**实验三**

**实验目的：**

1. 熟悉独立式和行列式键盘工作原理
2. 了解实验箱的各种键盘接口方式
3. 熟练掌握独立式键盘程序设计
4. 熟练掌握行列式键盘程序设计
5. 熟悉静态扫描和动态扫描显示原理
6. 了解实验箱LED数码管的接口方式
7. 熟练掌握8位LED显示模块程序设计

**实验内容**

1） 独立键盘：使用独立按键进行0-199循环计数，SW17用于计数加，SW18用于计数减。计数从0开始，当前计数值使用实验2的显示接口方式，显示在8位LED数码管末3位。

2） 行列键盘：读取实验箱行列键盘的编码，显示在LED数码管的末位上，编码须与键盘附近标示的数字相同。

3） 学号移动：按一次sw17学号开始循环左移，再按一次停在当前位置；按一次sw18学号开始循环右移，再按一次停在当前位置。左移过程中，按右移键，不停止，直接从当前位置开始循环右移，反之亦然。学号首末位数字间空一格。

4） 时钟走时：给实验二的24小时时钟，增加键盘修改时间功能。要求：按SW17进入/退出修改，进入后时钟暂停，退出后按照新时间开始走时。修改过程中当前位闪烁，按SW18在6位时、分、秒中切换修改位，按数字键修改当前位。

**实验步骤**

**1. 实验箱的键盘布局**

实验箱提供了三种方式的键盘，独立键盘、行列键盘和AD键盘。独立键盘是指一个按键占用一根I/O口线的键盘，适用于按键较少的场合，如MP3、显示器设置键。行列键盘又称为矩阵式键盘，常用于键盘比较多的场合，如银行数字键盘、电脑键盘、数控机械控制台键盘。还有一种键盘，既有独立键盘占用口线少的优点，又有行列键盘按键数量多的优点，就是AD键盘，它使用的是一个模拟通道线，在以前单片机内没有集成ADC等模拟电路时不常见，可见于各种日本随身听，如sony Walkman, discman。

STC实验箱中，有三个独立按键，一个复位按键sw19，仅用于断电和重新上电，一般无法用于读取按键状态。另外两个按键sw17和sw18，分别接于P3.2和P3.3，实验二中使用它们的第二功能INT0和INT1产生中断，用于计数器的启停和运行方式。如果关闭中断，也可以将P3.2和P3.3当作普通I/O口来读取按键状态，判断独立按键是否被按下。尝试在关闭中断下对按键进行计数。



图1.行列式键盘



图2. 行列式键盘原理图

位于实验箱的右下角，其实物如图1，键盘接连接原理如图2：行列键盘的16个按键被连接到P0口，其中行线位于P0口的高4位，列线位于P0口的低4位，而且是不用跳线直接连接的，也就是说占用了整个P0口。原理图接在行列线上的8个300欧电阻R74~R81是为了防止端口不小心被损坏。假设P0端口被初始化为推挽方式，行线输出高电平，列线输出低电平，如果没有限流电阻，当某个按键被按下时，连接到这个按键上的两条口线就会形成尖峰贯穿电流，烧毁端口。4个上拉电阻R57~R60的作用是，与限流电阻之间形成分压，以确定正确的输入电平。

在行列键盘的上方是AD键盘。AD键盘与行列键盘最大的区别是每个按键附近都有电阻，这些电阻为分压电阻，以此确定每个按键所在位置的电压值。ADC通过读取电压值，确定按键位置。实验箱的AD键盘使用一阶RC滤波用于在一定范围内消除按键抖动。前两种按键可以通过软件方法处理串键，AD键盘天生对串键无能为力，这也是为什么一些年份比较久远的随身听串键比较严重。

**2. LED 数码管驱动方式**

实验2中已经提到，实验箱带有8位LED数码管，并提供了接口方式。本节根据其硬件连接，简要说明8位LED数码管的工作方式、工作原理以及接口软件的工作方式。实验箱上的8位LED数码管由2块3461AS 构成，3461AS是4位0.36英寸共阴极红色高亮LED数码管，其引脚如图3。



