


```
72 sbit output2 = P1^2;
73 sbit output3 = P1^3;
74 sbit output4 = P1^4;
75
76 sbit buzzer = P3^0;
77 sbit xbuzzer = P3^1;
78
79 sbit key1 = P3^2;
80 sbit key2 = P3^3;
81 sbit key3 = P3^4;
82 sbit key4 = P3^5;
83
84 #define on 0
85 #define off 1
86
87 /* mask byte of flag1 for bit testing */
88
89 #define mode 0x40          /* swap number entry mode */
90 #define disabling_key 0x80 /* disable consecutive key pressing */
91
92 // prototype declaration
93 void blink_timer1();
94 void blink_timer2();
95 void blink_timer3();
96 void blink_timer4();
97
98 char code convert[10] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
99 // LED pattern for 0 1 2 3 4 5 6 7 8 9
100
101 // write 64-bit data to 4094 x 8, COMS shift regsiter
102 // The output of each register was tied to common cathod 7-segment LED
103 // data was shifted with rising edge, 0-to-1 CLK signal
104 // all bits are sent to the output with 1-to-0 STROBE signal
105
106 void write_led()
107 {
108     char i,j;
109
110     for(j=0; j<8; j++)
111     {
112         for(i=0;i<8; i++)
113         {
114             if(buffer[j]&0x80) DIN = 1; // set bit data line
115             else DIN=0; // clear bit data line
116             CLK= 1; // make clock high
117             buffer[j] <<=1; // shift 1 bit left
118             CLK = 0; // make clock low
119         }
120     }
121     strobe = 1; // make strobe high
122     ;
123     strobe = 0; // then low
124 }
125
126
127 void blink()
128 {
129     if(flag1&0x02) blink_timer1();
130     if(flag1&0x08) blink_timer2();
131     if(flag1&0x10) blink_timer3();
132     if(flag1&0x20) blink_timer4();
133 }
134
135 offmsd() // turn msd off
136 {
137     if (buffer[1] == 0x3f) buffer[1] = 0x00;
138     if (buffer[3] == 0x3f) buffer[3] = 0x00;
139     if (buffer[5] == 0x3f) buffer[5] = 0x00;
140     if (buffer[7] == 0x3f) buffer[7] = 0x00;
141 }
142
```

```
143 void moveTimetoBuffer() /* convert 8-bit binary to 7-segment 8-digit */
144
145 {
146     if(timer1 != -1)
147     {
148         buffer[7] = convert[timer1/10];
149         buffer[6] = convert[timer1%10];
150
151         if(flag2&0x01) buffer[6] |= 0x80;
152         else buffer[6] &= ~0x80;
153     }
154     else
155     {
156         buffer[7]=buffer[6]= 0x40;
157     }
158
159     if(timer2 != -1)
160     {
161         buffer[5] = convert[timer2/10];
162         buffer[4] = convert[timer2%10];
163
164         if(flag2&0x02) buffer[4] |= 0x80;
165         else buffer[4] &= ~0x80;
166     }
167     else
168     {
169         buffer[5]=buffer[4]= 0x40;
170     }
171
172     if(timer3 != -1)
173     {
174         buffer[3] = convert[timer3/10];
175         buffer[2] = convert[timer3%10];
176
177         if(flag2&0x04) buffer[2] |= 0x80;
178         else buffer[2] &= ~0x80;
179     }
180     else
181     {
182         buffer[3] = 0x40;    // --
183         buffer[2] = 0x40;
184     }
185
186     if(timer4 != -1)
187     {
188         buffer[1] = convert[timer4/10];
189         buffer[0] = convert[timer4%10];
190         if(flag2&0x08) buffer[0] |= 0x80;
191         else buffer[0] &= ~0x80;
192     }
193     else
194     {
195         buffer[1] = 0x40;    // --
196         buffer[0] = 0x40;
197     }
198     offmsd(); // turn off msd, e.g. 09 to 9, say
199
200 }
201
202
203 updatedisplay()
204
205 {
206     moveTimetoBuffer();
207     blink();
208     write_led();
209 }
210
211 all_digit_off()
212 {
213     char i;
```

```
214     for(i=0; i<8; i++)
215         buffer[i]=0;
216     write_led();
217 }
218
219 shutdown()
220 {
221     if((timer1&timer2&timer3&timer4)== -1)
222     {
223         timer7++;
224         if(timer7 >= 10) // 10 seconds timeout
225         {
226             all_digit_off();
227             EA = 0; // disable interrupt
228             PCON = 2; // enter power down mode now, press reset to wake up
229         }
230     }
231 }
232
233 // timer functions run every one second
234
235 void run_timer1(void)
236 {
237     if(timer1 != -1)
238     {
239         if(timer1 != 0)
240         {
241             timer7 = 0; // reset timeout
242             buzzer1 = 0;
243             flag1 |= 0x02;
244             output1=1;
245
246             if(++timer1_clk >= 60) // timer1 is one min based!
