



杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

《单片机原理及应用》作业报告

大作业：串口屏秒表实物系统

学院 卓越学院

学号 23040447

姓名 陈文轩

专业 智能硬件与系统(电子信息工程)

2025 年 6 月 2 日

1 系统功能概述

本系统实现实物 51 单片机数码管秒表功能，并通过 TJC 串口屏与 51 单片机的 UART 通信，并对其进行功能设置与控制。秒表主要包含以下三大功能：

- **计时器功能：**支持开始/暂停计时及复位操作，时间精度为 0.01 秒，最大计时范围为 99 分 59 秒 99。通过串口屏或按键 K1 实现暂停/继续计时，按键 K2 实现复位功能，直观显示当前计时状态。
- **倒计时功能：**支持自定义设置倒计时时间（1 秒至 59 分 59 秒），默认设置为 1 分钟。提供继续/暂停、复位操作，并在倒计时结束时通过三个 LED 指示灯同时亮起 1 秒进行提示。支持通过串口屏直接输入秒数进行精确设置。
- **E2PROM 掉电存储功能：**利用 AT24C02 芯片实现数据掉电存储。无论在计时器还是倒计时模式下，均可将当前显示时间保存至 AT24C02，并在需要时读取。这可用于记录重要时间点或恢复上次使用状态。

系统采用多模式设计，用户可以通过串口屏在计时器和倒计时模式之间自由切换。串口屏通信成功后，均有 LED 指示灯反馈，确保用户能够清晰了解当前系统态。数码管显示格式为“MM-SS-ss”（分钟-: 百分秒），为用户提供精确的时间信息。

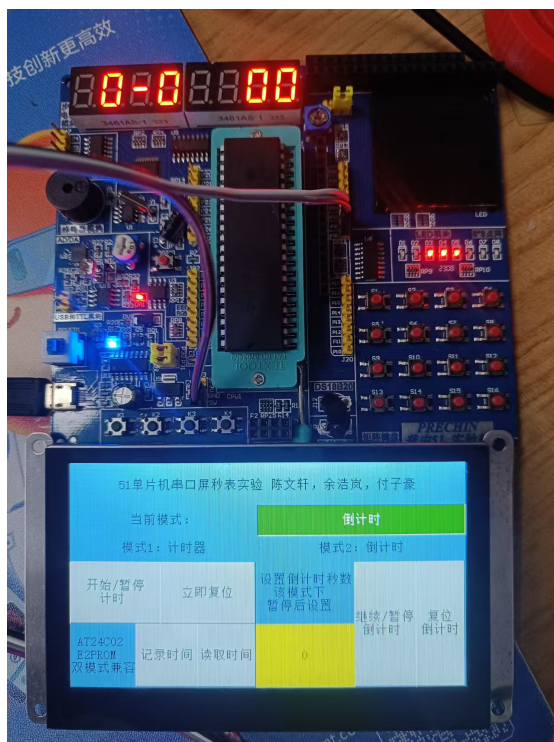


图 1 系统实物图（相机刷新率问题，数码管拍照不全，肉眼观看不会有此问题）

2 系统设计

2.1 硬件架构

本系统的硬件架构由以下主要组件构成：

- **控制核心：**51 单片机（STC89C52RC），负责系统的核心控制逻辑，作为数码管驱动的主控制器
- **人机交互界面：**TJC 串口屏，提供直观的操作界面，是系统的总输入集成
- **时间显示：**8 位数码管，用于显示时间信息（分: 秒: 百分秒），是系统的主要输出
- **状态指示：**多个 LED 指示灯，包括 UART 命令响应指示灯和倒计时结束指示灯，可以直观的监控系统状态
- **数据存储：**AT24C02 EEPROM 芯片，用于掉电数据存储，用于存贮时间

