STC 单片机四次作业——红外遥控 LCD 时钟

一、实验要求

使用 STC 单片机上的红外接收器和配套的红外遥控器，实现对 STC 单片机实验箱上的资源进行控制和交互。

（1）STC 单片机能正确接收到红外遥控器的编码信息，并显示（不限制显示介质，串口或 1602）（50 分）

（2） 能控制 LED（20 分）

（3） 能实现更复杂的显示交互和控制功能（30 分）

二、实现功能

（1）在 LCD1602 上第一行显示红外遥控编码信息。

（2）将第三次作业 LCD 时钟融合进来。

- 能正常计时
- 可以用红外遥控实现切换日历，调整时间的功能
- 调整时间时有闪烁光标指示
- 用一个指示灯 LED8 指示当前是时钟还是日历

三、状态机示意图

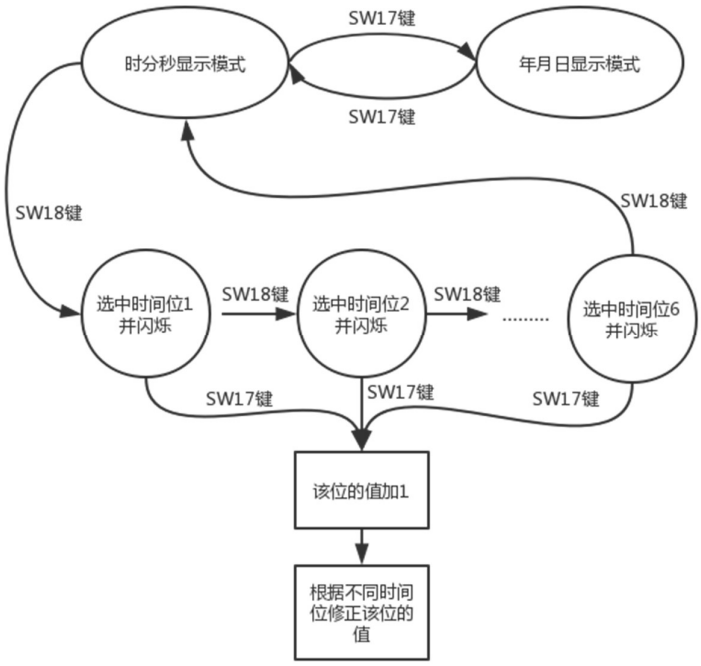

图 1 普通 LCD 时钟状态机

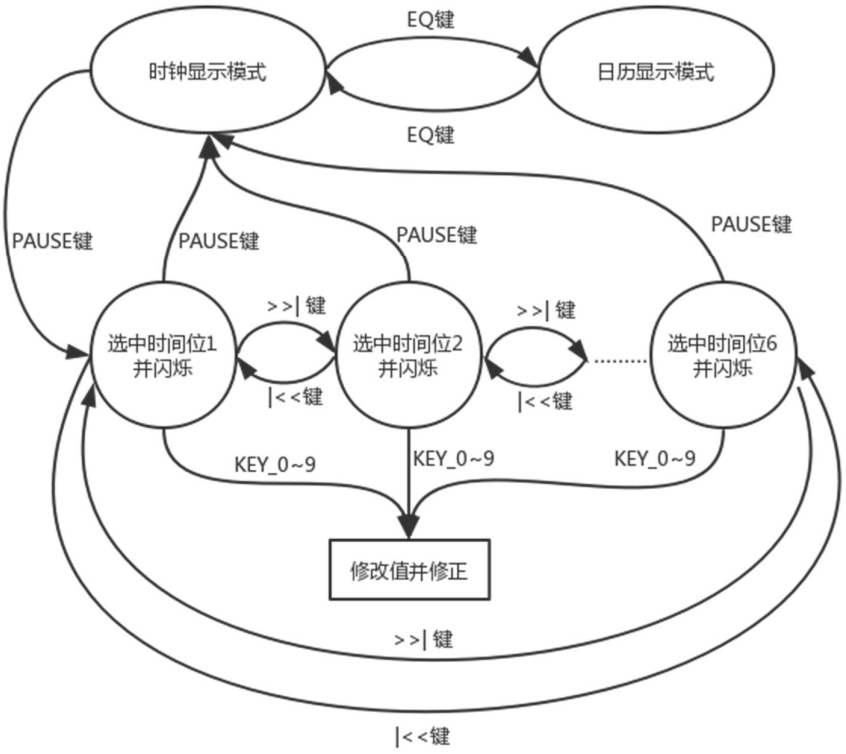

图 2 红外遥控 LCD 时钟状态机

四、具体实现

1、变量定义及初始化

先将按键数据码写成宏定义

```
// 遥控板按键数据码
#define LEFT    0x44
#define RIGHT   0x40
#define PAUSE   0x43
#define EQ      0x09
#define KEY_0   0x16
#define KEY_1   0x0C
#define KEY_2   0x18
#define KEY_3   0x5E
#define KEY_4   0x08
#define KEY_5   0x1C
#define KEY_6   0x5A
#define KEY_7   0x42
#define KEY_8   0x52
#define KEY_9   0x4A
```

```
/**********时钟显示部分用到变量**********/

// 时间数据存储区长度
#define T1LEN 8
#define T2LEN 10

char time1[] = "00:00:00";
char time2[] = "2016/06/07";
char * hour;
char * min;
char * sec;
char * year;
char * month;
char * day ;
volatile bit time_mod = 0;
volatile bit conf_mod = 0;
volatile char * pos;
char disp_pos;
```

