上海海事大学

工业控制器原理及应用（1） 课程设计

课程设计题目： 零食称重售卖机

小组成员

1. 电控荷 161 张理博 201610234033

2. 电控荷 161 邓宗豪 201610234028

学院： 物流工程学院

目录

1 课程设计题目与要求

课程设计题目：题 1. 零食称重售卖机

功能要求：

（1） 开机后能够显示及修改当前时间；

（2） 有两种零食 AB 可供选择，两种零食均可通过按键修改单价；

（3） 每种零食均可选择 1 包（满杯 600g）或半包（约 300g）：零食称重后，当称重传感器

（滑动变阻器模拟）检测到>550g：亮绿灯，数码管显示所选零食的种类、重量、当前单价

和总价，并计入每种零食的累计总价；

（4） 当称重传感器检测到 280g<重量<400g，亮黄灯，数码管显示所选零食的种类、重量、

当前单价和总价，并计入每种零食的累计总价；当重量小于 280g 或 400g< 重量<550g，亮

红灯，此时不显示和不计算总价，且不计入累计金额；

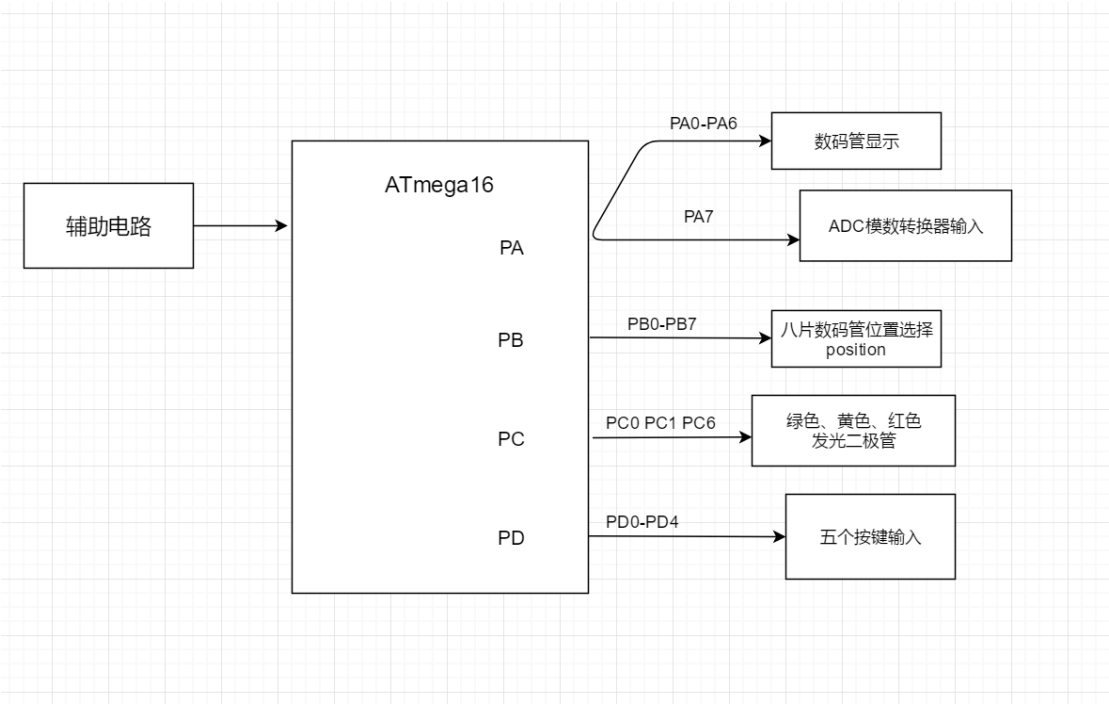
（5） 可通过按键选择显示开机后的每种零食的销售总金额，也可通过按键清零总金额，重新

