芯片，DS18B20 温度芯片，若干连接线。

2.1 MSP430F5438A 芯片简介

MSP430 系列单片机是美国德州仪器（TI）1996 年开始推向市场的一种 16 位超低功耗、具有精简指令集（RISC）的混合信号处理器（Mixed Signal Processor）。称之为混合信号处理器，是由于其针对实际应用需求，将多个不同功能的模拟电路、数字电路模块和微处理器集成在一个芯片上，以提供“单片机”解决方案。该系列单片机多应用于需要电池供电的便携式仪器仪表中。在超低功耗方面，其处理器功耗（1.8-3.6V,0.1-400uA）和口线输入漏电流（最大 50nA）在业界都是最低的，在运算性能上，指令速度可以达到 16MIPS，在开发工具上，支持最先进的 JTAG 调制，在系统整合方面，根据其不同产品，集成了许多功能模块，性能稳定，可靠性高。MSP430F5438A 主要性能参数：

（1）高达25MHz的 CPU速度。

（2）1.8-3.6V 工作电压。

（3）高达256KB的闪存。

（4）高达18KB的 RAM

（5）独特的 USB开发套件。

（6）超低功耗5us 之内快速从待机模式唤醒。

图 2 MSP430F5438A 单片机实物图

2.2矩阵键盘模块

2.2.1矩阵键盘介绍

矩阵键盘将键盘以矩阵形式排列，本次课设使用的是 4*3 矩阵键盘，4 行 3 列，用来修改时间，如图 3 所示。在矩阵式键盘中，每条水平线和垂直线在交叉处不直接连通，而是通过一个按键加以连接，将按键当作一个开关，以行扫描为例，在矩阵键盘函数中会对四行不断进行扫描，直到有按键按下后（扫描到对应行时），该键即可导通，从而使单片机收到按键信息。

2.2.2矩阵键盘实物图

图 3 矩阵键盘实物图

2.2.3矩阵键盘与 MSP430F5438A 接口电路

图 4 矩阵键盘电路接口

矩阵键盘每一列分别于单片机的 P2.4~P2.6 相连,每行分别于单片机的 P2.0~P2.3 相连。

2.3 液晶 12864 模块

2.3.1液晶介绍

12864 汉字图形点阵液晶显示模块，可显示汉字及图形。其主要特征有如下几点：

（1）显示分辨率：128*64

（2）内置汉字字库，提供 8192 个 16*16 点阵汉字

（3）内置 128 个 8*16 点阵字符

（4）通信方式：串口、并口

（5）低电源电压（3～5.5V）

（6）工作温度：0～55 度

图 5 12864液晶的引脚图

本次的课程设计使用的液晶屏是 12864。具有 8 位并行和 3 线串行连接方式，内部含有简体中文字库的点阵图形液晶显示模块；其显示分辨率为 128×64,内置 8192 个 16*16 点阵的中文汉字，和 128 个 8*16 点阵的字符，可以显示 4 行 8 列 16*16 点阵的汉字，也可完成图形显示，具有光标显示、画面移位、自定义字符、睡眠模式等功能，操作简单。

2.3.2液晶与 MSP430F5438A 接口电路

图 6 12864与单片机的接口电路

液晶的 DB0-DB7 与 P4.0~P4.7 是数据线，RS 与 P3.2 相连，RW 与 P3.3 相连，EN 与 P3.6 相连,RST 与 P3.7 相连，PSB 与 P3.0 相连，置高，表示并行。

2.4 DS1302 实时时钟芯片模块

2.4.1 DS1302 实时时钟芯片简介

DS1302是 DALLAS 公司推出的一种高性能、低功耗的实时时钟芯片，内含一个实时时钟/日历和 31 字节静态 RAM,采用 SPI 三线接口与 CPU通信。