系统硬件构成总框图如下：

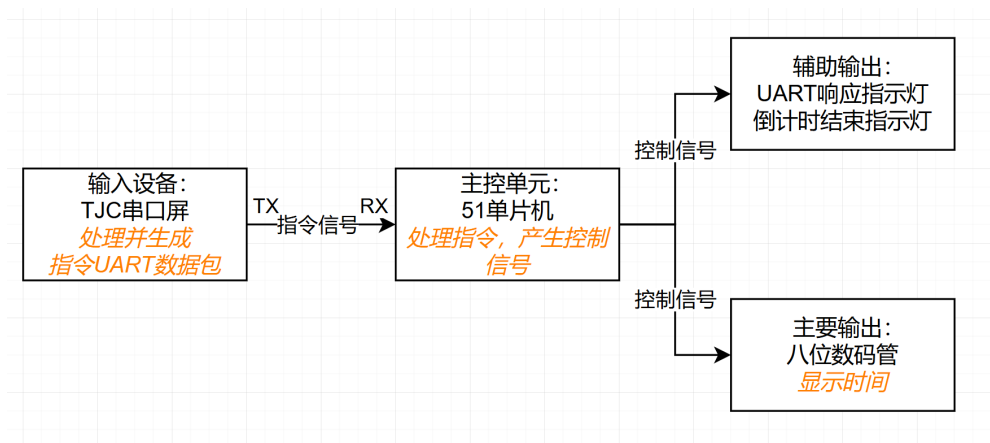


图 2 系统功能总框图

2.2 软件架构

软件设计采用模块化结构，包含以下主要模块：

- **主控模块：**负责系统初始化和主循环控制，主要就是对八位数码管的显示逻辑进行针对 UART 指令的实时调整
- **定时器模块：**提供精确的时基，驱动数码管显示和按键扫描
- **UART 通信模块：**处理串口屏与单片机间的数据交换
- **模式控制模块：**管理计时器和倒计时两种工作模式

- 数据存储模块：实现 AT24C02 的读写操作
- 时间显示驱动模块：控制数码管和 LED 的显示状态，并根据时分秒变量更新显示

同时，软件架构采用基于中断的工作方式，主要包括定时器中断和串口接收中断，确保系统实时响应外部输入并准确计时。

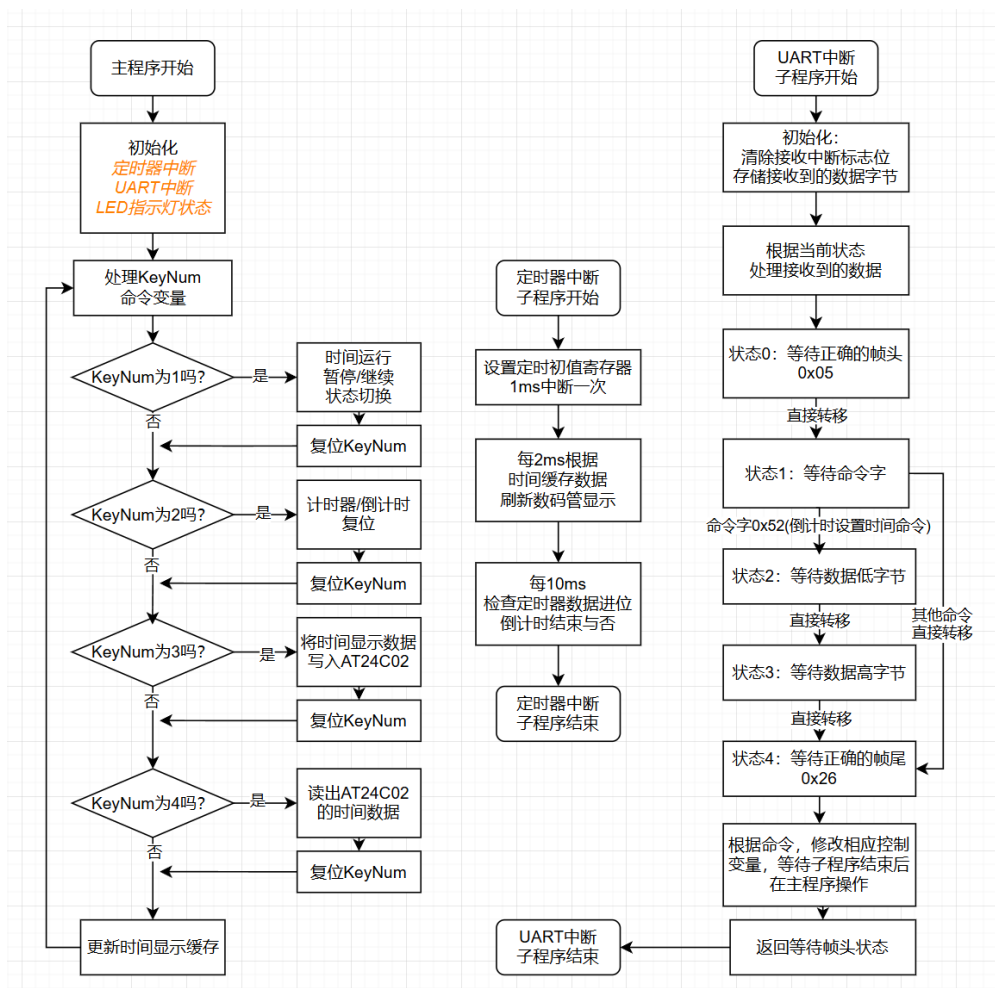


图 3 主程序控制流程图

3 软件系统设计

3.1 简易 UART 协议

本系统设计了一套简单有效的通信协议，用于串口屏与单片机之间的数据交换：

3.1.1 协议格式

每个通信帧由以下部分组成：

- 帧头：固定值 0x05，标识数据帧开始
- 命令字：表示操作类型，如 0x30，就表示在计时器模式下的开始/暂停命令
- 数据字段：只有 0x52 命令字下有效（倒计时设置命令），两个字节表示一个 0-65535 的秒数，小端模式表示数据
- 帧尾：固定值 0x26，标识数据帧结束

