微机原理与系统课程设计

交通信号灯自动控制模拟指示系统设计

一．课程设计目的

1．掌握 CPU 与各芯片管脚连接方法，提高接口扩展硬件电路的连接能力；
2．加深对定时器/计数器和并行接口芯片的工作方式和编程方法的理解；
3．掌握交通信号灯自动控制系统的设计思路和实现方法。

二．课程设计内容

设计并实现十字路口交通信号自动控制模拟指示系统。设该路口由 A、B 两条通行干道相交而成，两条干 道各设一组红、黄、绿三色信号灯，用四位数码管作倒计时显示。

三．系统功能与设计要求

1．基本功能要求

(1) 以秒为计时单位，四位数码管以十进制递减计数显示两个路口红黄绿灯剩余时间，在递减计数回零瞬间转换。十字路口交通灯的变化规律及控制时序：

① A 干道的绿灯、B 干道的红灯同时亮 30 秒，同时 A 干道数码管递减显示绿灯剩余时间，B 干道数码管递减显示红灯剩余时间；

② A 干道的黄灯闪烁 5 秒钟，A 干道数码管递减显示黄灯剩余时间，同时 B 干道的红灯继续亮，剩余红灯时间继续递减；

③ A 干道的红灯、B 干道的绿灯同时亮 30 秒，同时 B 干道数码管递减显示绿灯剩余时间，A 干道数码管递减显示红灯剩余时间；

④ A 干道的红灯继续亮，剩余红灯时间继续递减，同时 B 干道的黄灯闪烁 5 秒钟，B 干道数码管递减显示黄灯剩余时间;

⑤ 转 ① 重复。

(2) 通过实验箱 4×4 键盘对红、黄、绿三色信号灯所亮时间在 0～99 秒内任意设定。

2．发挥部分

(1) 增加人工干预模式，在特殊情况下可通过人工干预，手动控制 A、B 道路交通 灯的亮灭，并可以随时切换为自动运行模式；

(2) 增加夜间控制功能，交通灯在进入夜间模式后，A、B 两个干道上红、绿灯均 不亮，黄色信号灯闪烁显示；

3．创新功能

(1) 通过 8259 中断控制进行时钟延时，使得时间更精准，显示更加稳定；

(2) 综合运用各种器件使得系统架构设计更加高效，系统源码设计结构合理、简洁高效；

四．课程设计实验环境

1．硬件配置

微机一台、微机接口技术实验箱一个、ISA – PCI 转接卡一块、连接电缆一条、微机接口技术实验讲义一本、连接导线若干。

2．软件环境

Windows XP/2000/Win 7 平台以及星研集成开发环境。

五、设计思路

1．微机系统架构设计

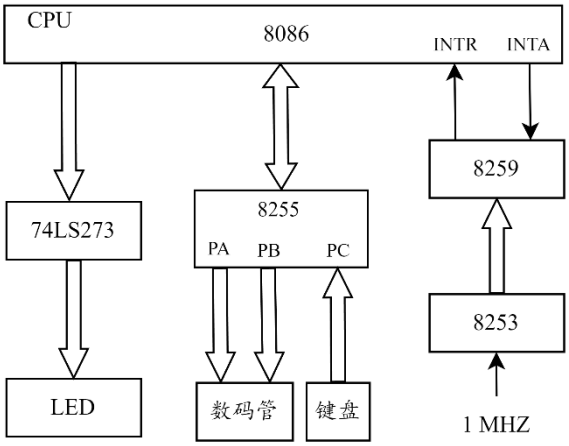

图 1 微机系统架构设计

如图 1 所示，本项目使用 8255、8253、8259 以及 74LS273 等多个器件控制外设进行数据的读入和输出，实现了一个小型的微机系统。

根据系统设计模式，对各器件作用进行说明如下：

- 8253 计数器

  8253 计数器用于产生时钟周期为 1s 的脉冲信号，用于实现 8259 中断控制器的中断请求，采用计数器 0 和计数器 1 对输入频率为 1MHZ 的时钟信号进行分频，计数初值均为 1000，实现 10^6 分频，因此得到周期为 1s 的脉冲信号。

- 8259 中断控制器

  8253 计数器产生每秒一次的中断请求，在开中断的情况下通过 8259 向 CPU 提出中断请求，调用终端服务子程序进行处理，如此实现对交通信号灯每秒一次的处理功能。