247             {
248                 timer1_clk = 0;
249                 timer1--;
250             }
251         }
252     }
253     else // timer1 == 0 then fire output
254     {
255         output1=0; // fire output1
256         buzzer1 = 1;
257         flag1 &= ~0x02;
258         flag2 &= ~0x01;
259     }
260 }
261 else
262 {
263     flag1 &= ~0x02; // no blink when timer1 == -1
264     flag2 &= ~0x01;
265     buzzer1 = 0;
266 }
267 }
268
269 void run_timer2()
270 {
271     if(timer2 != -1)
272     {
273         if(timer2 != 0)
274         {
275             timer7 = 0;
276             buzzer2 = 0;
277             flag1 |= 0x08;
278             output2=1;
279
280             if(++timer2_clk >= 60) //timer2 is one min based!
281             {
282                 timer2_clk = 0;
283                 timer2--;
284             }

```

```
285     }
286     else // timer2 == 0 then fire output
287     {
288         output2=0; // fire output1
289         // putchar('2');
290         buzzer2 = 1;
291         flag1 &= ~0x08;
292         flag2 &= ~0x02;
293     }
294 }
295 else
296 {
297     flag1 &= ~0x08; // no blink when timer1 == -1
298     flag2 &= ~0x02;
299     buzzer2 = 0;
300 }
301 }
302
303 void run_timer3()
304 {
305     if(timer3 != -1)
306     {
307         if(timer3 != 0)
308         {
309             timer7 = 0;
310             flag1 |= 0x10;
311             output3=1;
312
313             if(++timer3_clk >= 3600) // timer3 is one hour based!
314             {
315                 timer3_clk = 0;
316                 timer3--;
317             }
318         }
319         else // timer1 == 0 then fire output
320         {
321             output3=0; // fire output1
322             // putchar('3');
323             flag1 &= ~0x10;
324             flag2 &= ~0x04;
325         }
326     }
327     else
328     {
329         flag1 &= ~0x10; // no blink when timer1 == -1
330         flag2 &= ~0x04;
331     }
332 }
333
334 // I modified timer4 for sdcc testing and for my wife mixer delay off function!
335
336 void run_timer4()
337 {
338     if(timer4 != -1)
339     {
340         if(timer4 != 0)
341         {
342             timer7 = 0;
343             flag1 |= 0x20;
344             output4=0; // turn on when run!
345
346             if(++timer4_clk >= 60) // timer4 is one min based!
347             {
348                 timer4_clk = 0;
349                 timer4--;
350             }
351         }
352         else // timer4 == 0 then fire output
353         {
354             output4=1; // turn off when timeover!