3.1.2 命令集

系统支持以下命令类型：

- 计时器控制命令
 - 0x30：暂停/继续计时
 - 0x31：复位计时器
 - 0x32：将当前时间写入 AT24C02
 - 0x33：从 AT24C02 读取时间
- 模式切换命令
 - 0x40：切换到计时器模式
 - 0x41：切换到倒计时模式
- 倒计时控制命令
 - 0x50：倒计时暂停/继续
 - 0x51：倒计时复位
 - 0x52：设置倒计时时间，后跟两字节表示秒数（低字节在前）

通过该协议，串口屏可以完全控制 51 单片机秒表系统的各项功能，并实现人机交互界面的快速响应。

3.2 数码管显示实现

本系统采用 8 位动态扫描数码管作为主要时间显示设备，能够实时显示“分-秒-百分秒”格式的计时信息。显示实现主要包括以下几个方面：

- **显示缓存区：**通过 Nixie_Buf 数组作为显示缓冲区，主程序每次更新时间后，调用 Nixie_SetBuf 函数将各位数字（如分钟、秒、百分秒）及分隔符“-”写入对应位置。
- **段码表：**NixieTable 数组存储了 0-9 数字和“-”等符号的段码，便于快速查表显示。
- **动态扫描：**在定时器中断中周期性调用 Nixie_Loop 函数，依次点亮每一位数码管，实现动态显示，避免鬼影和亮度不均。
- **位选与段选：**通过 P2.2 P2.4 控制位选，P0 输出段码，实现对 8 位数码管的独立控制。

主循环根据当前计时状态实时刷新显示缓冲区，数码管显示内容始终与内部时间变量保持同步，保证显示的准确性和实时性。该模块结构清晰，便于维护和扩展。

3.3 AT24C02 E2PROM 掉电存储实现

为实现掉电数据保存，系统采用 AT24C02 串行 EEPROM 芯片，通过 I2C 总线与单片机通信。主要实现方式如下：

- **数据结构：**将当前分钟、秒、百分秒分别存储在 AT24C02 的 0、1、2 地址单元。
- **写入操作：**调用 AT24C02_WriteByte 函数，依次将 Min、Sec、MiniSec 写入对应地址，写入后适当延时以确保数据可靠保存。
- **读取操作：**调用 AT24C02_ReadByte 函数，从 0、1、2 地址读取数据，恢复到 Min、Sec、MiniSec 变量，实现断电后时间的恢复。
- **I2C 协议：**底层通过 I2C 协议实现数据传输，包含起始、发送地址、数据、应答和停止等标准流程，保证通信的稳定性。

该功能支持用户通过按键或串口屏命令随时保存和恢复当前时间，极大提升了系统的实用性和可靠性，适用于需要断电记忆的应用场景。

3.4 TJC 串口屏设计

串口屏不是 51 单片机的软件设计内容，在本实验中作为可编程外设与输入设备参与系统搭建，这里进行简要介绍。

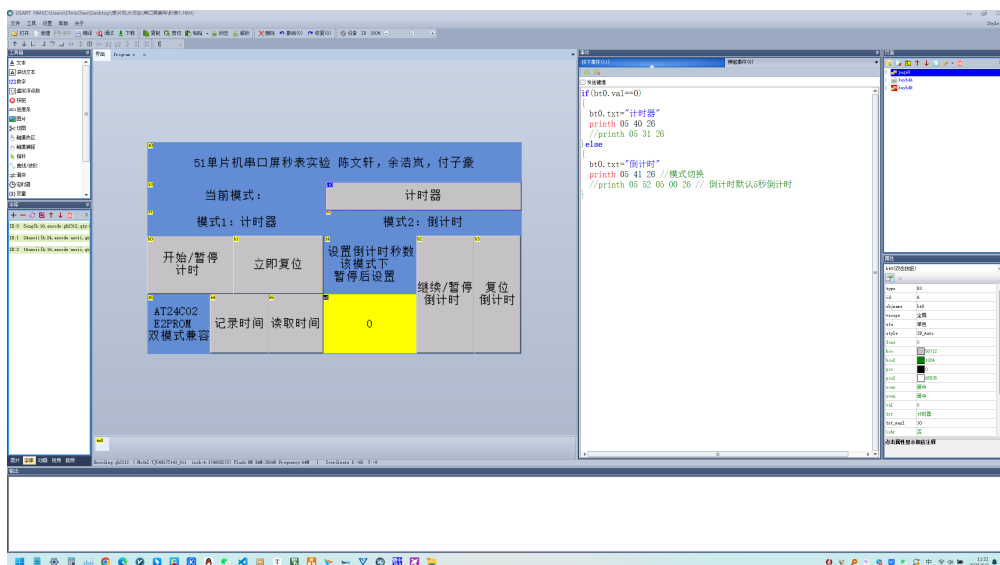


图 4 TJC 串口屏设计界面

如图，串口屏设计主要包含两大部分。首先就是界面中虚拟按钮/文本框/整形数字框等模块的位置设定，需要进行位置对齐等美观化调整；其次，对于主要输入的虚拟按钮，需要设计相应的按钮按下后，发送相应的 UART 数据命令包。如图中示例的是按下状态切换按钮后，根据当前按钮状态，发送不同的状态数据包，并对模式按钮上的文字进行更改，以指示当前系统工作模式（计时器/倒计时）。其余按钮也按类似的逻辑进行设计。