图3. 3461AS LED数码管引脚

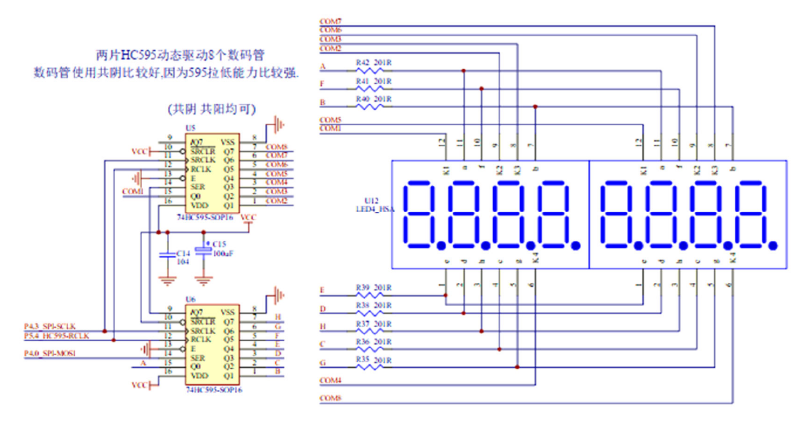
3461AS 数码管与MCU连接，构成8位LED显示器如图4。

图4. 8位LED数码管显示器

HC595芯片的引脚图、逻辑图、功能图和真值表如图5和表1所示

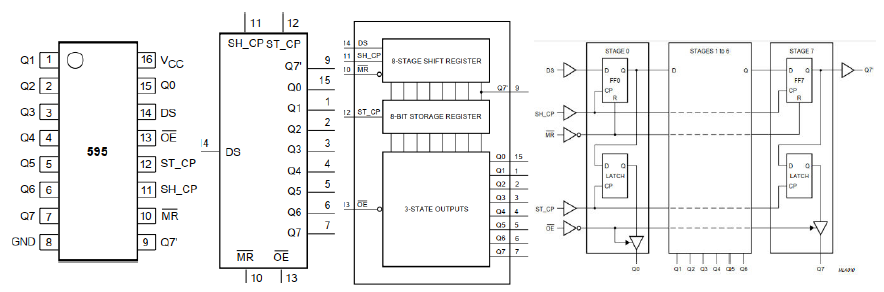
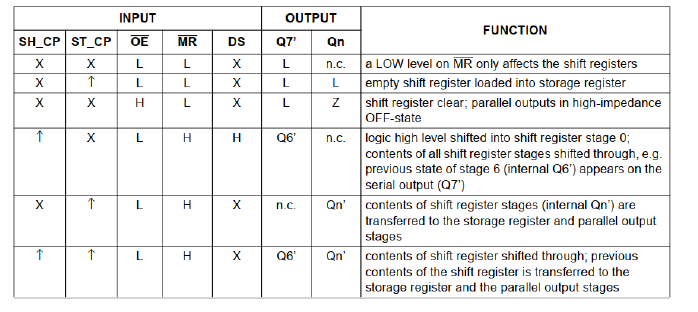


图5. HC595 移位寄存器引脚、结构和功能

表1. HC595 移位寄存器真值表



实验箱中，一方面，两片595构成的16位移位寄存器的功能是通过串行SPI总线（软件模拟）完成MCU的16位并行口扩展。U6的串行输入端SER，也就是16位移位寄存器串行数据源，为P4.0，移位寄存器的串行移位时钟端为P4.3，并行打入时钟端为P5.4。另一方面，此16位移位寄存器也担当着LED数码管显示驱动的功能。

数码管的段码驱动采用软件译码的方式，码表如表2。

表2. 3461AS 16进制数段码表



表2列出了0~F对应的段码，只要ABCDEFGH按序接到Q0~Q7位，所有共阴极LED数码管的段码都是相同的，共阳极LED需对表2的段码取反。如果要显示小数点，则需在取出段码后，将其最高位置1。实验二中，有特别说明要显示小数点，可以将编码+0x20，这是显示小数位的另一种方法。LED8.C中，段码表是这样组织的：