基本想法是将时钟和日历的数据存于某个区域，time1 即时钟数据区域的首地址，time2 为日历数据区域首地址，pos 是寻址指针，disp_pos 是寻址偏移量，如下图所示

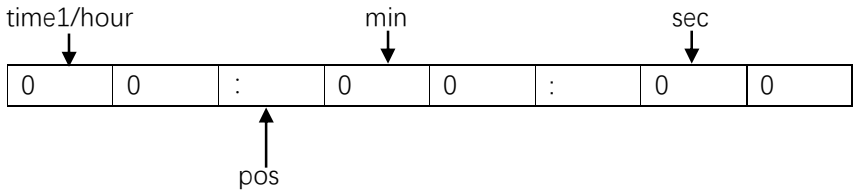

图 3 时钟数据存储模式（日历同理）

由于全局量初始化时只能赋常量值，所以这些指针在主函数中赋初值

```
void main(void)
{
  char i = 0;

  // 全局指针赋值
  hour = time1;
  min  = time1 + 3;
  sec  = time1 + 6;
  year = time2;
  month = time2 + 5;
  day   = time2 + 8;
  pos   = time1;
  disp_pos = pos - time1;
```

```
/**********红外部分用到变量**********/

#define FOSC  6000000L
#define BAUD  115200

/* 保存红外解码数据,下面调用了sprintf函数
 * 因keil库函数sprintf
 * 没有内存对齐，需将irdata设成
 * 4字节数据才能使sprintf函数所得结果正常
 */
unsigned int irdata[4]={0,0,0,0};
bit flag=0;
char disp_data[12]; // 解调后数据转换的字符串，用于lcd1602输出
```

disp_data 是由 irdata 通过 sprintf()函数转换成 16 进制字符串用于屏幕输出的

```
sprintf(disp_data,"%.2x %.2x %.2x %.2x",irdata[0],irdata[1],irdata[2],irdata[3]);
lcdshowstr(0,0,disp_data); // 显示按键解调数据
```

使用 6MHz 时钟，T1 计时器，12 分频，16 位自动重装载模式，最低得不到 1Hz 的频率，所以先得到 8Hz，再在 T1 中断函数里 8 分频

```
// 定时器T1寄存器装载值,得到8Hz的频率，定时中断内再8分频
#define TIMS 3036 // 计算公式 6MHz/12/(65536 - TIMS) = 8Hz
```

```
 * T1计时器中断
 */
void timer_1() interrupt 3
{
  static unsigned char n = 0;

  n++;
  if (n > 7) // 8分频
  {
    (*(sec + 1))++;  // 秒的个位加1
    n = 0;
  }
}
```