- 8255 通用并行接口

  为了将当前信号灯持续时间输出显示在数码管中以及允许接受键盘输入，系统利用 8255 进行外设控制，且配置 8255 为通用 IO 接口模式，分别利用 A、B 组接口连接数码管，B、C 组接口连接键盘。

- 74LS273 数据锁存器

  处理器系统在处理中断服务子程序时，只有在交通信号灯信号状态发生改变时，才将信号灯状态输出到 LED，但是 LED 灯需要实时显示当前信号状态，因此需要利用数据锁存器将上一状态锁存器来供 LED 输出显示。

2．硬件工作方式及初始化配置

(1) 8253 计数器

- 工作方式：计数器 0 方式 2 频率发生器；计数器 1 方式 3 方波发生器
- 初始化：

  控制端口: 00110101 // 计数器 0, 先低后高字节写计数初值，方式 2，十进制计数

  低字节：00000000

  高字节：00010000 BCD: 1000

  控制端口: 01110111 // 计数器 1, 先低后高字节写计数初值，方式 3，十进制计数

  低字节：00000000

  高字节：00010000 BCD: 1000

(2) 8255 通用并行 IO 接口

- 工作方式：A 口输出，B 口输出、输入，C 口输入
- 初始化：

控制端口：10001001 // 控制字，A 口方式 0、输出，C 口高四位输入，B 口方式 0、输出，C 口低四位输入

(3) 8259 中断控制器

- 工作方式：边沿触发，单片，非自动 EOI
- 初始化：

偶地址：ICW1 00010011 // 标志，边沿触发，8086 系统，单片方式，需要 ICW4

奇地址：ICW2 00001000 // 中断向量码

奇地址：ICW4 00001001 // 一般嵌套，从片，非自动 EOI，8086

奇地址：OCW1 11111110 // IR0 非屏蔽，IR1-IR7 屏蔽

3．微机系统线路设计

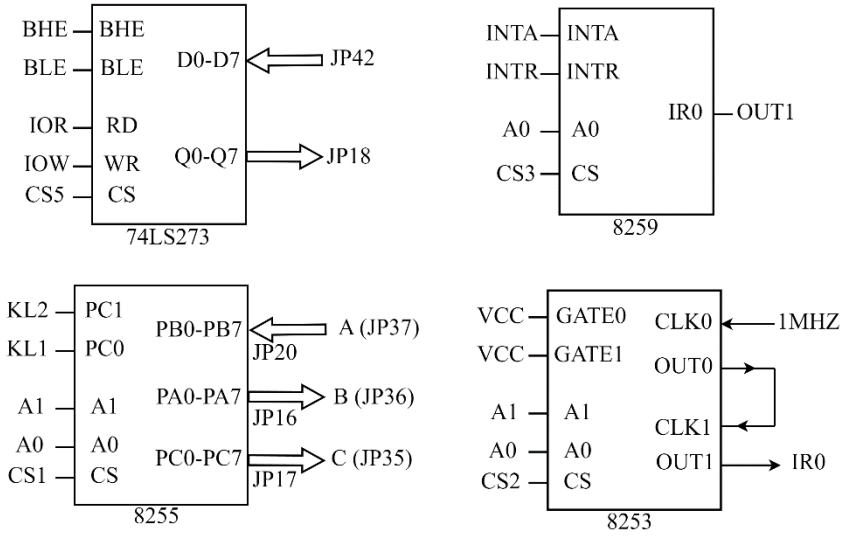

图 2 微机系统线路设计

4．微机系统程序设计

使用主循环中不断检查键盘输入，并根据输入执行相应的操作：如果按下了特定键（如 C，D，E，F），则根据键值执行相应操作，切换到正常模式、夜间模式、设置时间模式、人工模式等；否则维持在正常模式下，更新显示的交通信号灯状态。具体的模式则分别写成函数进行实现。