开始累计，可通过按键切换/选择显示当前重量、当前时间。

硬件调试实现以上功能。

2 分析功能要求与确定方案

2.1 硬件系统框图



2.2 任务划分与时间进度表

任务	时间进度	任务负责人
设计总体功能结构	18 周周一	张理博
绘制软仿真线路图	18 周周一-周四	张理博
编写 C 语言程序	18 周周一-周四	张理博
进行软件仿真	18 周周一-周四	张理博
硬件调试	18 周周三-周四	张理博，邓宗豪
完成课程设计报告	18 周周四-周五	张理博，邓宗豪

2.3 确定选用芯片型号，进行相应计算

芯片型号： ATmega16

内部时钟频率： $f_{clk-CPU}=8MHz$

Timer2：

用于时间计数，t=1s,预分频系数选择 256 即 N=256

$$n=\frac{t\cdot f_{clk-CPU}}{N}=\frac{1\cdot 8\cdot 10^{6}}{256}=31250$$

$$\frac{n}{adopter}=\frac{31250}{250}=125$$

$$OCR2=\frac{n}{adopter}-1=125-1=124$$

Timer0:

用于触发 ADC 模数转换器，t=2ms,预分频系数选择 64 即 N=64

$$n=\frac{t\cdot f_{clk-CPU}}{N}=\frac{2\cdot 10^{-3}\cdot 8\cdot 10^{6}}{64}=250$$