主函数中的计时器初始化

```
/**********Timer2 Initial**********/
TMOD = 0x00;
AUXR = 0x14;
TL2 = (65536-((FOSC/4)/BAUD));
TH2 = (65536-((FOSC/4)/BAUD))>>8;
AUXR2 |= 0x10;

/**********Timer0 Initial**********/

TL1 = TIMS;
TH1 = TIMS >> 8;
TR1 = 1;  // 打开计时器T1
ET1 = 1;  // 使能T1中断

EA=1; // 开总中断
```

2、红外中断函数

主要任务：

- 对遥控按键解码，此处参考书上例子
- 判断按键，执行相应操作

  如下图所示

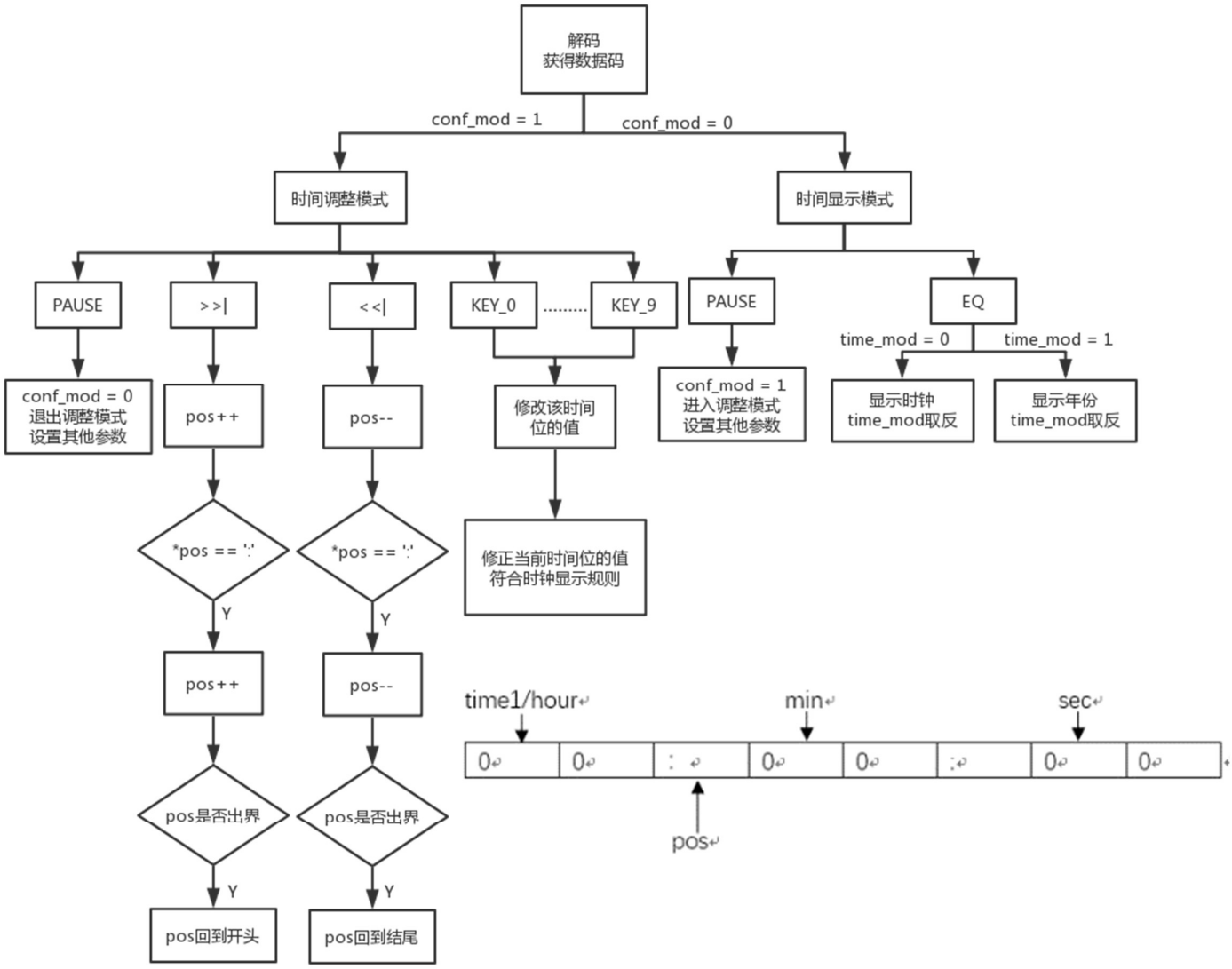

图 4 红外中断程序流程图

```
// 按键判断
if (!conf_mod) // 时间显示模式
  switch (irdata[2])
  {
   else // 时间调成模式
  {
    switch (irdata[2])
    {
    disp_pos = pos - time1; // 更新时钟位偏移量

    // 根据时钟位，对调整值进行修正，符合时钟显示规则
    switch(disp_pos)
    {
    lcdshowchar(disp_pos, 1, *pos); // 更改屏幕显示
    lcdsetcursor(disp_pos, 1); // 光标强制归位，减少显示错误

  }
```

根据数据手册，闪烁光标的打开与关闭，通过对 lcd1602 写控制字实现
lcdwritecmd(0x0f); // 光标所在位置的字符闪烁
lcdwritecmd(0x0c); // 关闭闪烁

3、主函数

任务：

- 显示按键编码
- 时间显示模式完成时间进位，并刷新显示
- 时间调整模式，刷新光标位置，改变及打印由红外中断完成

除了显示按键编码以外，其余都和第三次作业相同，详见第三次作业报告，不再赘述

```
while(1)
 {
 // 显示按键编码
  if(flag == 1)
  {
    lcdshowstr(0, 0, clear); // 清除第一行显示
    flag = 0;
    sprintf(disp_data,"%.2x %.2x %.2x %.2x",irdata[0],irdata[1],