六、程序函数详细设计

1．函数功能说明

(1) 延时函数

程序提供了一个简单的延时函数 delay，用于实现信号灯状态切换时时间间隔的控制。

(2) 初始化函数

Init8255、Init8253、Init8259 函数用于初始化相关的硬件设备，即 8255、8253、 8259 等。

(3) 中断服务处理程序

8259 的 IR0 口对应的中断服务程序，用于处理 1s 一次的时钟中断请求，对数码管和 led

灯的状态更新或切换。

2．函数功能设计

(1) 判断交通灯状态

- 首先判断南北方向交通灯的剩余时间和东西方向交通灯的剩余时间的大小关系；
- 如果南北方向剩余时间大于东西方向，表示南北方向应该继续绿灯状态，执行相应操作；
- 如果南北方向剩余时间小于东西方向，表示东西方向应该继续绿灯状态，执行相应操作；
- 如果两者剩余时间相等，则表示应该进行黄灯状态的切换，根据 change_light 标志位来决定是从红灯到黄灯还是从黄灯到红灯的转换。

(2) 状态转换操作

根据判断结果执行相应的状态转换操作：

- 如果南北方向剩余时间大于东西方向：南北方向绿灯时间减一，并检查东西方向剩余时间是否为 1，若不为 1 则将东西方向红灯时间减一，否则表示东西方向将由黄灯切换至红灯，需要将 change_light 标志位设置为 0，并重新设置东西方向绿灯剩余时间为黄灯时间。根据当前状态控制 LED 显示，显示东西方向绿灯，南北方向红灯状态。
- 如果南北方向剩余时间小于东西方向：东西方向绿灯时间减一，并检查南北方向剩余时间是否为 1，若不为 1 则将南北方向红灯时间减一，否则表示南北方向将由黄灯切换至红灯，需要将 change_light 标志位设置为 1，并重新设置南北方向绿灯剩余时间为黄灯时间。 根据当前状态控制 LED 显示，显示东西方向红灯，南北方向绿灯状态。
- 如果南北方向剩余时间与东西方向剩余时间相等：南北方向黄灯时间和东西方向黄灯时间都减一，根据 count 的奇偶性控制黄灯闪烁的效果。 如果 change_light 为 0，则控制 LED 显示为东西方向黄灯闪烁，南北方向红灯状态；如果为 1，则显示东西方向红灯，南北方向黄灯状态。

(3) 状态未定义或结束状态处理

如果以上条件都不满足，表示当前交通灯状态未定义或已到达结束状态。

- 如果 change_light 为 0，则将南北方向绿灯剩余时间和东西方向绿灯剩余时间重新设置为正常绿灯时间，控制 LED 显示为东西方向红灯，南北方向绿灯状态。
- 如果 change_light 为 1，则将南北方向绿灯剩余时间和东西方向绿灯剩余时间重新设置为正常红灯时间，控制 LED 显示为东西方向绿灯，南北方向红灯状态。

(4) 退出中断

最后通过向 8259 主片发送命令 0x20 来退出中断。

3．特殊模式功能设计

(1) 夜间模式

n_lighting 函数实现了夜间模式的灯光闪烁效果，通过关中断来停止数码管和 LED 灯的更 新，通过 while 死循环和两个 for 循环实现灯光的闪烁，并且不断检测键盘的输入以便随时切换回正常模式。

(2) 人工模式

hn_lighting 函数实现了人工控制灯光的亮灭，通过关中断来停止数码管和 LED 灯的更 新，通过 while 死循环使 led 灯常灭，并且不断检测键盘的输入以便随时切换回正常模式。

(3) 键盘输入处理

使用 cGetKeyA 和 cGetKeyB 函数来获取键盘输入的值，根据不同的键值执行相

应的操作。

七、课程设计心得体会

在本次课程设计中，为了探索出微机系统的工作模式以及系统架构，我们参考实验手册深入了解了各个器件的功能特点、线路结构，结合系统要求，设计出了一个出色的微机系统。但在开发过程中，我们也遇到了一些问题，比如 8259 中断控制不起作用，尝试了进程中开中断，线路检查，单步调试等多种测试检验方式，最终发现是 8259 的工作方式初始化错误；另外一个较大的问题是信号灯时间显示不稳定，一直有较大延迟，我们尝试调整了时钟中断频率，以及加入了 74LS273 锁存器才使得显示输出功能更加精准稳定。

通过本次课程设计，我进一步增强了自己的开发项目工程能力，一个小型化的微机系统是以后开发大型系统的基本模型，这夯实了我的专业基础，同时结合实践，学习到的理论也得到了进一步验证，更加加深了我对微机系统的理解。

