

Embedded system experiment report #1: Clock

Hong Chenhui^a

^a*DrRedTheRed*

Professor/You Wang

Abstract—A rudimentary digital clock with timer and alarm functionalities, adjustable via a mere three buttons. Powered by AT89S52.

keywords—*L^AT_EX* No.

Contents

1	Task description	1
2	Code structure	1
3	Implementation	1
3.1	Hardware	1
	Items • Wiring	
3.2	Manual	1
3.3	Code	1

1. Task description

Develop a real-time clock capable of displaying on a digital display, adjustable via a minimal number of buttons, not exceeding three. Please establish a logical button configuration. The system should feature programmable timer and alarm functions, indicated by a buzzer. The experiment will be conducted using a laboratory kit.

2. Code structure

Nothing particularly interesting on code structure. Primitive implementation as follows.

1. **Main loop:** contains the main program, where
 - The keys undergo a scanning process to procure the input slated for processing.
 - The alarm or timer is engaged to initiate the activation of the buzzer.
 - The digital displays are configured.
2. **Interrupt:** contains the interrupt service routine, with
 - The timer 0 and 1.
 - The countdown.
 - The display refreshes.

3. Implementation

3.1. Hardware

3.1.1. Items

1. STC89C51 chip
2. Buzzer
3. Digital Display
4. 74HC138 line decoder
5. Key module

3.1.2. Wiring

- STC89C51 P2.2-P2.4 → 74HC138 C-A
- STC89C51 P0 → Digital Display
- STC89C51 P1.5 → Buzzer J7
- STC89C51 P3.0-P3.2 → Key module K1-K3

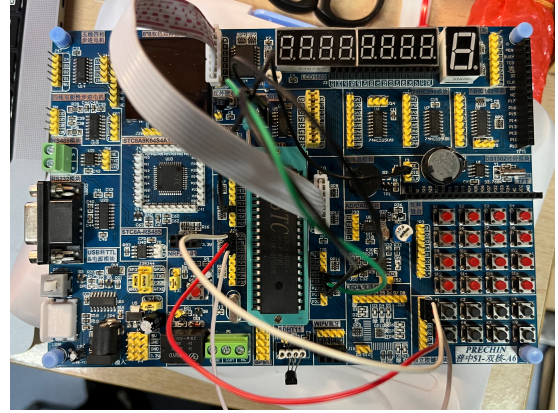


Figure 1. Board layout.

3.2. Manual

- **K1:** Mode selection.

Table 1. Mode division

Clock(1-3)	Alarm(4-6)	Countdown(7-9)
Hour(1)	Alarm.hour(4)	Countdown.hour(7)
Minute(2)	Alarm.minute(5)	Countdown.minute(8)
Second(3)	Alarm.second(6)	Countdown.second(9)

Note: Default mode is set to hours, each press of the K1 key increments sequentially

- **K2:** Each press increases the mode number incrementally.
- **K3:** Activate/Deactivate the alarm/countdown in corresponding mode.