（1）DS1302实时时钟具有能计算 2100 年之前的秒、分、时、日、日期、星期、月、年的能力，还有闰年调整的能力。

（2）内部含有 31 个字节静态 RAM 可提供用户访问。

（3）采用串行数据传送方式，使得管脚数量最少，简单 SPI 3 线接口。

（4）工作电压范围宽：2.0～5.5V。

（5）工作电流：2.0V 时，小于 300nA。

（6）时钟或 RAM 数据的读/写有两种传送方式：单字节传送和多字节传送方式。

（7）采用 8 脚 DIP 封装或 SOIC封装。

（8）与 TTL兼容，Vcc=5V。

（9）可选工业级温度范围：-40 C～+85 C。

（10）具有涓流充电能力。

（11）采用主电源和备份电源双电源供应。

（12）备份电源可由电池或大容量电容实现。

图 7 DS1302 芯片的引脚图

2.4.2 DS1302 实时时钟芯片实物图

图 8 DS1302 芯片实物图

2.4.3 DS1302 实时时钟芯片与 MSP430F5438A 接口电路

DS1302 与单片机的连接仅需要 3 条线：时钟线 SCLK 数据线 I/O 和复位线 RST。连接图如下图。时钟线 SCLK与 P6.7 相连，数据线 I/O 与 P6.6 相连，复位线 RST 与 P6.5 相连，x1 和 x2 之间连接了一个基频为 32768HZ的外部晶振，GND 接地，Vcc1 为备用电源，Vcc2 接单片机上 3V电压。

图 9 DS1302 接口电路图

2.4.4 SPI 协议简介

SPI 总线是 Motorola 公司推出的三线同步接口，同步串行 3 线方式进行通信:一条时钟线 SCK，一条数据输入线 MOSI，一条数据输出线 MISO;用于 CPU 与各种外围器件进行全双工、同步串行通讯。SPI 主要特点有:可以同时发出和接收串行数据;可以当作主机或从机工作;提供频率可编程时钟;发送结束中断标志;写冲突保护;总线竞争保护等。

2.5 DS18B20 温度芯片模块

2.5.1 DS18B20 温度芯片简介

图 10 DS18B20 芯片图

该芯片仅需一个端口引脚进行通信，每个器件有唯一的 64 位的序列号存储在存储器中，高速暂存器中含有两个字节的温度寄存器，这两个寄存器用来存储温度传感器输出的数据，除此之外，高速暂存器提供一个直接的温度报警值寄存器

（TH 和 TL），和一个字节的配置寄存器，它们是非易失性的可擦除程序寄存器（EEPROM），所以存储的数据在器件掉电时不会消失。可通过数据线供电，供电范围为 3.0V 到 5.5V，测温范围为-55℃到+125℃，在-10℃到+85℃范围内精确度为±5℃。温度计分辨率可以被使用者选择为 9-12 位，DS18B20 芯片图如图 9 所示。

GND:电源地。

VDD:电源引脚。

DQ:数据 I/O。

2.5.2 DS18B20 温度芯片实物图

图 11　DS18B20 芯片实物图

2.5.3 DS18B20 温度芯片与 MSP430F5438A 接口电路

图 12　DS18B20 接口电路图

（1）DS18B20 的 VDD 接单片机的电源（3.0V 到 5.5V），GND 接地，DQ 为数据接口，负责发送和接收数据。

（2）DS18B20 的核心功能是它的直接读数字的温度传感器。温度传感器的精度为用户可编程的 9，10，11 或 12 位，分别以 0.5℃，0.25℃，0.125℃和 0.0625℃增量递增。在上电状态下默认的精度为　12 位。

（3）DS18B20 启动后保持低功耗等待状态；当需要执行温度测量和 AD 转换时，总线控制器必须发出[44h]命令。在那之后，产生的温度数据以两个字节的形式被存储到高速暂存器的温度寄存器中，DS18B20 继续保持等待状态。当 DS18B20 由外部电源供电时，总线控制器在温度转换指令之后发起“读时序”，DS18B20 正在温度转换中返回 0,转换结束返回 1。如果 DS18B20 由寄生电源供电，除非在进入温度转换时总线被一个强上拉拉高，否则将不会由返回值。

（4）VDD 连接 5V 电压，GND 接地，DQ 连接 P6.0。

（5）DS18B20 储存器如图 13 所示，位 0 和位 1 为测得温度信息的 LSB 和 MSB。这两个字节是只读的。第 2 和第 3 字节是 TH 和 TL 的拷贝。位 4 包含配置寄存器数据。位 5，6 和 7 被器件保留，禁止写入；这些数据在读回时全部表现为逻辑 1。所以当读取温度时，需要连续读取两次（位 0 和位 1）才能将温度信息完全读出。