八、微机系统源码

```
1.  extern void outportb(unsigned int, char); // 写 I/O
2.  extern char inportb(unsigned int);        // 读 I/O
3.  extern int inportw(unsigned int);
4.  extern void outportw(unsigned int, int);
5.  extern void enable(void);                                  // 开中断
6.  extern void disable(void);                                 // 关中断
7.  extern void setvect(int, void interrupt (*isr)(void));     // 写中断向量
8.  extern void interrupt(far *getvect(int __interruptno))(); // 读中断向量
9.  #define u8 unsigned char
10. #define u16 unsigned int
11. // 对键盘、数码管管理器初始化
12. extern void cInitKeyDisplay();
13. // 将 pBuffer 指向的 8 字节缓冲区内容显示于 F5 区数码管上
14. extern void cDisplay8(u8 *pBuffer);
15. // 接收一个 16 进制键值，如果没有按键，返回 0xff
16. extern u8 cGetKeyA();
17. // 8259_addr
18. #define IO8259_0 0x250
19. #define IO8259_1 0x251
20. // 8253_addr
21. #define T0_8253 0x260
22. #define T1_8253 0x261
23. #define Con_8253 0x263
24. // 8255_addr
25. #define IO8255_PA 0x270
26. #define IO8255_PB 0x271
27. #define IO8255_PC 0x272
28. #define IO8255_Con 0x273
29. // 273_addr
30. #define IO273 0x230
31. u8 buffer[8]; // 显示缓冲区，8 个字节
```

```
// six light
// 定义南北方向正常情况下的交通灯时间：红灯持续 35 个单位时间，绿灯持续 30 个单位时间，黄灯持续 5 个单位时间，未定义状态持续 35 个单位时间
u8 normal_sn[4] = {35, 30, 5, 35}; // normal mode
// 定义东西方向正常情况下的交通灯时间：红灯持续 35 个单位时间，绿灯持续 30 个单位时间，黄灯持续 5 个单位时间，未定义状态持续 30 个单位时间
u8 normal_ew[4] = {35, 30, 5, 30}; // 0->red ; 1->green ; 2->yellow
// 当前东西方向交通灯剩余时间，默认同正常情况下的东西方向交通灯状态
u8 current_ew[4] = {35, 30, 5, 35};
// 当前南北方向交通灯剩余时间，默认同正常情况下的南北方向交通灯状态
u8 current_sn[4] = {35, 30, 5, 30};
u8 change_light =
    0; // 用于表示交通灯状态变化，0 表示红灯到绿灯的变化，1 表示黄灯到红灯的变化
u8 KeyResult; // 存放键值
u8 time;      // set time value
u8 k = 0;
u8 count = 0;

u8 led_data[] = {
    0xbe, // 东西绿灯，南北红灯
    0xbf, // 东西黄灯闪烁，南北红灯
    0xbd, // 东西黄灯亮，南北红灯
    0xeb, // 东西红灯，南北绿灯
    0xfb, // 东西红灯，南北黄灯闪烁
    0xdb, // 东西红灯，南北黄灯亮
    0xdd  // 东西南北黄灯闪烁

// 延时 1s 的函数
void delay(u16 ms) {
  u16 i;

  while (ms--) {
    i = 100;

    do {
      ;
    } while (--i);
  }
}

// 通过中断来管理交通灯的切换
void interrupt cur_time(void) { // 中断服务程序：倒计时及交通灯状态控制
  count++; // 每次进入中断服务程序，计数器 count 自增，用于跟踪时间
  // 如果南北方向交通灯的剩余时间大于东西方向交通灯的剩余时间，则执行以下操作
```

```
74.   if (current_sn[3] > current_ew[3]) {
75.     current_sn[3]--; // 南北方向绿灯时间减一
76.
77.     // 如果东西方向的剩余时间不为 1，表示应继续绿灯状态
78.     if (current_ew[3] != 1) {
79.       current_ew[3]--; // 东西方向红灯时间减一
80.       outportb(IO273,
81.                led_data[0]); // 控制 LED 显示为东西方向绿灯，南北方向红灯状态
82.     } else {
83.       change_light =
84.           0; // 改变灯光变化标志为 0，表示接下来南北方向将由黄灯切换至红灯
85.       current_ew[3] = current_ew[2]; // 重新设置东西方向绿灯剩余时间为黄灯时间
86.       outportb(IO273,
87.                led_data[1]); // 控制 LED 显示为东西方向黄灯闪烁，南北方向红灯状态
88.     }
89.   }
90.   // 如果南北方向交通灯的剩余时间小于东西方向交通灯的剩余时间，则执行以下操作
91.   else if (current_sn[3] < current_ew[3]) {
92.     current_ew[3]--; // 东西方向绿灯时间减一
93.
94.     // 如果南北方向的剩余时间不为 1，表示应继续绿灯状态
95.     if (current_sn[3] != 1) {
96.       current_sn[3]--; // 南北方向红灯时间减一
97.       outportb(IO273,
98.                led_data[3]); // 控制 LED 显示为东西方向红灯，南北方向绿灯状态
99.     } else {