其中，对于倒计时时间设置，串口屏导入了数字键盘界面，通过添加相应的控制逻辑（按下键盘“OK”键逻辑添加），即可发送键盘上的数值，自动生成相应的数据包，最后传输给 51 单片机进行处理。

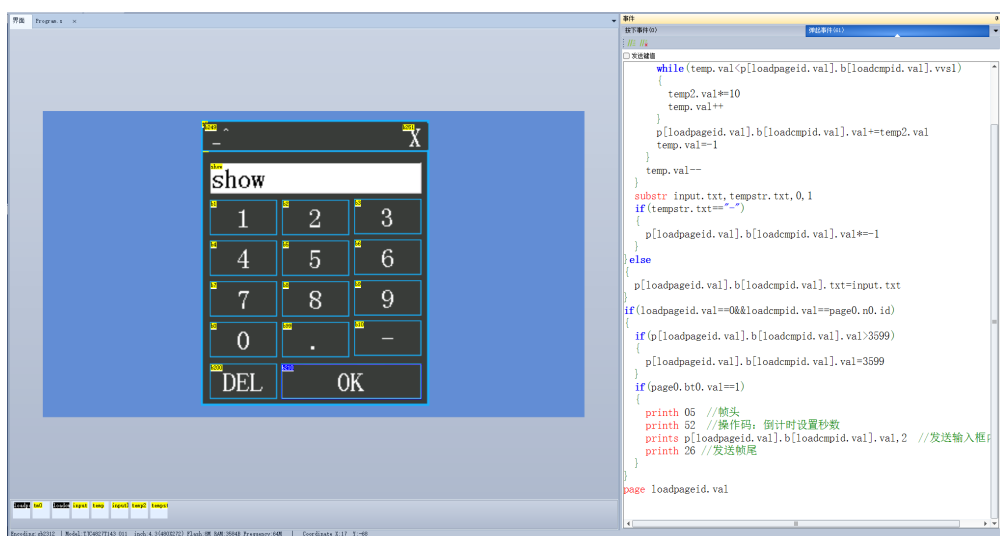


图 5 串口屏数字键盘设计，右侧 TJC 代码就是数据发送逻辑

4 系统测试与结果

本系统完成后进行了功能测试，测试结果表明系统能够稳定运行并满足设计要求：

- 计时器功能：计时基本准确，暂停/继续和复位功能工作正常
- 倒计时功能：能够准确设置倒计时时间，倒计时结束 LED 提示正常
- 数据存储功能：能够正确保存和读取 AT24C02 中的时间数据
- 串口通信：稳定可靠，可实现所有功能的远程控制

5 结论与展望

本设计成功实现了基于 51 单片机的多功能秒表系统，通过串口屏实现了良好的人机交互。系统整体性能稳定，功能齐全，达到了预期的设计目标。

未来可考虑以下改进方向：

- 增加更多计时模式，如 lap 计时、多段计时等
- 增强串口屏 UI 设计，提供更直观的用户界面
- 扩展存储功能，支持存储多组时间记录
- 添加蜂鸣器等声音提示功能，增强人机交互体验

6 本成员分工

本次大作业中，本人主要负责以下内容：

- **TJC 串口屏设计**：包括串口屏界面布局、虚拟按钮与数字键盘的设计、串口数据包发送逻辑的实现等，确保人机交互友好且功能完整。
- **UART 协议设计**：设计并实现了单片机与串口屏之间的通信协议，制定命令格式、数据帧结构及各类控制命令，保证系统各功能的可靠远程控制与数据交互。