$$OCR0=n-1=250-1=249$$

2.4 I/O 口分配

PA: PA0-PA6 七个管脚作为输出，用于数码管数字显示；

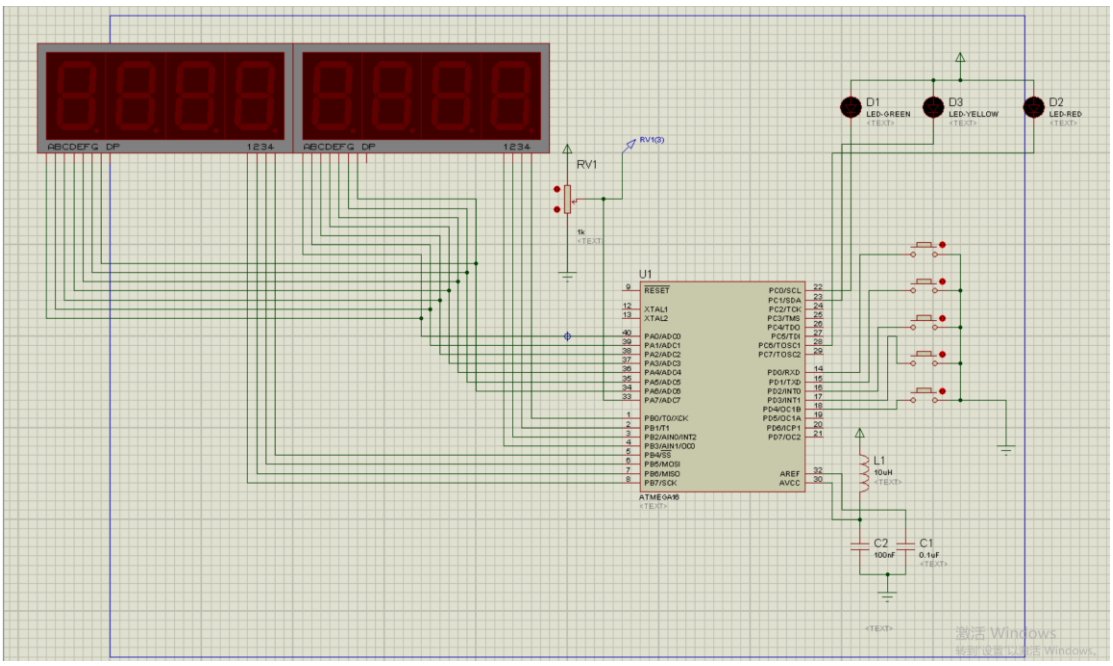
PA7 管脚作为输入，用于 ADC 模数转换器输入。

PB: PB0-PB7 八个管脚作为八片数码管位置选择 position。