100.          change_light =
101.              1; // 改变灯光变化标志为 1，表示接下来东西方向将由黄灯切换至红灯
102.          current_sn[3] = current_sn[2]; // 重新设置南北方向绿灯剩余时间为黄灯时间
103.          outportb(IO273,
104.                   led_data[4]); // 控制 LED 显示为东西方向红灯，南北方向黄灯闪烁状态
105.        }
106.      }
107.      // 如果南北方向交通灯的剩余时间与东西方向交通灯的剩余时间相等，并且当前状态为红灯到黄灯变化
108.      else if (current_sn[3] == current_ew[3] && current_sn[3] != 0 &&
109.               change_light == 0) {
110.        current_sn[3]--; // 南北方向黄灯时间减一
111.        current_ew[3]--; // 东西方向黄灯时间减一
112.
113.        // 通过 count 计数器的奇偶性来控制黄灯闪烁的效果
114.        if (count % 2 == 0)
115.          outportb(IO273,
116.                   led_data[1]); // 控制 LED 显示为东西方向黄灯闪烁，南北方向红灯状态
```

```
117.        if (count % 2 == 1)
118.          outportb(IO273,
119.                   led_data[2]); // 控制 LED 显示为东西方向黄灯亮，南北方向红灯状态
120.      }
121.      // 如果南北方向交通灯的剩余时间与东西方向交通灯的剩余时间相等，并且当前状态为黄灯到红灯变化
122.      else if (current_sn[3] == current_ew[3] && current_sn[3] != 0 &&
123.               change_light == 1) {
124.        current_sn[3]--; // 南北方向黄灯时间减一
125.        current_ew[3]--; // 东西方向黄灯时间减一
126.
127.        // 通过 count 计数器的奇偶性来控制黄灯闪烁的效果
128.        if (count % 2 == 0)
129.          outportb(IO273,
130.                   led_data[4]); // 控制 LED 显示为东西方向红灯，南北方向黄灯闪烁状态
131.        if (count % 2 == 1)
132.          outportb(IO273,
133.                   led_data[5]); // 控制 LED 显示为东西方向红灯，南北方向黄灯亮状态
134.      }
135.      // 如果以上条件均不满足，表示当前交通灯状态未定义或已到达结束状态
136.      else {
137.        // 如果当前状态是由红灯到绿灯状态变化
138.        if (change_light == 0) {
139.          current_sn[3] =
140.              current_sn[1]; // 重新设置南北方向绿灯剩余时间为正常绿灯时间
141.          current_ew[3] =
142.              current_ew[0]; // 重新设置东西方向绿灯剩余时间为正常绿灯时间
143.          outportb(IO273,
144.                   led_data[3]); // 控制 LED 显示为东西方向红灯，南北方向绿灯状态
145.        }
146.        // 如果当前状态是由黄灯到红灯状态变化
147.        else {
148.          current_sn[3] =
149.              current_sn[0]; // 重新设置南北方向绿灯剩余时间为正常红灯时间
150.          current_ew[3] =
151.              current_ew[1]; // 重新设置东西方向绿灯剩余时间为正常红灯时间
152.          outportb(IO273,
153.                   led_data[0]); // 控制 LED 显示为东西方向绿灯，南北方向红灯状态
154.        }
155.      }
156.      outportb(IO8259_0, 0x20);
157.    }
158.
159.    // 夜间模式
```

```
160.  void n_lighting() {
161.    u16 i;
162.    while (1) {
163.      i = 10;
164.      while (i--) {
165.        outportb(IO273, led_data[6]); // sn_ew_yellow_on
166.        delay(1);
167.        KeyResult = cGetKeyA();
168.        if (KeyResult == 12)
169.          break;
170.      }
171.      i = 10;
172.      while (i--) {
173.        outportb(IO273, 0xff); // sn_ew_yellow_off
174.        delay(1);
175.        KeyResult = cGetKeyA();
176.        if (KeyResult == 12)
177.          break;
178.      }
179.      if (KeyResult == 12)
180.        break;
181.    }
182.  }
183.
184.  // 人工模式
185.  void h_lighting() {
186.    while (1) {
187.      outportb(IO273, 0xff); // trun_off
188.      KeyResult = cGetKeyA();
189.      if (KeyResult == 12)
190.        break;
191.    }
192.  }