3.3. Code

The code could roughly be divided into two parts: the main loop and the two interrupt. The main loop is responsible for the main program, while both interrupts is responsible for the interrupt service routine.

```
1 #include<reg52.h>
2
3 #define KeyPort P3
4 #define u8 unsigned char //but actually nobody use it
5 #define u16 unsigned int //same tbh
6
7 /*-----
8                      Proclaims
9 -----*/
10
11 sbit LSA=P2^2;
12 sbit LSB=P2^3;
13 sbit LSC=P2^4;
14
15 sbit ALA=P1^5; //ports
16
17 u8 dis=0; //used to change the display number in digital
18           display
19
20 unsigned char hour,minute,second;
21
22 bit UpdateTimeFlag;
23
24 unsigned char code dofly_DuanMa[10]={0x3f,0x06,0x5b,0
25                                     x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
26 unsigned char code dofly_WeiMa[]={0xfe,0xfd,0xfb,0xf7,0
27                                   xef,0xdf,0xbf,0x7f};
```

```

26 u8 code smgduan[17]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d
    ,0x07,
27     0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};
28
29 unsigned char TempData[8],Key_Num;
30
31 /*My proclaims*/
32 enum MODE{HOUR,MINUTE,SECOND,ALARM_H,ALARM_M,ALARM_S,
    COUNT_H,COUNT_M,COUNT_S};
33
34 enum MODE mode = HOUR;
35
36 struct TIMESTRUCT{
37     unsigned char hour;
38     unsigned char minute;
39     unsigned char second;
40 };
41
42 struct TIMESTRUCT alarm = {0,0,0};
43
44 struct TIMESTRUCT count = {0,0,0};
45
46 enum CONFIRM{NO,YES};
47
48 enum CONFIRM confirm = NO;
49
50 enum ACTIVATE{NO_1,YES_1};
51
52 enum ACTIVATE alarm_activate = NO_1;
53 enum ACTIVATE count_activate = NO_1;
54 enum ACTIVATE buzzer_activate = NO_1;
55 /*My proclaims ended*/
56
57 void countdown();
58 void playChineseTea();
59
60 void DelayUsNo(unsigned int t); //same as the next one
61 void DelayUs2x(unsigned char t); //Delay on microsecond
62 void DelayMs(unsigned char t); //Delay on millisecond
63 void DigDisplay(); //Digital display
64 unsigned char KeyScan(void); //Keyboard scan
65 void Init_Timer0(void); //Timer initialization
66 /*-----
67     Main functions
68 -----*/
69 void delay(u16 i)
70 {
71     while(i--);
72 }
73 /*-----
74 void main (void)
75 {
76     Init_Timer0();
77
78     while (1)
79     {
80         int i = 0;
81
82         Key_Num=KeyScan();
83         switch(Key_Num)
84         {
85             case 1:
86                 mode++;
87                 mode = (mode%9);
88                 break;
89             case 2:
90                 if(mode == HOUR)
91                 {
92                     hour++;if(hour==24)hour=0;
93                 }else if(mode == MINUTE)
94                 {
95                     minute++;if(minute==60)minute=0;
96                 }else if(mode == SECOND)
97                 {
98                     second++;if(second==60)second=0;
99                 }else if(mode == ALARM_H)
100                 {
101                     alarm.hour++;if(alarm.hour==24)
102                         alarm.hour=0;
103                 }else if(mode == ALARM_M)
104                 {
105                     alarm.minute++;if(alarm.minute
106                         ==60)alarm.minute=0;
107                 }else if(mode == ALARM_S)
108                 {
109                     alarm.second++;if(alarm.second
110                         ==60)alarm.second=0;
111                 }else if(mode == COUNT_H)
112                 {
113                     count.hour++;if(count.hour==24)
114                         count.hour=0;
115                 }
116             }
117         }
118     }
119 }

```

```

111     }else if(mode == COUNT_M)
112     {
113         count.minute++;if(count.minute
114             ==60)count.minute=0;
115     }else if(mode == COUNT_S)
116     {
117         count.second++;if(count.second
118             ==60)count.second=0;
119     }
120     break;
121     case 3:confirm = YES;
122     break;
123     default:break;
124 }
125
126 if(mode == HOUR || mode == MINUTE || mode
127 == SECOND)
128 {
129     TempData[0]=dofly_DuanMa[hour/10];
130     TempData[1]=dofly_DuanMa[hour%10];
131     TempData[2]=0x40;
132     TempData[3]=dofly_DuanMa[minute/10];
133     TempData[4]=dofly_DuanMa[minute%10];
134     TempData[5]=0x40;
135     TempData[6]=dofly_DuanMa[second/10]; //
136     Store the time
137     TempData[7]=dofly_DuanMa[second%10];
138 }
139 else if (mode == ALARM_H || mode == ALARM_M
140 || mode == ALARM_S )
141 {
142     TempData[0]=dofly_DuanMa[alarm.hour/10];
143     TempData[1]=dofly_DuanMa[alarm.hour%10];
144     TempData[2]=0x40;
145     TempData[3]=dofly_DuanMa[alarm.minute
146 /10];
147     TempData[4]=dofly_DuanMa[alarm.minute
148 %10];
149     TempData[5]=0x40;
150     TempData[6]=dofly_DuanMa[alarm.second
151 /10];
152     TempData[7]=dofly_DuanMa[alarm.second
153 %10];
154 }
155 else if (mode == COUNT_H || mode == COUNT_M
156 || mode == COUNT_S )
157 {
158     TempData[0]=dofly_DuanMa[count.hour/10];
159     TempData[1]=dofly_DuanMa[count.hour%10];
160     TempData[2]=0x40;
161     TempData[3]=dofly_DuanMa[count.minute
162 /10];
163     TempData[4]=dofly_DuanMa[count.minute
164 %10];
165     TempData[5]=0x40;
166     TempData[6]=dofly_DuanMa[count.second
167 /10];
168     TempData[7]=dofly_DuanMa[count.second
169 %10];
170 }
171
172 if(confirm == YES)
173 {
174     confirm = NO;
175     if(mode == ALARM_H || mode == ALARM_M ||
176 mode == ALARM_S)
177     {
178         alarm_activate = !alarm_activate;
179     }else if(mode == COUNT_H || mode ==
180 COUNT_M || mode == COUNT_S)
181     {
182         count_activate = !count_activate;
183     }
184     mode = HOUR;
185 }
186
187 if( alarm_activate == YES_1 && alarm.hour
188 == hour && alarm.minute == minute && alarm.second
189 == second)
190 {
191     buzzer_activate = YES_1;
192 }
193
194 if(buzzer_activate == YES_1)
195 {
196     buzzer_activate = !buzzer_activate;
197
198     playChineseTea();
199 }