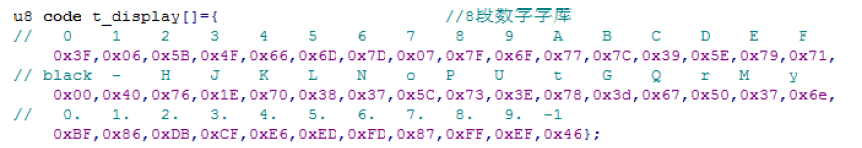


图6. LED8.c中LED显示段码表

**3. LED 数码管的驱动程序**

595移位寄存器具有两级缓冲，也就是在移位过程中，输出锁存器中的数据不会改变，作为显示驱动芯片的好处是不会因为移位引起显示紊乱或暗影。

每调用显示驱动函数一次，LED数码管显示器上就有一位得到刷新，调用8次显示驱动函数，即完成LED8位数码管的一次刷新，可称为一帧（frame）。当帧率超过50fps 的时候，一般认为，因为人眼的视觉暂留，我们看到的场景是连贯的。在实际显示中，50fps 还有一定的闪烁感。

**代码实现**

1. **200计数**
2. #include <stc15.h>
3. #include "led8.h"
4. /\*\*\*\*\*\*\*\*\*\*\*\*\* 外部变量和本地变量声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*/
5. **extern** u8 LED8[8]; //显示缓冲，在 LED8.c 中声明
6. bit F\_T0=0; //1ms 中断软标志
7. bit F\_incdec=0,F\_stop=0; //运行参数，初始为增量计数器，不停止
8. u8 Count=0; //unsigned char 与 s8,u16 在 led8.h 中定义
9. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* T0 中断初始化函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
10. **void** T0\_Init()
11. {
12. #define MAIN\_Fosc 22118400L //定义主时钟
13. #define Timer0\_Reload (65536UL -(MAIN\_Fosc / 1000))
14. AUXR = 0x80; //Timer0 set as 1T, 16 bits timer auto-reload,
15. TH0 = (u8)(Timer0\_Reload / 256);
16. TL0 = (u8)(Timer0\_Reload % 256);
17. ET0 = 1; //T0 interrupt enable
18. PT0 = 1; //设置 T0 高优先级
19. TR0 = 1; //启动 T0
20. }
21. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT0/INT1 中断初始化函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
22. **void** INT\_Init()
23. {
24. IT0 = 1; //INT0 下降沿中断
25. IT1=1;
26. IE1 = 0; //外中断 1 标志位
27. IE0 = 0; //外中断 0 标志位
28. EX1 = 1; //INT1 Enable
29. EX0 = 1; //INT0 Enable
30. EA = 1; //允许总中断
31. }
32. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 端口初始化 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
33. **void** Port\_Init()
34. {
35. P0M1 = 0; P0M0 = 0; //设置为准双向口
36. P1M1 = 0; P1M0 = 0; //设置为准双向口
37. P2M1 = 0; P2M0 = 0; //设置为准双向口
38. P3M1 = 0; P3M0 = 0; //设置为准双向口
39. P4M1 = 0; P4M0 = 0; //设置为准双向口
40. P5M1 = 0; P5M0 = 0; //设置为准双向口
41. P6M1 = 0; P6M0 = 0; //设置为准双向口
42. P7M1 = 0; P7M0 = 0; //设置为准双向口
43. }
44. **void** delay\_ms(u8 ms)
45. {
46. u16 i;
47. **do**{
48. i = MAIN\_Fosc / 16000;
49. **while**(--i) ; //16T per loop
50. }**while**(--ms);
51. }
52. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT0 中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
53. **void** INT0\_int (**void**) interrupt 0 //进中断时已经清除标志
54. {
55. F\_incdec^=1; //sw17 乒乓键，改变计数方向标志
56. delay\_ms(20); //软件延时消抖动
57. }
58. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT1 中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
59. **void** INT1\_int (**void**) interrupt 2 //进中断时已经清除标志
60. {
61. F\_stop^=1; //sw18 乒乓键，改变计数启停标志
62. delay\_ms(20); //软件延时消抖动
63. }
64. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* T0 1ms 中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
65. **void** T0\_int (**void**) interrupt 1
66. {
67. DisplayScan(); //1ms 扫描显示一位
68. F\_T0 = 1; //1ms 时间到标志
69. }
70. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 显示计数函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
71. **void** DisplayCount(**void**)
72. {
73. LED8[5] = Count/100; //百位