在完成本部分内容后，进行了本部分的功能完整性的测试。小组成员共同验证完成后，进行了软硬件联调，最终实现了系统整体功能。

附录

Code Listing 1: main.c 核心功能实现代码

```

1
2 #include <REGX52.H>
3 #include "Timer0.h"
4 #include "Key.h"
5 #include "Nixie.h"
6 #include "Delay.h"
7 #include "AT24C02.h"
8
9 unsigned char KeyNum;
10 unsigned char Min,Sec,MiniSec;
11 unsigned char RunFlag;
12
13 // 模式控制变量
14 #define MODE_TIMER 0          // 计时器模式
15 #define MODE_COUNTDOWN 1     // 倒计时模式
16 unsigned char CurrentMode = MODE_TIMER; // 默认为计时器模式
17
18 // 倒计时设置值
19 unsigned char CDMin = 1, CDSec = 0, CDMiniSec = 0; // 倒计时默认值: 1分钟
20
21 // LED灯的定义
22 #define LED P2_0
23 #define CD_LED1 P2_5          // 倒计时结束指示灯1
24 #define CD_LED2 P2_6          // 倒计时结束指示灯2
25 #define CD_LED3 P2_7          // 倒计时结束指示灯3
26 unsigned char led_state = 1;
27 unsigned char DEAL_LED = 0; // 正确处理指令后, 再闪灯
28 // 定义通信协议
29 #define FRAME_HEADER 0x05     // 帧头
30 #define FRAME_FOOTER 0x26     // 帧尾
31
32 // 计时器指令
33 #define CMD_STOP_OR_CONTINUE 0x30 // 命令: 暂停/继续计时器
34 #define CMD_RESET 0x31 // 命令: RESET计时器
35 #define CMD_WRITE_AT 0x32 // 命令: 将当前时间放入AT存储器
36 #define CMD_READ_AT 0x33 // 命令: 读取AT存储器, 覆盖当前时间
37
38 // 模式切换指令
39 #define CMD_MODE_TIMER 0x40     // 切换到计时器模式