PC: PC0、PC1、PC6 三个管脚作为输出分别控制绿色、黄色、红色发光二极管的亮与否，其余 PC 管脚不使用。

PD: PD0-PD4 五个管脚作为输入，用于按键输入，其余 PD 管脚不使用。

3 电路原理图设计



4 软件设计

4.1 程序块划分

（1）主程序；

（2）延时程序；

（3）按键扫描子程序；

（4）显示子程序：

4-1 数码管显示子程序；

4-2 三盏有色发光二极管显示子程序；

（5）中断服务程序：

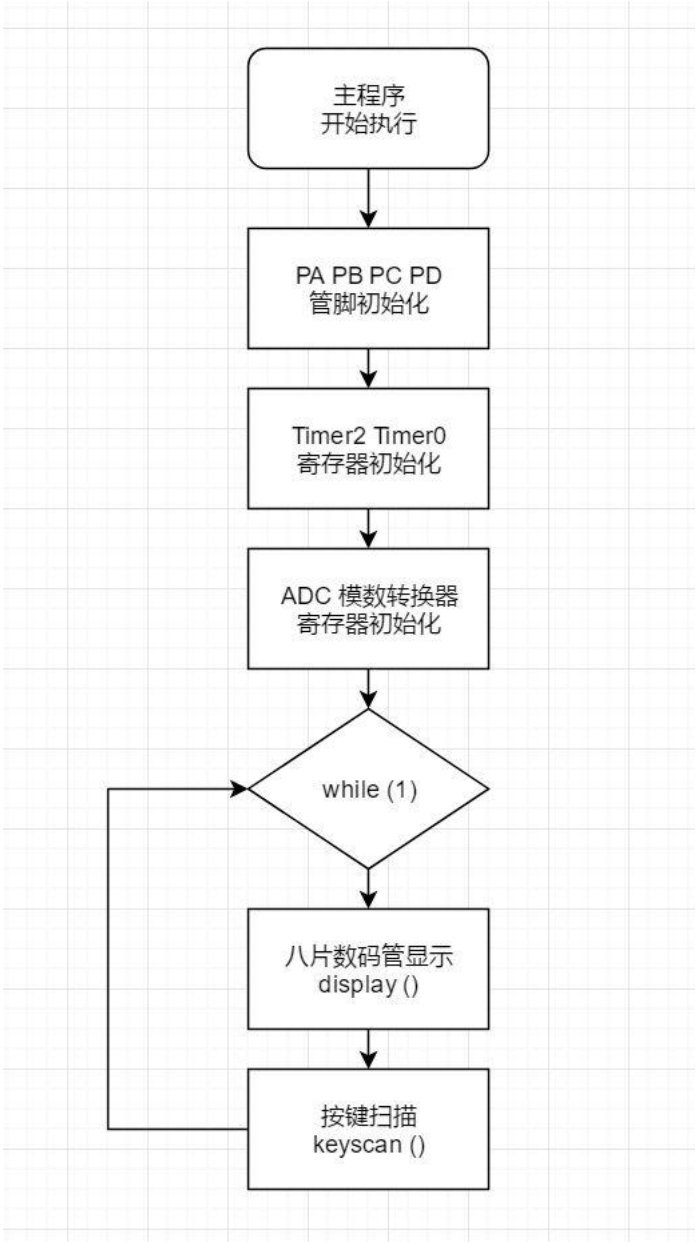
5-1 Timer0 中断服务程序；

5-2 Timer2 中断服务程序；

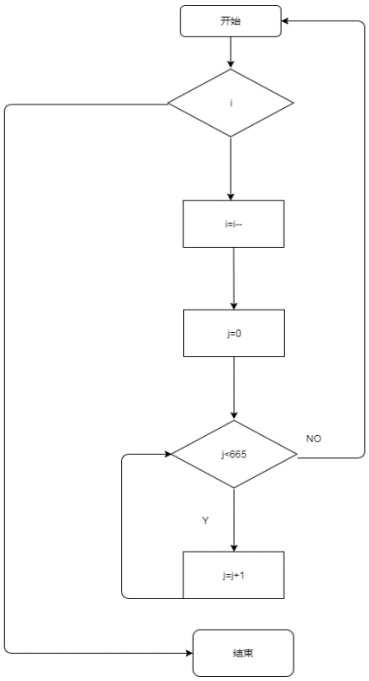
5-3 ADC 模数转换器中断服务程序。

4.2 绘制各程序的流程框图

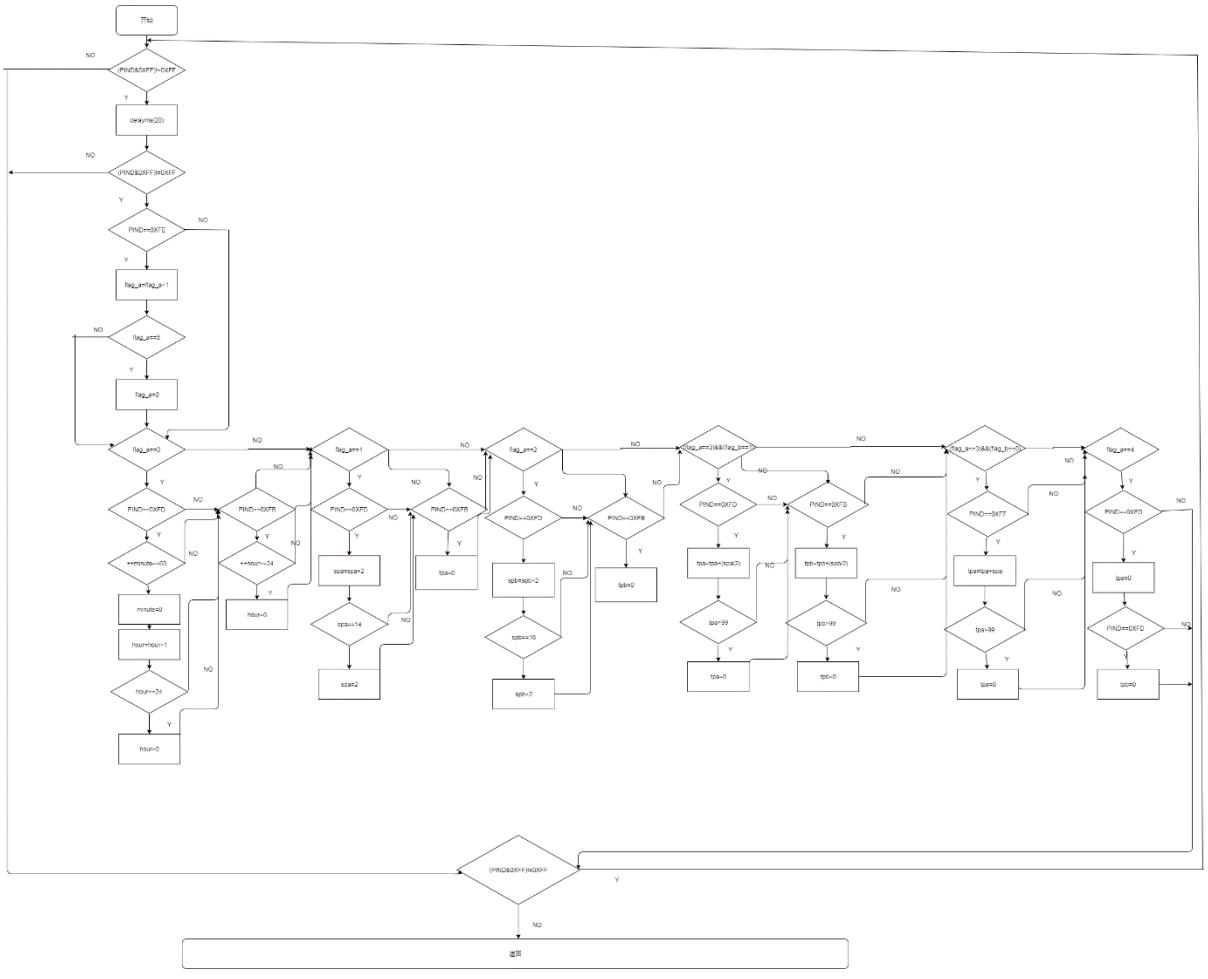