74. LED8[6] = (Count%100) / 10; //十位
75. LED8[7] = Count % 10; //个位
76. }
77. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 主函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
78. **void** main(**void**)
79. {
80. u16 i=0;
81. Port\_Init(); //初始化
82. T0\_Init();
83. INT\_Init();
84. **for**(i=0; i<8; i++) LED8[i] = DIS\_BLACK; //上电消隐
85. **while**(1) {
86. **if**(F\_T0) { //1ms 到
87. F\_T0 = 0;
88. **if**(F\_stop){
89. **if**(Count<200)Count++;
90. **else** Count=0;
91. }
92. **if**(F\_incdec){ //根据计数方向更新计数
93. **if**(Count==0) Count=199;
94. **else** Count--;
95. }
96. DisplayCount();
97. F\_stop=0;
98. F\_incdec=0;
99. }
101. }
102. }
103. **行列键盘**
104. /\*\*\*\*\*\*\*\*\*\*\*\*\*  本程序功能说明  \*\*\*\*\*\*\*\*\*\*\*\*\*\*
105. 显示效果为: 24小时时钟 HH-MM-SS
106. 工作模式：正常走时方式和修改时钟方式
107. 中断键，按SW17修改，按SW18切换当前位
108. 修改方式下，当前位闪烁，自动切换到位
109. 退出修改后，重新进入修改，修改位记存
110. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
111. #include <stc15.h>
112. #include "led8.h"
114. /\*\*\*\*\*\*\*\*\*\*\*\*\*  外部变量和本地变量声明    \*\*\*\*\*\*\*\*\*\*\*\*\*\*/
115. **extern** u8  LED8[8];         //显示缓冲
116. bit F\_KS=0;
118. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* T0中断初始化函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
119. **void** T0\_Init()
120. {
121. #define     MAIN\_Fosc       22118400L   //定义主时钟
122. #define     Timer0\_Reload   (65536UL -(MAIN\_Fosc / 1000))       //Timer 0 中断频率, 1000次/秒
124. AUXR = 0x80;    //Timer0 set as 1T, 16 bits timer auto-reload,
125. TH0 = (u8)(Timer0\_Reload / 256);
126. TL0 = (u8)(Timer0\_Reload % 256);
127. ET0 = 1;    //Timer0 interrupt enable
128. PT0 = 1;
129. TR0 = 1;    //Tiner0 run
130. EA = 1;     //允许总中断
131. }
133. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 端口初始化 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
134. **void** Port\_Init()
135. {
136. P0M1 = 0;   P0M0 = 0;   //设置为准双向口
137. P1M1 = 0;   P1M0 = 0;   //设置为准双向口
138. P2M1 = 0;   P2M0 = 0;   //设置为准双向口
139. P3M1 = 0;   P3M0 = 0;   //设置为准双向口
140. P4M1 = 0;   P4M0 = 0;   //设置为准双向口
141. P5M1 = 0;   P5M0 = 0;   //设置为准双向口
142. P6M1 = 0;   P6M0 = 0;   //设置为准双向口
143. P7M1 = 0;   P7M0 = 0;   //设置为准双向口
144. }
146. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* T0 1ms中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
147. **void** T0\_int (**void**) interrupt 1
148. {
149. **static** u8 Count50=50;
151. DisplayScan();  //1ms扫描显示一位
152. **if**(--Count50==0) {
153. F\_KS=1;
154. Count50 = 50;
155. }
156. }
158. **extern** u8 KeyCode;    //给用户使用的键码, 1~16有效
159. **void** Key\_Init(**void**);
160. **void** IO\_KeyScan(**void**);
162. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 主函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
163. **void** main(**void**)
164. {
165. u8 i;
167. Port\_Init();
168. T0\_Init();
169. Key\_Init();
171. **for**(i=0;i<8;i++) LED8[i] = DIS\_BLACK;     //初始熄灭
172. **while**(1) {
173. **if**(F\_KS) {              //50ms，扫描键盘
174. IO\_KeyScan();