图 13　DS18B20 存储器图

2.5.4单总线协议简介

DS18B20 通过达拉斯公司独有的单总线协议依靠一个单线端口通讯。当全部器件经由一个 3 态端口或者漏极开路端口(DQ 引脚在 DS18B20 上的情况下)与总线连接的时候，控制线需要连接一个弱上拉电阻。在这个总线系统中，微控制器（主器件)依靠每个器件独有的 64 位片序列号辨认总线上的器件和记录总线上的器件地址。由于每个装置有一个独特的片序列码,总线可以连接的器件数目事实上是无限的。

3.软件设计

3.1 系统总体设计

课设中单片机连接显示屏 12864、实时时钟芯片 DS1302、温控芯片 DS18B20 以及矩阵键盘，在正常情况下矩阵键盘不工作，单片机从 DS1302 和 DS18B20 中读取相应信息并显示在显示屏 12864 上，具体读写代码在附录中有详细说明，当需要修改时间时，矩阵键盘开始工作。按矩阵键盘“0”键进入修改界面，选择要修改的内容（年、月、日、时、分、秒、周），修改过程中 DS1302 停振（通过将秒寄存器中的最高位 CH 置 1 使其停振），DS18B20 正常工作。选择修改内容时，每一个键盘值对应修改一个内容（其中一个键代表退出），选择后即可进入修改，修改后，将修改内容发送给 DS1302，最后退出修改界面，正常显示时间和温度。