```

```
40 #define CMD_MODE_COUNTDOWN 0x41 // 切换到倒计时模式
41
42 // 倒计时控制指令
43 #define CMD_CD_PAUSE_CONTINUE 0x50 // 倒计时暂停/继续
44 #define CMD_CD_RESET 0x51 // 倒计时复位到设定值
45 #define CMD_CD_SET_TIME 0x52 // 设置倒计时时间
46
47 // 通信协议变量
48 unsigned char rx_state = 0; // 接收状态: 0-等待帧头, 1-等待操作数, 2-等
    待数据低字节, 3-等待数据高字节, 4-等待帧尾
49 unsigned char rx_command = 0; // 接收到的操作数
50 unsigned int rx_time_data = 0; // 接收到的时间数据(秒数)
51 unsigned char rx_time_low = 0; // 接收到的时间数据低字节
52 unsigned char rx_time_high = 0; // 接收到的时间数据高字节
53
54 void uart_init(unsigned int baud) // 9600:0XFA
55 {
56     TMOD |= 0X20; // 设置计数器工作方式2
57     SCON = 0X50; // 设置为工作方式1
58     PCON = 0X80; // 波特率加倍
59     TH1 = baud; // 计数器初始值设置
60     TL1 = baud;
61
62     // 配置中断器
63     ES = 1; // 打开接收中断
64     EA = 1; // 打开总中断
65     TR1 = 1; // 打开计数器
66 }
67
68 // 将所有倒计时指示LED设置为指定状态
69 void SetCountdownLEDs(bit state)
70 {
71     CD_LED1 = state;
72     CD_LED2 = state;
73     CD_LED3 = state;
74 }
75
76 void main()
77 {
78     Timer0_Init();
79     uart_init(0XFA); // 波特率为9600
80     // 初始化LED状态
81     LED = 0; // 初始状态为点亮(低电平)
```

```
82     led_state = 0;
83
84     // 初始化倒计时LED为熄灭状态
85     SetCountdownLEDs(1); // 高电平熄灭
86
87     while(1)
88     {
89         // 处理按键和命令
90         if(KeyNum==1) // K1按键按下
91         {
92             if(CurrentMode == MODE_TIMER)
93             {
94                 RunFlag=!RunFlag; // 计时器模式：启动标志
95                                     位翻转
96             }
97             else // 倒计时模式
98             {
99                 RunFlag=!RunFlag; // 倒计时模式也使用相同
100                                     的运行标志
101             }
102             KeyNum = 0; //处理完成，状态复位
103             DEAL_LED=1;
104         }
105         if(KeyNum==2) // K2按键按下
106         {
107             if(CurrentMode == MODE_TIMER)
108             {
109                 Min=0; // 计时器模式：分秒清0
110                 Sec=0;
111                 MiniSec=0;
112             }
113             else // 倒计时模式
114             {
115                 Min=CDDMin; // 倒计时模式：复位到设
116                                     定值
117                 Sec=CDDSec;
118                 MiniSec=CDDMiniSec;
119                 SetCountdownLEDs(1); // 复位指示灯
120             }
121             KeyNum = 0; //处理完成，状态复位
122         }
123         if(KeyNum==3) //K3按键按下
```

```
122         {
123             AT24C02_WriteByte(0,Min);           //将分秒写入AT24C02
124             Delay(5);
125             AT24C02_WriteByte(1,Sec);
126             Delay(5);
127             AT24C02_WriteByte(2,MiniSec);
128             Delay(5);
129             KeyNum = 0; //处理完成，状态复位
130
131         }
132         if(KeyNum==4)                          //K4按键按下
133         {
134             Min=AT24C02_ReadByte(0);           //读出AT24C02数据
135             Sec=AT24C02_ReadByte(1);
136             MiniSec=AT24C02_ReadByte(2);
137             KeyNum = 0; //处理完成，状态复位
138
139         }
140         // 显示当前时间
141         Nixie_SetBuf(1,Min/10);                //设置显示缓存，显示数据
142         Nixie_SetBuf(2,Min%10);
143         Nixie_SetBuf(3,11);
144         Nixie_SetBuf(4,Sec/10);
145         Nixie_SetBuf(5,Sec%10);
146         Nixie_SetBuf(6,11);
147         Nixie_SetBuf(7,MiniSec/10);
148         Nixie_SetBuf(8,MiniSec%10);
149     }
150 }
151
152 /**
153  * @brief 时间驱动函数，在中断中调用，根据当前模式执行不同操作
154  * @param 无
155  * @retval 无
156  */
157 void Sec_Loop(void)
158 {
159     if(RunFlag)
160     {
161         if(CurrentMode == MODE_TIMER) // 计时器模式
162         {
163             MiniSec++;
164             if(MiniSec>=100)
```

```
165         {
166             MiniSec=0;
167             Sec++;
168             if(Sec>=60)
169             {
170                 Sec=0;
171                 Min++;
172                 if(Min>=60)
173                 {
174                     Min=0;
175                 }
176             }
177         }
178     }
179     else // 倒计时模式
180     {
181         if(Min==0 && Sec==0 && MiniSec==0)
182         {
183             // 倒计时已经结束，停止运行
184             RunFlag = 0;
185
186             // 点亮指示灯
187             SetCountdownLEDs(0); // 低电平点亮
188             Delay(1000);         // 延时1秒
189             SetCountdownLEDs(1); // 熄灭指示灯
190
191             return;
192         }
193
194         // 倒计时逻辑
195         if(MiniSec==0)
196         {
197             if(Sec==0)
198             {
199                 if(Min!=0)
200                 {
201                     Min--;
202                     Sec=59;
203                     MiniSec=99;
204                 }
205             }
206             else
207             {
```

```
208         Sec--;
209         MiniSec=99;
210     }
211 }
212 else
213 {
214     MiniSec--;
215 }
216 }
217 }
218 }
219
220 // 串口通信中断函数
221 void uart() interrupt 4
222 {
223     unsigned char rec_data; // 接收到的数据
224
225     RI = 0; // 清除接收中断标志位
226
227     rec_data = SBUF; // 存储接收到的数据
228     SBUF = rec_data; // 将接收到的数据放入到发送寄存器
229
230     // 根据当前状态处理接收到的数据
231     switch (rx_state) {
232         case 0: // 等待帧头
233             if (rec_data == FRAME_HEADER) {
234                 rx_state = 1; // 进入等待操作数状态
235             }
236             break;
237
238         case 1: // 等待操作数
239             rx_command = rec_data; // 保存操作数
240             if (rx_command == CMD_CD_SET_TIME) {
241                 rx_state = 2; // 如果是设置时间命令，进入等待数据低字节状态
242             } else {
243                 rx_state = 4; // 否则直接进入等待帧尾状态
244             }
245             break;
246
247         case 2: // 等待数据低字节
248             rx_time_low = rec_data; // 保存时间数据低字节
249             rx_state = 3; // 进入等待数据高字节状态
```

```
250         break;
251
252     case 3: // 等待数据高字节
253         rx_time_high = rec_data; // 保存时间数据高字节
254         rx_state = 4; // 进入等待帧尾状态
255         break;
256
257     case 4: // 等待帧尾
258         if (rec_data == FRAME_FOOTER)
259         {
260             // 完整接收到一帧数据，根据操作数执行相应操作
261
262             // 计时器模式命令
263             if (rx_command == CMD_STOP_OR_CONTINUE &&
                CurrentMode == MODE_TIMER)
264             {
265                 KeyNum = 1; // 计时器暂停/继续
266                 DEAL_LED=1;
267             }
268             else if (rx_command == CMD_RESET &&
                CurrentMode == MODE_TIMER)
269             {
270                 KeyNum = 2; // 计时器复位
271                 DEAL_LED=1;
272             }
273             else if (rx_command == CMD_WRITE_AT)
274             {
275                 KeyNum = 3; // 将当前时间写入AT24C02
276                 DEAL_LED=1;
277             }
278             else if (rx_command == CMD_READ_AT)
279             {
280                 KeyNum = 4; // 从AT24C02读取时间
281                 DEAL_LED=1;
282             }
283             // 模式切换命令
284             else if (rx_command == CMD_MODE_TIMER)
285             {
286                 CurrentMode = MODE_TIMER;
287                 RunFlag = 0; // 切换模式时停止计时
288                 Min = 0;
289                 Sec = 0;
```

```
290         MiniSec = 0;
291         DEAL_LED=1;
292     }
293     else if (rx_command == CMD_MODE_COUNTDOWN)
294     {
295         CurrentMode = MODE_COUNTDOWN;
296         RunFlag = 0; // 切换模式时停止计时
297         Min = CDMIN; // 设置为默认倒计时值
298         Sec = CDSec;
299         MiniSec = CDMiniSec;
300         DEAL_LED=1;
301     }
302     // 倒计时控制命令
303     else if (rx_command == CMD_CD_PAUSE_CONTINUE
304             && CurrentMode == MODE_COUNTDOWN)
305     {
306         if (CurrentMode == MODE_COUNTDOWN)
307         {
308             KeyNum = 1; // 倒计时暂停/继续
309             DEAL_LED=1;
310         }
311     }
312     else if (rx_command == CMD_CD_RESET &&
313             CurrentMode == MODE_COUNTDOWN)
314     {
315         if (CurrentMode == MODE_COUNTDOWN)
316         {
317             KeyNum = 2; // 倒计时复位到
318                         设定值
319         }
320         DEAL_LED=1;
321     }
322     // 设置倒计时时间
323     else if (rx_command == CMD_CD_SET_TIME &&
324             CurrentMode == MODE_COUNTDOWN)
325     {
326         // 合并两个字节得到总秒数
327         rx_time_data = (unsigned int)
328             rx_time_high << 8 | rx_time_low;
329         if (rx_time_data > 3599) // 最大支持
330             59分59秒
```



```
326         {
327             rx_time_data = 3599; // 限制
                                   最大值为59分59秒
328         }
329         else if (rx_time_data <= 1) // 最小
                                   值为1秒
330         {
331             rx_time_data = 1; // 限制最
                                   小值为1秒
332         }
333         // 计算分和秒
334         CDMIn = rx_time_data / 60;
335         CDSec = rx_time_data % 60;
336         CDMiniSec = 0;
337
338         // 如果当前是倒计时模式并且没有在运
                                   行, 则立即更新显示
339         if (CurrentMode == MODE_COUNTDOWN &&
                                   RunFlag == 0)
340         {
341             Min = CDMIn;
342             Sec = CDSec;
343             MiniSec = CDMiniSec;
344         }
345         DEAL_LED=1;
346     }
347     if (DEAL_LED) // 如果需要闪烁LED指示
348     {
349         // 执行LED闪烁 指示收到了UART数据包
                                   并处理了
350         LED = 0; // 点亮LED (低电平点亮)
351         Delay(50); // 延时50ms
352         LED = 1; // 熄灭LED
353     }
354
355 }
356 // 不管帧尾是否正确, 都回到等待帧头状态
357 rx_state = 0;
358 break;
359 }
360
361 while(!TI); // 等待发送数据完成
362 TI = 0; // 清除发送完成标志位
```

```
363 }
364
365 void Timer0_Routine() interrupt 1
366 {
367     static unsigned int T0Count1,T0Count2,T0Count3;
368     TL0 = 0x18;           //设置定时初值
369     TH0 = 0xFC;           //设置定时初值
370     T0Count1++;
371     if(T0Count1>=20)
372     {
373         T0Count1=0;
374         Key_Loop();        //20ms调用一次按键驱动函数
375     }
376     T0Count2++;
377     if(T0Count2>=2)
378     {
379         T0Count2=0;
380         Nixie_Loop();      //2ms调用一次数码管驱动函数
381     }
382     T0Count3++;
383     if(T0Count3>=10)
384     {
385         T0Count3=0;
386         Sec_Loop();        //10ms调用一次数秒表驱动函数
387     }
388 }
```