175. F\_KS=0;
176. }
178. **if**(KeyCode > 0) {        //有键按下，处理键
179. LED8[7] = KeyCode-1;
180. KeyCode=0;
181. }
182. }
183. }
184. **学号移位**
185. #include <stc15.h>
186. #include <intrins.h>
187. #include <string.h>
188. #include "led8.h"
190. #define StrLen strlen(disp\_string)
192. /\*\*\*\*\*\*\*\*\*\*\*\*\*  外部变量和本地变量声明    \*\*\*\*\*\*\*\*\*\*\*\*\*\*/
193. **extern** u8  LED8[8];         //显示缓冲
195. bit F\_T0=0;                 //1ms中断软标志
196. bit F\_L=0, F\_R=0;           //F\_L,F\_R左右键标记
197. u8  disp\_index=0, F\_Dir=1;  //F\_Dir方向标记，0左移 1 不动 2右移
199. u8 code disp\_string[]="3200104869 ";
201. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* T0中断初始化函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
202. **void** T0\_Init()
203. {
204. #define     MAIN\_Fosc       22118400L   //定义主时钟
205. #define     Timer0\_Reload   (65536UL -(MAIN\_Fosc / 1000))       //Timer 0 中断频率, 1000次/秒
207. AUXR = 0x80;    //Timer0 set as 1T, 16 bits timer auto-reload,
208. TH0 = (u8)(Timer0\_Reload / 256);
209. TL0 = (u8)(Timer0\_Reload % 256);
210. ET0 = 1;    //Timer0 interrupt enable
211. PT0 = 1;
212. TR0 = 1;    //Tiner0 run
213. }
215. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT0/INT1中断初始化函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
216. **void** INT\_Init()
217. {
218. IT0 = 1;    //INT0 下降沿中断
219. IT1 = 1;    //INT1 下降沿中断
220. IE1  = 0;   //外中断1标志位
221. IE0  = 0;   //外中断0标志位
223. EX1 = 1;    //INT1 Enable
224. EX0 = 1;    //INT0 Enable
225. EA = 1;     //允许总中断
226. }
228. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 端口初始化 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
229. **void** Port\_Init()
230. {
231. P0M1 = 0;   P0M0 = 0;   //设置为准双向口
232. P1M1 = 0;   P1M0 = 0;   //设置为准双向口
233. P2M1 = 0;   P2M0 = 0;   //设置为准双向口
234. P3M1 = 0;   P3M0 = 0;   //设置为准双向口
235. P4M1 = 0;   P4M0 = 0;   //设置为准双向口
236. P5M1 = 0;   P5M0 = 0;   //设置为准双向口
237. P6M1 = 0;   P6M0 = 0;   //设置为准双向口
238. P7M1 = 0;   P7M0 = 0;   //设置为准双向口
239. }
241. **void** Delay20ms()        //@22.1184MHz
242. {
243. unsigned **char** i, j, k;
245. \_nop\_(); \_nop\_();
246. i = 2;
247. j = 175;
248. k = 75;
249. **do**  {
250. **do**  {
251. **while** (--k);
252. } **while** (--j);
253. } **while** (--i);
254. }
256. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT0中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
257. **void** INT0\_int (**void**) interrupt 0      //进中断时已经清除标志
258. {
259. F\_L^=1; //乒乓键
260. Delay20ms();
261. }
263. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT1中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
264. **void** INT1\_int (**void**) interrupt 2      //进中断时已经清除标志
265. {
266. F\_R^=1;     //乒乓键
267. Delay20ms(); //软件延时消抖动
268. }
270. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* T0 1ms中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
271. **void** T0\_int (**void**) interrupt 1
272. {
273. DisplayScan();  //1ms扫描显示一位