3.1.1系统流程图

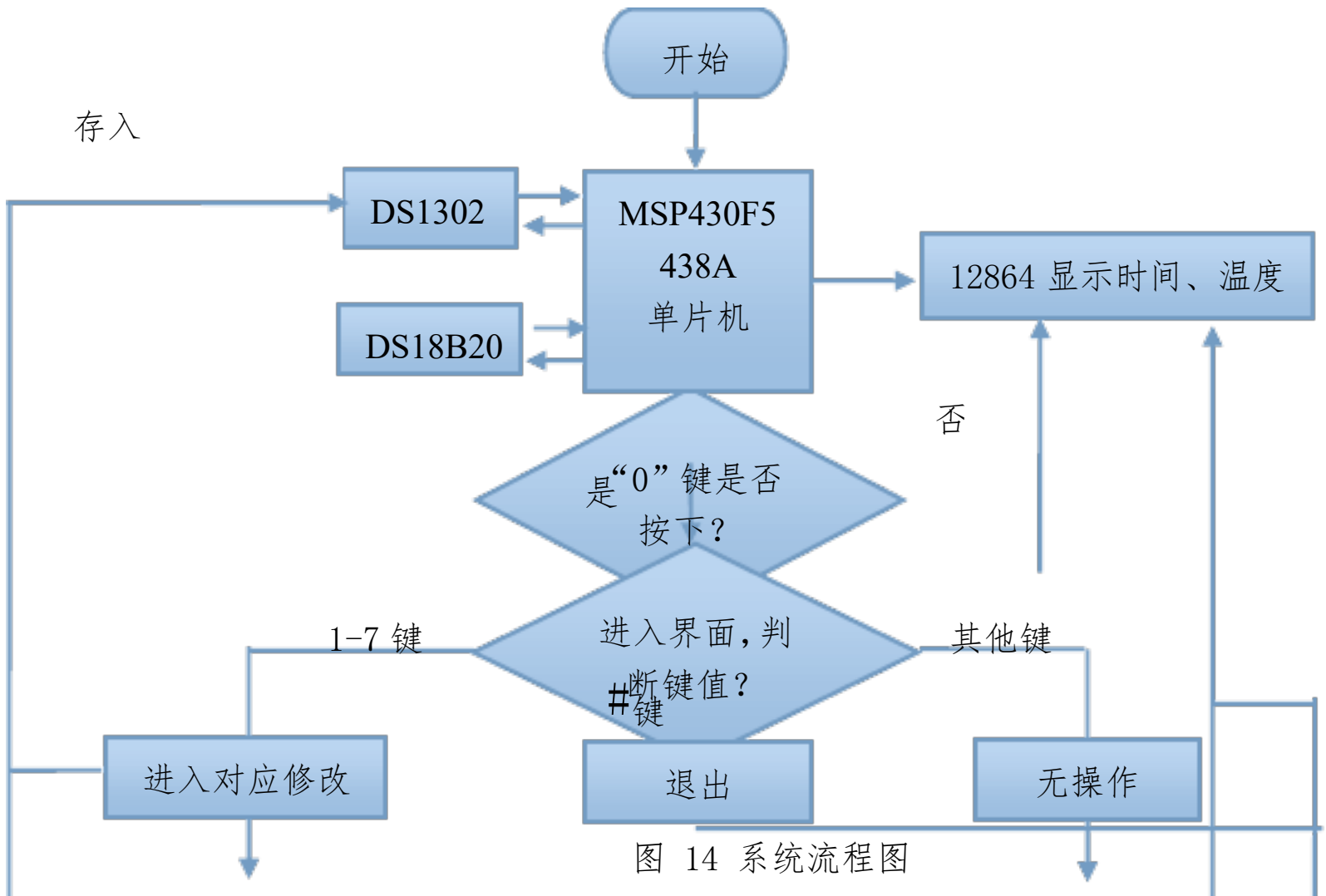

图 14 系统流程图

单片机在未收到矩阵键盘信号时，只是读取 DS1302和 DS18B20的时间和温度信息并在 12864 上进行显示，当有“0”按键按下时，进入按键中断，1-7 号按键

分别对应年、周、月、日、时、分、秒的修改，“#”为退出修改，修改时间时，DS1302停止工作，时间修改结束后从上一次进入中断的时间开始运行。

3.2.1按键进入修改界面

```
if(P2IFG!= 0X00)
    delay_ms(30);
    if(P2IFG!= 0X00)
     _EINT();
```

1、首先要进行按键进行修改界面的部分，由于程序采用按键中断，所以在判断时没有进行扫描，而是直接判断终端标志位，当有按键按下时，P2IFG 将不等于 0X00，以此判断出有按键按下，所以按下任意键都可以进入修改程序，但在本程序中，在液晶屏上显示为按“0”键进入修改，原因是本次课设矩阵键盘不太灵敏，1-7 键在进入修改界面后都有选择修改内容的作用，而“0”键在进入修改界面后不再做功能键使用，即使出现消抖不到位的情况也不会对修改产生影响。

2、在确认有键按下后，进入中断，DS1302停振，进入修改界面，屏幕上显示供给用户修改的选择。

3.2.2按键选择修改内容

```
#pragma vector=PORT2_VECTOR
__interrupt void Port2(void)    //              进入中断
 do
  change_disp();          //                    修改界面
  delayms(50);
  key2();                 //key2                用来取数值,选择对应修改内容
 }while(j);               //j                   初始化为 1，当取到键值后变为 0
```

1、进入修改界面后，用户可以选择修改内容（1.年、2.周、3.月、4.日、5.时、6.分、7.秒、#.退出），key2 函数就是用来选择这些功能。变量 j 初始化为 1，一直进行循环，当在循环时有按键按下时进入 key2 函数，对矩阵键盘进行行扫描（0XFE, 0XFD, 0XFB, 0XF7），当按键按下时，对应按键的开关被打开（由列控制），当行扫描扫到时，按键出形成通路，相应的按键信息将发送给单片机，按键成功。

2、key2 函数中，不同键值实质代表不同的 s 值，s 是选择修改哪种信息的变量，键盘值 1-7 代表 s 值 1-7；而 s 值（1-7）分别代表进入年、周、月、日、时、

分、秒修改，s 值为 11 时表示按下为“#”键，意为退出修改。当按键成功后，将 j 置为 0，表示跳出当前 DO-WHIL循环，程序继续向下执行。

```
__interrupt void Port2(void)
 if(s==1)                  //s=1                    时修改年
  change_year();
 if(s==7)                  //s=7                    时修改秒
  change_second();
if(s==11)                 //s=11                   时退出修改
  P2IFG= 0X00;
  lcd_clr();
if(s!=7)                  //                       如果修改的不是秒，则将中断前的秒还原
                                                    //这是将之前时钟停振进行复原的操作
TIME[0]=TIME1[0];
Ds1302Init();
```

3、该段为选择修改内容的部分，键值 1-7 在上一阶段已经介绍，当 s=11 时意为按下“#”键，退出修改界面，回到主界面。

4、最后 if（s!=7 ）TIME[0]=TIME1[0]; 意为假如用户之前修改的内容不是秒，则修改完成后，时钟恢复运转后将秒数还原成停振时的秒数，也就是满足题目要求的修改过程中时钟停振的要求（TIME1[0]是暂存停振前秒数的数组，TIME[0]是用来存储时间信息的数组），如果修改的是秒（s=7），则停振后秒数已经发生改变，所以不用进行此操作（秒数已经有了新值），但并不影响时钟恢复工作（因为正常的秒数为 0-59，CH位为 0）。