Code Listing 2: Nixie.c 数码管显示实现代码

```
1
2 #include <REGX52.H>
3 #include "Delay.h"
4
5 //数码管显示缓存区
6 unsigned char Nixie_Buf[9]={0,10,10,10,10,10,10,10,10};
7
8 //数码管段码表
9 unsigned char NixieTable[]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0
    x6F,0x00,0x40};
10
11 /**
12  * @brief 设置显示缓存区
13  * @param Location 要设置的位置, 范围: 1~8
```

```
14  * @param Number 要设置的数字, 范围: 段码表索引范围
15  * @retval 无
16  */
17 void Nixie_SetBuf(unsigned char Location, Number)
18 {
19     Nixie_Buf[Location]=Number;
20 }
21
22 /**
23  * @brief 数码管扫描显示
24  * @param Location 要显示的位置, 范围: 1~8
25  * @param Number 要显示的数字, 范围: 段码表索引范围
26  * @retval 无
27  */
28 void Nixie_Scan(unsigned char Location, Number)
29 {
30     P0=0x00; //段码清0, 消影
31     switch(Location) //位码输出
32     {
33         case 1: P2_4=1; P2_3=1; P2_2=1; break;
34         case 2: P2_4=1; P2_3=1; P2_2=0; break;
35         case 3: P2_4=1; P2_3=0; P2_2=1; break;
36         case 4: P2_4=1; P2_3=0; P2_2=0; break;
37         case 5: P2_4=0; P2_3=1; P2_2=1; break;
38         case 6: P2_4=0; P2_3=1; P2_2=0; break;
39         case 7: P2_4=0; P2_3=0; P2_2=1; break;
40         case 8: P2_4=0; P2_3=0; P2_2=0; break;
41     }
42     P0=NixieTable[Number]; //段码输出
43 }
44
45 /**
46  * @brief 数码管驱动函数, 在中断中调用
47  * @param 无
48  * @retval 无
49  */
50 void Nixie_Loop(void)
51 {
52     static unsigned char i=1;
53     Nixie_Scan(i, Nixie_Buf[i]);
54     i++;
55     if(i>=9){i=1;}
56 }
```