274. F\_T0 = 1;      //1ms标志
275. }
277. u8 ASC2num(u8 asc)
278. {
279. **if**(asc<='9'&&asc>='0')
280. **return** asc-'0';
281. **else** **return** DIS\_BLACK;
282. }
284. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 显示数串函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
285. **void** ReflashDisp(**void**)
286. {
287. u8 i;
289. **for**(i=0;i<8;i++)
290. LED8[i] = ASC2num(disp\_string[(disp\_index+i)%StrLen]);
291. }
293. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 主函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
294. **void** main(**void**)
295. {
296. **int** i;
298. Port\_Init();
299. T0\_Init();
300. INT\_Init();
302. ReflashDisp();    //显示学号
303. **while**(1) {
304. **if**(F\_L) {
305. **switch**(F\_Dir) {
306. **case** 0: F\_Dir=1;**break**;
307. **case** 1: F\_Dir=0; **break**;
308. **case** 2: F\_Dir=0;**break**;
309. }
310. F\_L=0;
311. }
312. **if**(F\_R) {
313. **switch**(F\_Dir) {
314. **case** 0: F\_Dir=2; **break**;
315. **case** 1: F\_Dir=2; **break**;
316. **case** 2: F\_Dir=1;**break**;
317. }
318. F\_R=0;
319. }
321. **if**(F\_T0) {
322. F\_T0=0;
323. **if**(++i >= 1000) {  //1秒到
324. i=0;
325. disp\_index=(disp\_index-F\_Dir+1+StrLen)%StrLen;
326. ReflashDisp();
327. }
328. }
329. }
330. }
331. **时钟走时**
332. #include <stc15.h>
333. #include "led8.h"
334. /\*\*\*\*\*\*\*\*\*\*\*\*\* 外部变量和本地变量声明 \*\*\*\*\*\*\*\*\*\*\*\*\*\*/
335. **extern** u8 LED8[8]; //显示缓冲，在 LED8.c 中声明
336. bit F\_T0=0; //1ms 中断软标志
337. bit F\_KS=0;
338. bit F\_incdec=0,F\_stop=0; //运行参数，初始为增量计数器，不停止
339. u8 Count=0; //unsigned char 与 s8,u16 在 led8.h 中定义
340. u8 miao=0;
341. u8 fen=0;
342. u8 a=2;
343. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* T0 中断初始化函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
344. **void** T0\_Init()
345. {
346. #define MAIN\_Fosc 22118400L //定义主时钟
347. #define Timer0\_Reload (65536UL -(MAIN\_Fosc / 1000))
348. AUXR = 0x80; //Timer0 set as 1T, 16 bits timer auto-reload,
349. TH0 = (u8)(Timer0\_Reload / 256);
350. TL0 = (u8)(Timer0\_Reload % 256);
351. ET0 = 1; //T0 interrupt enable
352. PT0 = 1; //设置 T0 我高优先级
353. TR0 = 1; //启动 T0
354. }
355. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT0/INT1 中断初始化函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
356. **void** INT\_Init()
357. {
358. IT0 = 1; //INT0 下降沿中断
359. IT1 = 1;
360. IE1 = 0; //外中断 1 标志位
361. IE0 = 0; //外中断 0 标志位
362. EX1 = 1; //INT1 Enable
363. EX0 = 1; //INT0 Enable
364. EA = 1; //允许总中断
365. }
366. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 端口初始化 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
367. **void** Port\_Init()
368. {
369. P0M1 = 0; P0M0 = 0; //设置为准双向口
370. P1M1 = 0; P1M0 = 0; //设置为准双向口
371. P2M1 = 0; P2M0 = 0; //设置为准双向口
372. P3M1 = 0; P3M0 = 0; //设置为准双向口
373. P4M1 = 0; P4M0 = 0; //设置为准双向口
374. P5M1 = 0; P5M0 = 0; //设置为准双向口
375. P6M1 = 0; P6M0 = 0; //设置为准双向口
376. P7M1 = 0; P7M0 = 0; //设置为准双向口
377. }
378. **void** delay\_ms(u8 ms)
379. {