key1（）：键入个位内容函数

key0（）：键入十位内容函数

1、当进入修改时间函数后（如 change_year()...change_second() ），用 key1（）和 key0（）函数分别键入所要修改值的个位和十位，根据我们平时的习惯，我们一般先输入十位，再输入个位，所以先调用 key0（）函数，再调用 key1（）函数，矩阵键盘扫描原理与之前相同。假设我们要输入的年份为 16 年，则调用 key0（）函数时，按下按键“1”，得到 p=0X10，然后调用 key1（）函数，按下键“6”，得到 h=0X06，然后通过 q=h|p ，得到 BCD码 0X16，写入 DS1302 中，修改

成功。

电子万年历中，液晶的作用是显示时间和温度，以及给用户提供修改的显示界面，显示用户修改的信息。

显示功能在之前的实验中已经多次涉及，只需要注意汉字为 16X16的点阵，一个屏幕上只能显示 4X8个汉字即可，用数组存储汉字，通过 for 循环将所需要显示的相关信息读出。

需要注意的是在修改时间的时候，输入的时间需要显示在屏幕上，且输入的数字要实时显示在屏幕上，以修改年为例：

```
void change_year()
 dis[16]=
        {0xd6,0xd0+12,0xc9,0xe0+8,0xd6,0xc0+3,",0,0,0,0,0,0,0,0,0,0};
        Lcd_WriteCmd(0x90);
    for(int i=0;i<7;i++)
      Lcd_WriteData(dis[i]);
  do
     u=1;
    key0();
    temp1=p>>4;
    dis[7]=askii[temp1];
  }while(u==1);
  Lcd_WriteData(dis[6]);
      Lcd_WriteData(dis[7]);
      do
        u=1;
       key1();
       temp2=h&0x0f;
       dis[8]=askii[temp2];
      }while(u==1);
      Lcd_WriteData(dis[8]);
      delayms(1000);
```

在未按键输入的时候先显示“设置年”三个字，然后扫描键盘，当输入了数字后，将第一个数字存入 dis[7] 中（前六个存了设置年三个字），然后显示在屏幕上，然后继续扫描，将输入的按键存在 dis[8] 中，然后进行显示，延时一秒后返回初始界面。

其他相关函数：

```
void WaitBusy(void)      //          延时一小段时间，等待 LCD空闲
void lcd_clr(void)       //              清除 LCD的显示内容
void Lcd_WriteCmd(uchar cmdcode)  //       写指令
void Lcd_WriteData(uchar dispdata)  //            写数据
void Lcd_Init()       //              初始化 LCD
void hanzi_Disp()     //          初始界面
void change_disp()    //          修改时间界面
```

实时时钟芯片模块

首先要对　　的各个寄存器进行操作，那就要熟悉它的控制寄存器，控制寄存器用于存放 DS1302 的控制命令字，DS1302 的 RST 引脚回到高电平后写入的第一个字就为控制命令。它用于对 DS1302 读写过程进行控制，它的格式如下：

表 2　DS1302 的控制命令的格式

D7	D6	D5	D4	D3	D2	D1	D0
1	RAM/CK	A4	A3	A2	A1	A0	R/W

3.4.1 DS1302 的初始化

1、首先要注意的是秒寄存器 D7 位的 CH，该位如果置 1 的话，时钟就会停振，进入低功耗模式，所以在初始化该寄存器时注意该位不能置 1.

2、所谓初始化也就是对 DS1302 各个寄存器进行写操作，将你要显示的值写进 DS1302 中，接下来就是要对 DS1302 各个寄存器进行写操作，做这一步之前首先要将写保护寄存器的写保护位去掉，也就是将该寄存器的值改为 0x80，因为不去掉写保护位的话就不能对　DS1302 各个寄存器进行写操作。

3、接下来就是确定好写操作模式，是单字节写还是突发模式写，本次课设我采用的方式是突发模式写，所谓突发模式写就是一次性对 8 个特殊寄存器进行写操作。用 TIME[7]保存着特殊寄存器的值，其中 TIME[7] = {0x40, 0x12, 0x17, 0x16, 0x11, 0x01, 0x15};然后在利用 Ds1302Init()函数对 7 个寄存器进行突发写模式。