Code Listing 3: AT24C02.c E2PROM 掉电存储实现代码

```
1
2 #include <REGX52.H>
3 #include "I2C.h"
4
5 #define AT24C02_ADDRESS      0xA0
6
7 /**
8  * @brief  AT24C02写入一个字节
9  * @param  WordAddress 要写入字节的地址
10  * @param  Data 要写入的数据
11  * @retval 无
12  */
13 void AT24C02_WriteByte(unsigned char WordAddress,Data)
14 {
15     I2C_Start();
16     I2C_SendByte(AT24C02_ADDRESS);
17     I2C_ReceiveAck();
18     I2C_SendByte(WordAddress);
19     I2C_ReceiveAck();
20     I2C_SendByte(Data);
21     I2C_ReceiveAck();
22     I2C_Stop();
23 }
24
25 /**
26  * @brief  AT24C02读取一个字节
27  * @param  WordAddress 要读出字节的地址
28  * @retval 读出的数据
29  */
30 unsigned char AT24C02_ReadByte(unsigned char WordAddress)
31 {
32     unsigned char Data;
33     I2C_Start();
34     I2C_SendByte(AT24C02_ADDRESS);
35     I2C_ReceiveAck();
36     I2C_SendByte(WordAddress);
37     I2C_ReceiveAck();
38     I2C_Start();
39     I2C_SendByte(AT24C02_ADDRESS|0x01);
40     I2C_ReceiveAck();
```

```
41     Data=I2C_ReceiveByte();
42     I2C_SendAck(1);
43     I2C_Stop();
44     return Data;
45 }
```

Code Listing 4: I2C.c I2C 通信功能实现代码

```
1  #include <REGX52.H>
2
3  sbit I2C_SCL=P2^1;
4  sbit I2C_SDA=P2^0;
5
6  /**
7   * @brief I2C 开始
8   * @param 无
9   * @retval 无
10  */
11 void I2C_Start(void)
12 {
13     I2C_SDA=1;
14     I2C_SCL=1;
15     I2C_SDA=0;
16     I2C_SCL=0;
17 }
18
19 /**
20 * @brief I2C 停止
21 * @param 无
22 * @retval 无
23 */
24 void I2C_Stop(void)
25 {
26     I2C_SDA=0;
27     I2C_SCL=1;
28     I2C_SDA=1;
29 }
30
31 /**
32 * @brief I2C 发送一个字节
33 * @param Byte 要发送的字节
34 * @retval 无
35 */
```

```
36 void I2C_SendByte(unsigned char Byte)
37 {
38     unsigned char i;
39     for(i=0;i<8;i++)
40     {
41         I2C_SDA=Byte&(0x80>>i);
42         I2C_SCL=1;
43         I2C_SCL=0;
44     }
45 }
46
47 /**
48  * @brief I2C接收一个字节
49  * @param 无
50  * @retval 接收到的一个字节数据
51  */
52 unsigned char I2C_ReceiveByte(void)
53 {
54     unsigned char i,Byte=0x00;
55     I2C_SDA=1;
56     for(i=0;i<8;i++)
57     {
58         I2C_SCL=1;
59         if(I2C_SDA){Byte|=(0x80>>i);}
60         I2C_SCL=0;
61     }
62     return Byte;
63 }
64
65 /**
66  * @brief I2C发送应答
67  * @param AckBit 应答位, 0为应答, 1为非应答
68  * @retval 无
69  */
70 void I2C_SendAck(unsigned char AckBit)
71 {
72     I2C_SDA=AckBit;
73     I2C_SCL=1;
74     I2C_SCL=0;
75 }
76
77 /**
78  * @brief I2C接收应答位
```

```
79  * @param 无
80  * @retval 接收到的应答位，0为应答，1为非应答
81  */
82  unsigned char I2C_ReceiveAck(void)
83  {
84      unsigned char AckBit;
85      I2C_SDA=1;
86      I2C_SCL=1;
87      AckBit=I2C_SDA;
88      I2C_SCL=0;
89      return AckBit;
90  }
```