355             flag1 &= ~0x20;
```

```
356         flag2 &= ~0x08;
357     }
358 }
359 else
360 {
361     flag1 &= ~0x20; // no blink when timer1 == -1
362     flag2 &= ~0x08;
363     output4= 1; // turn output off
364 }
365 }
366
367 run_timer()
368 {
369     if(++one_sec>=100)
370     {
371         one_sec = 0;
372         run_timer1();
373         run_timer2();
374         run_timer3();
375         run_timer4();
376         shutdown(); // run shutdown checking every second
377     }
378 }
379
380 // reload preset time value from preset array for each key pressed
381
382 void set_timer()
383 {
384     if((flag1&1) == 0) // enter only when keys have been released
385     {
386         if(!key4) // if key 1 has been pressed
387         {
388             flag1 |= 1;
389             if(index1>=10) index1 =0;
390             timer1= preset_timer1[index1++];
391         }
392         if(!key3)
393         {
394             flag1 |= 1;
395             if(index2>=10) index2 =0;
396             timer2= preset_timer2[index2++];
397         }
398
399         if(!key2)
400         {
401             flag1 |= 1;
402             if(index3>=14) index3 =0;
403             timer3= preset_timer3[index3++];
404         }
405         if(!key1)
406         {
407             flag1 |= 1;
408             if(index4>=10) index4 =0;
409             timer4= preset_timer4[index4++];
410         }
411     }
412 }
413
414 }
415
416 void key_release()
417 {
418     if((P3&0x3c) == 0x3c)
419         flag1 &= ~1;
420 }
421
422 void blink_timer1()
423 {
424     ++blink1;
425     if(blink1 >= 20) flag2 &= ~0x01;
426 }
```

```
427         if( blink1 >= 100)
428             {
429                 flag2 |= 0x01;
430                 blink1 = 0;
431             }
432     }
433
434 void blink_timer2()
435 {
436     ++blink2;
437     if(blink2 >= 20) flag2 &= ~0x02;
438
439     if( blink2 >= 100)
440         {
441             flag2 |= 0x02;
442             blink2 = 0;
443         }
444 }
445
446 void blink_timer3()
447 {
448     ++blink3;
449     if(blink3 >= 20) flag2 &= ~0x04;
450
451     if( blink3 >= 200)
452         {
453             flag2 |= 0x04;
454             blink3 = 0;
455         }
456 }
457
458 void blink_timer4()
459 {
460     ++blink4;
461     if(blink4 >= 20) flag2 &= ~0x08;
462
463     if( blink4 >= 100) // <-- change for minute based blinking!
464         {
465             flag2 |= 0x08;
466             blink4 = 0;
467         }
468 }
469
470
471 void enable_buzzer(char i){
472     if (i) buzzer=0;
473     else buzzer=1;
474 }
475
476 void enable_buzzer2(char i){
477     if (i) xbuzzer=0;
478     else xbuzzer=1;
479 }
480
481 void counting_down2(char j){
482     timer5--;
483     if (timer5 == 0){
484         index5+=2;
485         if (index5 == j)
486             index5 = 0;
487         load_timer5 = 0;
488     }
489 }
490 }
491
492 void counting_down3(char j){
493     timer6--;
494     if (timer6 == 0){
495         index6+=2;
496         if (index6 == j)
497             index6 = 0;
```

```
498             load_timer6 = 0;
499
500             }
501     }
502 void ring1()
503 {
504     if (buzzer1){
505         if(load_timer5 == 0){
506             enable_buzzer(busy_pattern[index5]);
507             timer5 = busy_pattern[index5+1];
508             load_timer5 = 1;
509         }
510         counting_down2(4);
511     }
512     else buzzer=1;
513 }
514 }
515
516 void ring2()
517 {
518     if (buzzer2){
519         if(load_timer6 == 0){
520             enable_buzzer2(busy_pattern2[index6]);
521             timer6 = busy_pattern2[index6+1];
522             load_timer6 = 1;
523         }
524         counting_down3(4);
525     }
526     else xbuzzer=1;
527 }
528
529 // enter timer interrupt every 10ms
530 void timer_isr(void) interrupt 1 using 1
531 {
532     TH0 |= 0xe5; // reload value with 8MHz Xtal
533     TL0 |= 0xf5;
534     cputick++;
535 }
536
537 main()
538 {
539     flag1 = flag2 = 0;
540     P1 = P3 = 0xFF;
541     TMOD = 0x01; // timer0 mode 1
542     EA = ET0 = TR0 = 1; // enable timer0 interrupt, start timer
543     timer1= timer2=timer3=timer4 =-1;
544     timer1_clk=timer2_clk=timer3_clk=timer4_clk=0;
545     index1=index2=index3=index4=index5=index6=0;
546     load_timer5 = load_timer6=0;
547     buzzer1=buzzer2=0;
548     timer7 = 0;
549
550     while(1)
551     {
552         while(!cputick) // run following tasks every 10ms
553             ;
554         cputick = 0;
555         set_timer();
556         run_timer();
557         key_release();
558         updatedisplay();
559         ring1();
560         ring2();
561     }
562 }
563
```