193.
194.  void Init8255() {
195.    outportb(IO8255_Con, 0x89); // 8255 PA,PB_out,PC_in
196.  }
197.  void Init8253() {           // 1s
198.    outportb(Con_8253, 0x35); // 计数器 T0 设置在模式 2 状态,BCD 码计数
199.    outportb(T0_8253, 0);
200.    outportb(T0_8253, 0x10); // CLK0/1000
201.    outportb(Con_8253, 0x77); // 计数器 T1 为模式 3 状态，输出方波,BCD 码计数
202.    outportb(T1_8253, 0);
203.    outportb(T1_8253, 0x10); // CLK1/1000
```

```
204.    }
205.    void Init8259() {
206.      outportb(IO8259_0, 0x13);
207.      outportb(IO8259_1, 0x8);
208.      outportb(IO8259_1, 0x9);
209.      outportb(IO8259_1, 0xfe);
210.    }
211.    void main() {
212.      Init8255();
213.      Init8253();
214.      Init8259();
215.      buffer[0] = 5;
216.      buffer[1] = 3; // sn_red_sum
217.      buffer[2] = 5;
218.      buffer[3] = 3; // ew_red_sum
219.      buffer[4] = 5;
220.      buffer[5] = 3; // init_sn_red
221.      buffer[6] = 0;
222.      buffer[7] = 3; // init_ew_green
223.      cInitKeyDisplay();
224.      outportb(IO273, 0xff);
225.      setvect(8, cur_time); // time decline 1s
226.      enable();
227.
228.      while (1) {
229.        /enable();
230.        KeyResult = cGetKeyA(); // judge keyboard
231.
232.        if (KeyResult != 0xff) { // yes -> change mode
233.          if (KeyResult == 12) {
234.            for (k = 0; k < 4; k++) {
235.              current_sn[k] = normal_sn[k];
236.              current_ew[k] = normal_ew[k];
237.            }
238.          } else if (KeyResult == 13) { // night -> yellow light
239.            disable();
240.
241.            for (k = 0; k < 8; k++)
242.              buffer[k] = 0x10;
243.
244.            cDisplay8(buffer);
245.            n_lighting();
246.          } else if (KeyResult == 14) { // set mode
247.            time = cGetKeyB();
```

```
248.             time = time * 10 + cGetKeyB();
249.             current_ew[0] = time; // ew_red
250.             time = cGetKeyB();
251.             time = time * 10 + cGetKeyB();
252.             current_sn[0] = time; // sn_red
253.             current_ew[1] = current_sn[0] - 5;
254.             current_ew[2] = 5;
255.             current_ew[3] = current_ew[1];
256.             current_sn[1] = current_ew[0] - 5;
257.             current_sn[2] = 5;
258.             current_sn[3] = current_sn[0];
259.           } else if (KeyResult == 15) {
260.             disable();
261.
262.             for (k = 0; k < 8; k++)
263.               buffer[k] = 0x10;
264.
265.             cDisplay8(buffer);
266.             h_lighting();
267.           }
268.         } else {              // no -> normal mode
269.           cDisplay8(buffer); // display number
270.           buffer[7] = current_ew[3] / 10;
271.           buffer[6] = current_ew[3] % 10;
272.           buffer[5] = current_sn[3] / 10;
273.           buffer[4] = current_sn[3] % 10;
274.           buffer[3] = current_ew[0] / 10;
275.           buffer[2] = current_ew[0] % 10;
276.           buffer[1] = current_sn[0] / 10;
277.           buffer[0] = current_sn[0] % 10;
278.         }
279.       }
280.     }
```