```

```

184         //DigDisplay();
185     }
186 }
187 }
188
189 /*-----
190     Countdown is called in Interrupt
191 -----*/
192
193 void countdown()
194 {
195     static unsigned int num;
196
197     if( count_activate == YES_1 && count.hour == 0 &&
198         count.minute == 0 && count.second == 0)
199     {
200         count_activate = !count_activate;
201
202         buzzer_activate = YES_1;
203
204         mode = HOUR;
205     }
206
207     if(count_activate == YES_1)
208     {
209         num++;
210         if(num==500)           //Roughly 1 second
211         {
212             num=0;
213             count.second--;
214             if(count.second==255)
215             {
216                 count.second = 59;
217                 count.minute--;
218                 if(count.minute==255)
219                 {
220                     count.minute=59;
221                     count.hour--;
222                 }
223             }
224         }
225     }
226
227 /*-----
228     Yes I know this could be done better but idc.
229     This is a song in Touhou Project 6: Embodiment of
230     Scarlet Devil. After writing this the program
231     reached its memory limit.
232 -----*/
233
234 void playChineseTea()
235 {
236     int i;
237     //standard 1 note: 33800
238     for( i = 0; i< 150 ; i++ )
239     {
240         ALA = 1;
241         DelayUsNo(675/2);
242         ALA = 0;
243         DelayUsNo(675/2);
244     }
245     DelayMs(20);
246
247     for( i = 0; i< 100 ; i++ )
248     {
249         ALA = 1;
250         DelayUsNo(338/2);
251         ALA = 0;
252         DelayUsNo(338/2);
253     }
254     DelayMs(20);
255
256     for( i = 0; i< 134 ; i++ )
257     {
258         ALA = 1;
259         DelayUsNo(379/2);
260         ALA = 0;
261         DelayUsNo(379/2);
262     }
263     DelayMs(20);
264
265     for( i = 0; i< 179 ; i++ )
266     {
267         ALA = 1;
268         DelayUsNo(284/2);
269         ALA = 0;
270         DelayUsNo(284/2);
271     }
272     DelayMs(20);
273

```

```

274     for( i = 0; i< 112 ; i++ )
275     {
276         ALA = 1;
277         DelayUsNo(301/2);
278         ALA = 0;
279         DelayUsNo(301/2);
280     }
281     DelayMs(20);
282
283     for( i = 0; i< 88 ; i++ )
284     {
285         ALA = 1;
286         DelayUsNo(379/2);
287         ALA = 0;
288         DelayUsNo(379/2);
289     }
290     DelayMs(20);
291
292     for( i = 0; i< 74 ; i++ )
293     {
294         ALA = 1;
295         DelayUsNo(451/2);
296         ALA = 0;
297         DelayUsNo(451/2);
298     }
299     DelayMs(20);
300
301     for( i = 0; i< 88 ; i++ )
302     {
303         ALA = 1;
304         DelayUsNo(379/2);
305         ALA = 0;
306         DelayUsNo(379/2);
307     }
308     DelayMs(20);
309
310     for( i = 0; i< 44 ; i++ )
311     {
312         ALA = 1;
313         DelayUsNo(379/2);
314         ALA = 0;
315         DelayUsNo(379/2);
316     }
317     DelayMs(20);
318
319     for( i = 0; i< 550 ; i++ )
320     {
321         ALA = 1;
322         DelayUsNo(338/2);
323         ALA = 0;
324         DelayUsNo(338/2);
325     }
326     DelayMs(20);
327     //1st sentence
328     for( i = 0; i< 150 ; i++ )
329     {
330         ALA = 1;
331         DelayUsNo(675/2);
332         ALA = 0;
333         DelayUsNo(675/2);
334     }
335     DelayMs(20);
336
337     for( i = 0; i< 100 ; i++ )
338     {
339         ALA = 1;
340         DelayUsNo(338/2);
341         ALA = 0;
342         DelayUsNo(338/2);
343     }
344     DelayMs(20);
345
346     for( i = 0; i< 142 ; i++ )
347     {
348         ALA = 1;
349         DelayUsNo(358/2);
350         ALA = 0;
351         DelayUsNo(358/2);
352     }
353     DelayMs(20);
354
355     for( i = 0; i< 150 ; i++ )
356     {
357         ALA = 1;
358         DelayUsNo(338/2);
359         ALA = 0;
360         DelayUsNo(338/2);
361     }
362     DelayMs(20);
363
364     for( i = 0; i< 112 ; i++ )