4、写完之后再将写保护加上。

图 15　DS1302 读写时序图

1、DS1302 的写过程，先要从单片机发送八个字节的写指令（地址），然后接着发送写的数据，如 Ds1302Write(0x80,0x80);　　前一个 0x80 代表指令，后一个 0x80 代表数据。

2、DS1302 的读过程，要先从单片机发送八个字节的指令，告诉　DS1302 要读的地址是什么，然后单片机做输入，接受　DS1302 传送过来的数据，读过程完成。

3、写进去的值是以二进制的　BCD 码存储在 DS1302 中的，要将该值取出其实很简单，就以 0x56 为例，它就代表着十进制的 56，要取出十位 5，只要(0x56>>4)就可以了，要取出各位就只需(0x56 & 0x0f)就可以了。

3. 5DS18B20 温度芯片模块

3.5.1 DS18B20 初始化

图 16　DS18B20 初始化过程图

主机首先发出一个 480—960 微秒的低电平脉冲，然后释放总线变为高电平，并在随后的 480 微秒时间内对总线进行检测，如果有低电平出现说明总线上有器件已做出应答。若无低电平出现一直都是高电平说明总线上无器件应答。

做为从器件的 DS18B20 在一上电后就一直在检测总线上是否有 480—960 微秒的低电平出现，如果有，在总线转为高电平后等待 15—60 微秒后将总线电平拉低 60—240 微秒做出响应存在脉冲，告诉主机本器件已做好准备。若没有检测到就一直在检测等待。

3.5.2 DS18B20 写操作

图 17　DS18B20 写过程图

写周期最少为 60 微秒，最长不超过 120 微秒。写周期一开始做为主机先把总线拉低 1 微秒表示写周期开始。随后若主机想写 0，则继续拉低电平最少 60 微秒直至写周期结束，然后释放总线为高电平。若主机想写 1，在一开始拉低总线电平 1 微秒后就释放总线为高电平，一直到写周期结束。而做为从机的 DS18B20 则在检测到总线被拉底后等待 15 微秒然后从 15us 到 45us 开始对总线采样，在采样期内总线为高电平则为 1，若采样期内总线为低电平则为 0。

3.5.3 DS18B20 读操作

图 18　DS18B20 读过程图

读时隙是从主机把单总线拉低之后，在 1 微秒之后就得释放单总线为高电平，

以让 DS18B20把数据传输到单总线上。DS18B20在检测到总线被拉低 1 微秒后，便开始送出数据，若是要送出 0 就把总线拉为低电平直到读周期结束。若要送出 1 则释放总线为高电平。主机在一开始拉低总线 1 微秒后释放总线，然后在包括前面的拉低总线电平 1 微秒在内的 15 微秒时间内完成对总线进行采样检测,采样期内总线为低电平则确认为 0。采样期内总线为高电平则确认为 1。完成一个读时序过程，至少需要 60us 才能完成。

```
int Save_Ds18b20()
    int temp3,temp4;
    Init_Ds18b20();              //                初始化
    delay_50us(18);
    Write_Ds18b20(0xcc);         //            跳过 ROM 配置
    Write_Ds18b20(0x44);         //            启动温度转换
    Init_Ds18b20();
    delay_50us(18);
    Write_Ds18b20(0xcc);         //            跳过 ROM 配置
    Write_Ds18b20(0xbe);         //            读温度寄存器
  delay_50us(10);                //600us
    temp3 = Read_Ds18b20();      //            读温度值的低字节
    temp4 = Read_Ds18b20();      //            读温度值的高字节
  temp4 = (temp4<<8 | temp3);    //              高低位合并;
   return (temp4);               //                返回温度
```

由于温度存在两位寄存器中，所以在读取过程中要连续进行两次读取，用两个变量保存，取出后再将高低位合并即为读出的温度值，在读值之前，还需要发送温度转换指令和读温度寄存器的指令才能将温度读出。

3.6

无论是 DS1302还是 DS18B20，都需要将数值读出再进行显示，时钟芯片读出的数值为 BCD码，即 0X15 年为 15 年，0X01 月为 1 月，温控芯片读出的二进制数字则需要转换才能成为十进制数字。

读取时钟芯片值时，由于一个 BCD码代表两位（十位、个位），所以要分别取出来，通过*TIME &0x0f取出个位，存入预定数组，通过*TIME++ >>4取出十位，存入数组，自增的目的是让 TIME自动指向下一个元素，继续读取。经过这样一个