380. u16 i;
381. **do**{
382. i = MAIN\_Fosc / 16000;
383. **while**(--i) ; //16T per loop
384. }**while**(--ms);
385. }
386. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT0 中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
387. **void** INT0\_int (**void**) interrupt 0 //进中断时已经清除标志
388. {
389. F\_incdec^=1; //sw17 乒乓键，改变计数方向标志
390. delay\_ms(20); //软件延时消抖动
391. }
392. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INT1 中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
393. **void** INT1\_int (**void**) interrupt 2 //进中断时已经清除标志
394. {
395. F\_stop^=1; //sw18 乒乓键，改变计数启停标志
396. **if**(F\_incdec)
397. **if**(++a==8) a=2;
398. delay\_ms(20); //软件延时消抖动
399. }
401. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* T0 1ms中断函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
402. **void** T0\_int (**void**) interrupt 1
403. {
404. **static** u8 Count50=50;
406. DisplayScan();  //1ms扫描显示一位
407. **if**(--Count50==0) {
408. F\_KS=1;
409. Count50 = 50;
410. }
411. F\_T0 = 1;
412. }
413. **extern** u8 KeyCode;    //给用户使用的键码, 1~16有效
414. **void** Key\_Init(**void**);
415. **void** IO\_KeyScan(**void**);
417. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 显示计数函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
418. **void** DisplayCount(**void**)
419. {
420. LED8[2] = Count/10;
421. LED8[3] = Count%10+32;
422. LED8[4] = fen/10;
423. LED8[5] = fen%10+32;
424. LED8[6] = miao/10;
425. LED8[7] = miao%10;
426. }
427. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 主函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
428. **void** main(**void**)
429. {
430. u16 i=0;
431. Port\_Init(); //初始化
432. T0\_Init();
433. INT\_Init();
434. **for**(i=0; i<8; i++) LED8[i] = DIS\_BLACK; //上电消隐
435. **while**(1) {
436. **if**(F\_T0) { //1ms 到
437. F\_T0 = 0;
438. **if**(!F\_incdec){
440. **if**(++i >= 1000){//1s到
441. i=0;
442. **if**(++miao==60){
443. miao=0;
444. **if**(++fen==60){
445. fen=0;
446. **if**(++Count==24) Count=0;
447. }
448. }
449. }
450. DisplayCount();
451. F\_KS=0;
452. }
453. **else**{
454. **if**(++i<=500)LED8[a]=0x10;
455. **else** **if**(i<=1000)DisplayCount();//实现闪烁
456. **else** i=0;
457. **if**(F\_KS) {              //50ms，扫描键盘
458. IO\_KeyScan();
459. F\_KS=0;
460. }
461. **if**(KeyCode > 0&&KeyCode<=10){
462. **switch**(a){
463. **case** 2: **if**(Count%10+(KeyCode-1)\*10<24) Count=Count%10+(KeyCode-1)\*10;**break**;
464. **case** 3: **if**(Count/10\*10+(KeyCode-1)<24) Count=Count/10\*10+(KeyCode-1);**break**;
465. **case** 4: **if**(fen%10+(KeyCode-1)\*10<60) fen=fen%10+(KeyCode-1)\*10;**break**;
466. **case** 5: **if**(fen/10\*10+(KeyCode-1)<60)fen=fen/10\*10+(KeyCode-1);**break**;
467. **case** 6: **if**(miao%10+(KeyCode-1)\*10<60) miao=miao%10+(KeyCode-1)\*10;**break**;;**break**;
468. **case** 7: **if**(miao/10\*10+(KeyCode-1)<60) miao=miao/10\*10+(KeyCode-1);**break**;;**break**;
469. }
470. KeyCode=0;
471. }
472. }
473. }
474. }
475. }