```

```

365 {
366     ALA = 1;
367     DelayUsNo(301/2);
368     ALA = 0;
369     DelayUsNo(301/2);
370 }
371 DelayMs(20);
372
373 for( i = 0; i< 600 ; i++ )
374 {
375     ALA = 1;
376     DelayUsNo(451/2);
377     ALA = 0;
378     DelayUsNo(451/2);
379 }
380 DelayMs(20);
381 //second sentence
382 }
383
384 /*-----
385 for the us delay function, incorporating an input
386 parameter unsigned char t and no return value,
387 where unsigned char represents the definition of
388 an unsigned character variable with a value range
389 of 0-255, given a 12M crystal oscillator,
390 for precise delay, assembly language is recommended.
391 -----*/
392 void DelayUsNo(unsigned int t)
393 {
394     while(--t);
395 }
396
397 void DelayUs2x(unsigned char t)
398 {
399     while(--t);
400 }
401
402 /*-----
403 same as us buuuut changed to ms
404 -----*/
405 void DelayMs(unsigned char t)
406 {
407     while(t--)
408     {
409         //roughly 1ms
410         DelayUs2x(245);
411         DelayUs2x(245);
412     }
413 }
414
415 /*-----
416 Digital display
417 -----*/
418 void DigDisplay()
419 {
420     switch(dis)
421     {
422         case(0):
423             LSA=0;LSB=0;LSC=0; break;//0th digit
424         case(1):
425             LSA=1;LSB=0;LSC=0; break;//1st
426         case(2):
427             LSA=0;LSB=1;LSC=0; break;//2nd
428         case(3):
429             LSA=1;LSB=1;LSC=0; break;//3rd
430         case(4):
431             LSA=0;LSB=0;LSC=1; break;//4th
432         case(5):
433             LSA=1;LSB=0;LSC=1; break;//5th
434         case(6):
435             LSA=0;LSB=1;LSC=1; break;//6th
436         case(7):
437             LSA=1;LSB=1;LSC=1; break;//7th
438     }
439     P0=TempData[dis]; //segmentcode output
440     //delay(100); //scan once a while
441     //P0=0x00; //Eliminate residual
442 }
443
444 /*-----
445 Timer Initialization
446 -----*/
447 void Init_Timer0(void)
448 {
449     TMOD = 0x11; //mode 1 16bit timer
450     //TH0=0x00; //nope, not this
451     //TL0=0x00;
452     EA=1; //main Interrupt on
453     ETO=1; //Timer Interrupt on
454     TRO=1; //Timer on
455     ET1=1;
456     TR1=1;
457 }

```

```

456 /*-----
457 Interrupt sub program
458 -----*/
459 void Timer0_isr(void) interrupt 1
460 {
461     static unsigned int num;
462     TH0=(65536-2000)/256; //give 2ms value
463     TL0=(65536-2000)%256;
464
465     num++;
466
467     if(num==500) //roughly 1s
468     {
469         num=0;
470         second++;
471         if(second==60) //yup
472         {
473             second=0;
474             minute++;
475             if(minute==60) //same
476             {
477                 minute=0;
478                 hour++;
479                 if(hour==24) //same
480                     hour=0;
481             }
482         }
483     }
484
485     DigDisplay();
486
487     countdown();
488 }
489
490 void Timer0_isr1(void) interrupt 3
491 {
492     TH1=(65536-2000)/256; //2ms is enough for the display
493     //to be stable
494     TL1=(65536-2000)%256;
495
496     dis++;
497     if(dis == 8)
498     {
499         dis = 0;
500     }
501 }
502
503 /*-----
504 Scan the key and return the key value
505 -----*/
506 unsigned char KeyScan(void)
507 {
508     unsigned char keyvalue;
509
510     if(KeyPort!=0xff)
511     {
512         DelayMs(10);
513         if(KeyPort!=0xff)
514         {
515             keyvalue=KeyPort;
516             while(KeyPort!=0xff)
517             {
518                 DigDisplay();
519             }
520             switch(keyvalue)
521             {
522                 case 0xfe: return 1; break;
523                 case 0xfd: return 2; break;
524                 case 0xfb: return 3; break;
525                 case 0xf7: return 4; break;
526                 case 0xef: return 5; break;
527                 case 0xdf: return 6; break;
528                 case 0xbf: return 7; break;
529                 case 0x7f: return 8; break;
530                 default: return 0; break;
531             }
532         }
533     }
534     return 0;
535 }
536
537 }

```

Code 1. Full Implementation