过程，可以让时间信息的各位十进制存入指定的数组中，然后通过数组 ASCII[] 数组，将对应的值转化为 12864 可以显示的符号。

读取温控芯片值时，读出的是 16 位十六进制数值，要转化为十进制，根据用户手册可知该芯片默认精度为 12 位，对应增量为 0.0625，所以将取出的十六进制数乘以 0.0625 即可得出对应的十进制，从而用除法和求余等方法求出个位、十位、小数位，然后通过数组 ASCII[] 数组，将对应的值转化为 12864 可以显示的符号。

```
                                                选值:0-9
{                                       // 时 10.11   分 12.13  秒 14.15  年 16.17  月
                                          18.19  日 20.21  星 22.23  期 24.25
    0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0xca,0xb0+1,0
    xb7,0xd0+6,0xc3,0xe0+11,0xc4,0xe0+10,0xd4,0xc0+2,0xc8,0xd0+5,0x
    d0,0xc0+7,0xc6,0xd0+10,0x20};
```

4.实验结果

4.1 整体图

图 19 整体图

4.2运行过程

初始界面：

图 20 初始界面

按 0 键进入修改时间：

图 21 修改界面

将时间修改为 2016 年 1 月 1 日

图 22 修改年

图 23 修改月

图 24 修改日

图 25 修改后界面

输入错误时间如输入 80 秒，DS1302 停止工作：

图 26 输入错误时间

5. 缺陷与调试

5.1调试过程

（1）在最开始，读出温度和时间的数值的函数是通过 switch 语句判断的，这样造成了程序的冗长，导致程序易读性降低，在程序基本实现后，将程序进行了优化，使读值函数一目了然。

（2）程序一开始使用的不是按键中断，但是由于在本题中矩阵键盘被大量使用，需要多次扫描矩阵键盘，循环嵌套太多时，矩阵键盘使用效果不理想，请教杜神同学后，最终采用按键中断进行操作，使键盘的可靠性大幅提升。

（3）在实验中，由于线路较多，全部连接以后发现有些混乱，这是以后需要注意的，在连接之前要考虑到使用的方便，然后再分配引脚。

（4）在实验中，判断矩阵键盘的行列，直接通过程序判断，但答辩后知道用万用表测试会更加方便，也可以节省很多时间。

（5）在修改时间时，本考虑应该在输入时设置一个删除错误键值的功能，但考虑到程序已经将修改年、月、日等分开修改，即使输入错误也方便再一次进行输入，就没有添加删除功能。

（6）在设置 DS1302 和 DS18B20 时，部分代码忘记设置单片机引脚输入输出方向，导致芯片无法正常运行却也找不出原因，在今后还需要更加细心。

5.2程序的缺陷

程序未设置时间设置范围（正确时间），即使输入的是一个不正确的时间，也可以发送给 DS1302，应该在程序中针对年月日时分秒周的不同特点，对键入值进行限制，如果超出限制，则输入无效（附带提示）。