**实验结果（部分截图）**

**1 **

图7. 200计数

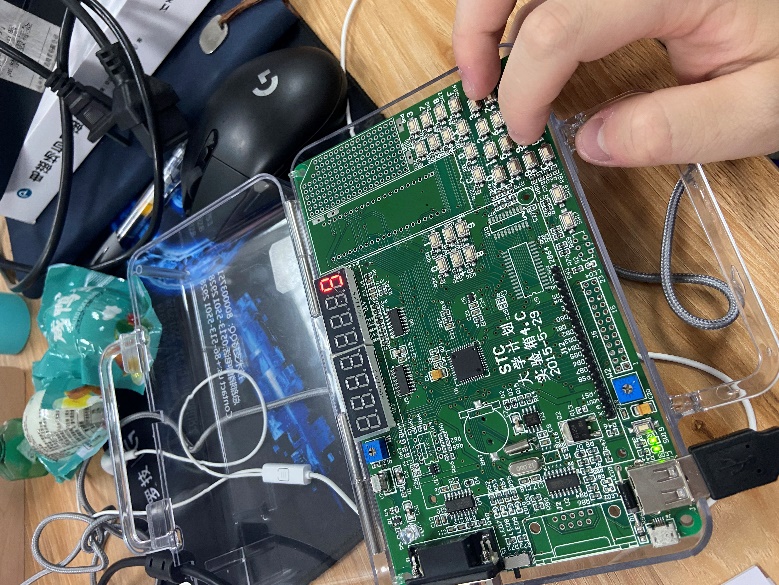
** **

图8. 行列式键盘

****

图9. 学号移位

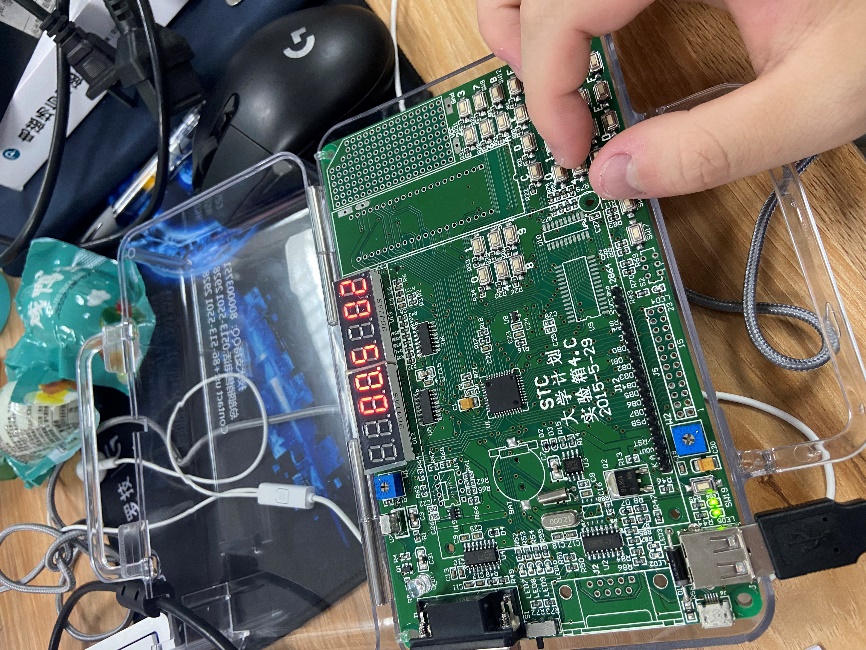
****

图10. 时钟走时

**练习与思考：**

1. 你还熟悉哪些人机接口输入方式？试查阅编码器飞梭原理。

在电子产品设计中，经常会用到旋转编码开关，也就是所说的旋转编码器、数码电位器、Rotary Encoder 。它具有左转，右转功能，有的旋转编码开关还有按下功能。为了了解旋转开关的编程，以EC11型编码开关为例,介绍一点原理和使用方法：

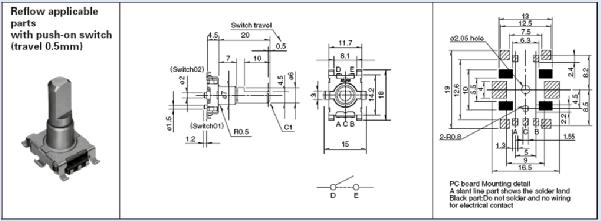


图11. 旋转编码器原理

二只脚：这边是按压式开关，按下通，松开断。

三只脚：1 2 3脚一般是中间2脚接地，1、3脚上拉电阻后，当左转、右转旋转时，在1、3脚就有脉冲信号输出了。在单片机编程时，左转和右转的判别是难点,用示波器观察这种开关左转和右转时两个输出脚的信号有个相位差，如图2。 由此可见，如果输出1为低电平时，输出2出现一个高电平，这时开关就是向顺时针旋转；当输出1 为高电平，输出2出现一个低电平，这时就一定是逆时针方向旋转。所以，在单片机编程时只需要判断当输出1为高电平时，输出2当时的状态就可以判断出是左旋转或是右旋转了。1脚与2脚同时为高电平时，说明开关没有动作。

1. 你还熟悉哪些人机接口输出方式？试查阅OLED工作原理。

OLED像素电路工作原理：相比LCD，OLED除了开关管T1之外，还多了控制管T2



图12. OLED原理

寻址信号Gate，加载到SW\_TFT(T1)的栅极，控制它的导通/开关管T1；

数据信号Source，通过T1，加载到DRV\_TFT(T2)的栅极，控制流过OLED的电流，实现不同灰阶度/控制管T2；

储存电容Cst：Source信号将保持在T2栅极持续到下一次寻址。

1. 实验1独立键盘查询方式与实验二使用的外中断的异同？

单片机在操作外部设备时，常用的有中断和查询两种方式。除了在编程方面的区别外，在性能和效率上都是有所区别。中断的性能要比查询强大，反应速度快，要求相应的ISR不能过于繁琐，而且要求电路板制作的水平要高，不易受干扰，否则一个小的干扰脉冲将造成误动作。而查询方式可以采用软件滤波的方法滤除此类干扰，但反应速度要慢于中断，因此要根据实际需要进行选择。

1. 根据动态显示原理，分析酒店门头屏移动显示的工作原理。

和本次实验内容基本一致，不断对屏幕进行刷新，利用人眼视觉暂留的原理，实现移动显示的功能。