    lcdshowstr(0,0,disp_data); // 显示按键解调数据
    AUXR2 |= 0x10;
  }


  // 时间显示模式
 if (!conf_mod)
 {
 // 调整时间模式
 else
   lcdsetcursor(disp_pos, 1); // 确定光标闪烁位

}
```

关于光标闪烁位的确定

对于闪烁光标，要做的就是打开关闭闪烁——通过对 LCD1602 写控制字实现，和确定光标位置。本能的想法是把这些功能都交给中断实现，但事实上还忽略了一点：光标位的刷新。

由于 LCD1602 打印字符串实际上靠光标来定位，在打印字符串的函数里可以证明这一点

```
void lcdshowstr(unsigned char x, unsigned char y,
                unsigned char *str)
{
 lcdsetcursor(x,y);
  while((*str)!='\0')
  {
     lcdwritedata(*str);
     str++;
  }
}
```

先设定光标位置，字符串才开始打印，其后每打印一次，光标会自动后移，这应该是由硬件完成的。也就是说任何一次刷新屏幕，都要用到光标。而现在在调整时钟时，要调整光标位置，就会带来大问题。

我一开始将光标的刷新也写在中断里，

```
//lcdsetcursor(time_n, 0); // 确定光标闪烁位
pos = time1 + time_n; // 确定需要改变的数值位置
disp_pos = time_n; // 记录位置偏移量
```

问题会是这样：当屏幕在刷新的过程中，实际上就是运行 lcdshowstr 这个函数的过程中，本来光标是一位一位自动后移，字符一个一个打印，两者一一对应，但是在打印没有完成时，忽然来了按键中断，改变了光标位置。而中断的特性是从程序哪里进中断，退出时就回到哪里。按键按了后，中断结束，这时候 lcdshowstr 函数继续执行 while 循环，挨个打印字符，但是光标的位置已经改变了，打印的位置随之改变，不再是之前的位置继续打印了。结果就是按下调整时间的按键，大概率出现显示错位问题。小概率正常，是因为恰好在字符串打印完时进了中断。

解决的办法如上所示，将光标位置的刷新写进主函数中，确保这个动作是在打印完任何一个字符串后执行，而不是在打印字符串过程中执行。