6. 实验心得

7. 附录

```
# include<msp430f5438a.h>
#include <stdlib.h>
#include <stdio.h>
#define uchar unsigned char
#define uint    unsigned int
//                                                  定义                                                  //
/*****DS1302 功能初始化指令*****/
#define hclk()       P6OUT |= BIT7          /时钟
#define lclk()       P6OUT &= (~BIT7)
#define hdat()       P6OUT |= BIT6          /数据
#define ldat()       P6OUT &= (~BIT6)
#define en()          P6OUT |= BIT5       /使能
#define disen()     P6OUT &= (~BIT5)
/时钟(寄存器) 命令
```

```
#define  CMD_ READ    0x01      /读操作位
#define CMD_WRITE    0x00      /写操作位
#define CMD_CONTROL 0x8E      /写保护
/时钟配置常量（位）
#define CFG_CLOCK_HALT 0x80    /暂停时钟运行 CH
#define CFG_PROTECT      0x80    /写保护
#define CFG_UNPROTECT   0x00   /写允许
/*****12864功能初始化指令*****/
#define CLEAR_SCREEN 0x01    /清屏指令：清屏且 AC 值为 00H
#define AC_INIT          0x02    /将 AC 设置为 00H。且游标移到原点位置
#define CURSE_ADD       0x06    /设定游标移到方向及图像整体移动方向（默认游标右移，
图像整体不动）
#define FUN_MODE        0x30    /工作模式：8 位基本指令集
#define DISPLAY_ON      0x0c    /显示开,显示游标，且游标位置反白
#define DISPLAY_OFF    0x08    /显示关
#define LCM_Data_Out P4OUT   /输出 DBO0-DBO7
#define LCM_Data_In    P4IN    /输入 DBO0-DBO7
#define SET_LCD_RS    do{P3OUT |= BIT2;}while(0)        // LCD_RS=1 1 为读写数据
#define CLR_LCD_RS    do{P3OUT &= ~(BIT2);}while(0)    // LCD_RS=0 0 为读写指令
#define SET_LCD_RW    do{P3OUT |= BIT3;}while(0)        // LCD_RW=1 1 为读
#define CLR_LCD_RW    do{P3OUT &= ~(BIT3);}while(0)    // LCD_RW=0 2 为写
#define SET_LCD_EN    do{P3OUT |= BIT6;}while(0)        // LCD_EN=1 1 为使能
#define CLR_LCD_EN    do{P3OUT &= ~(BIT6);}while(0)    // LCD_EN=0 0 为断开
#define SET_LCD_RST do{P3OUT |= BIT7;}while(0)        // LCD_RST=1 1 为复位
#define CLR_LCD_RST    do{P3OUT &= ~(BIT7);}while(0)    // LCD_RST=0 0 为不变
#define keyin    (P2IN&0xf0)
//                                          声明                                          //
void delayms(uint t);
void WaitBusy(void);
void Lcd_WriteCmd(uchar cmdcode);
void Lcd_WriteData(uchar dispdata);
void Lcd_Init();
void lcd_clr(void);
void read_t();
void Init_Ds1302();
unsigned char* TimeToString(unsigned char* TIME);
void Write_to_Ds1302(uchar address, uchar data);
char Read_from_Ds1302(uchar address);
void Transfer_Ds1302();
void key0();
void key1();
void key2();
void change_disp();
void change();
```

```
void change_year();
void change_week();
void change_month();
void change_day();
void change_hour();
void change_minute();
void change_second();
int a;//read_t 中临时变量
char h,p;
unsigned int x,y,z,w;//read_t 中临时变量
int j,temp,u=1,s,q,temp1,temp2,temp3;
unsigned char t1[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};//12864 第一行显示温度及周
unsigned char t2[12]={0,0,0,0,0,0,0,0,0,0,0,0};//12864 第二行显示年月日
unsigned char t3[12]={0,0,0,0,0,0,0,0,0,0,0,0};//12864 第三行显示时分秒
                        按 0 进入修改          第四行显示
unsigned char TIME[7] = {0x40, 0x12, 0x17, 0x16, 0x11, 0x01, 0x15};//存放秒，分，时，日，月，
周，年
unsigned char READ[7] = {0x81, 0x83, 0x85, 0x87, 0x89, 0x8b, 0x8d};//DS1302 读命令
unsigned char WRITE[7] = {0x80, 0x82, 0x84, 0x86, 0x88, 0x8a, 0x8c};//DS1302 写命令
                                    输入修改时间      修改界面
                                      按* 确认        修改界面
unsigned char change7[2] = {0,0};//修改时间值
unsigned char change8[2] = {0,0};//修改时间值
unsigned char ASCII[] =       //12864&DS18B20 选值:0-9
{                                   /时 10.11  分 12.13 秒 14.15
                                    /年 16.17 月 18.19 日 20.21 星 22.23 期 24.25
  0x30,0x31,0x32,0x33,0x34,0x35,0x36,
  0x37,0x38,0x39,0xca,0xb0+1,0xb7,0xd0+6,
  0xc3,0xe0+11,0xc4,0xe0+10,0xd4,0xc0+2,
  0xc8,0xd0+5,0xd0,0xc0+7,0xc6,0xd0+10,0x20
//                                  按键中断                                    //
#pragma vector=PORT2_VECTOR
__interrupt void Port2(void)
  j=1,s=0;
  unsigned char TIME1[1];                   /临时存放秒
  TIME1[0]=TIME[0];
  Write_to_Ds1302(0x8E,0X00);       /关闭写保护功能
  Write_to_Ds1302(0x80,0x80);               /时钟停振，CH 置 1
  Write_to_Ds1302(0x8E,0X80);       /打开写保护功能
  do
    change_disp();
    delayms(50);
    key2();
  }while(j);                                //j 初始为 1，取到按键后 j 变为 0
```