（1）主程序流程框图



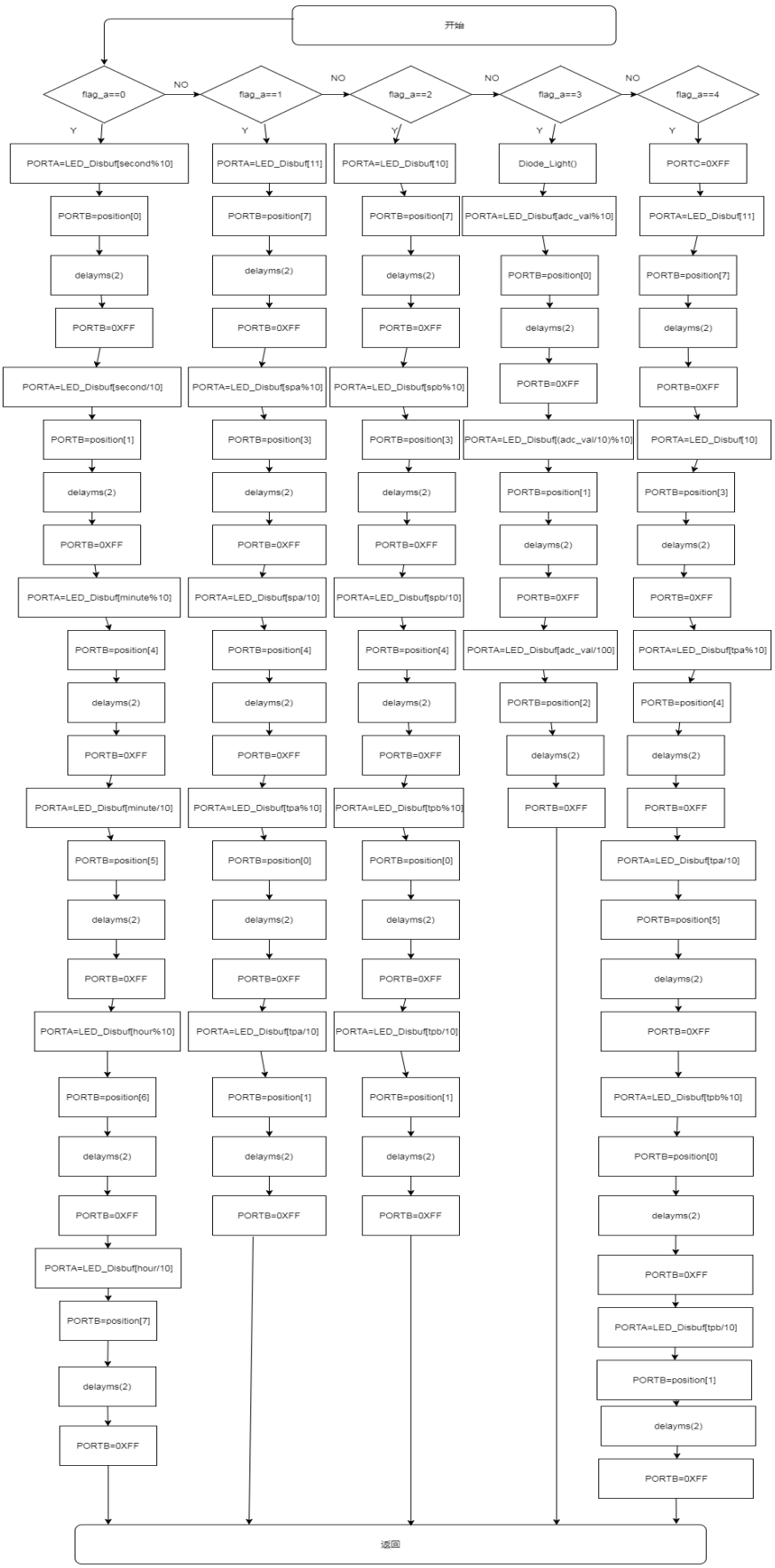
（2）延时程序流程框图



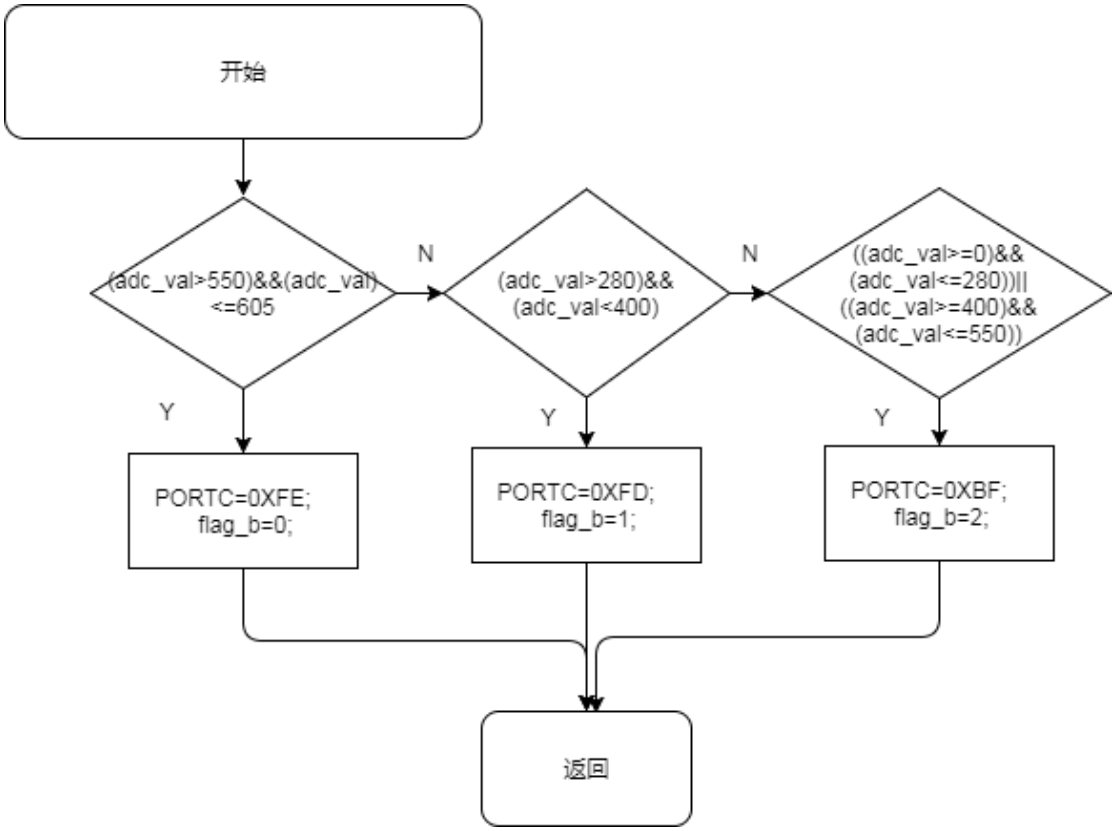
（3）按键扫描子程序流程框图



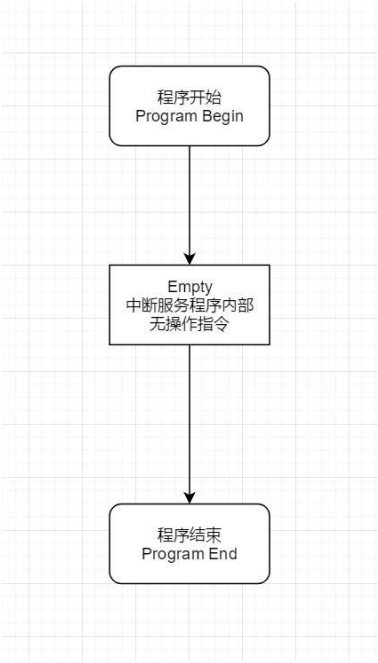
（4-1）数码管显示子程序流程框图



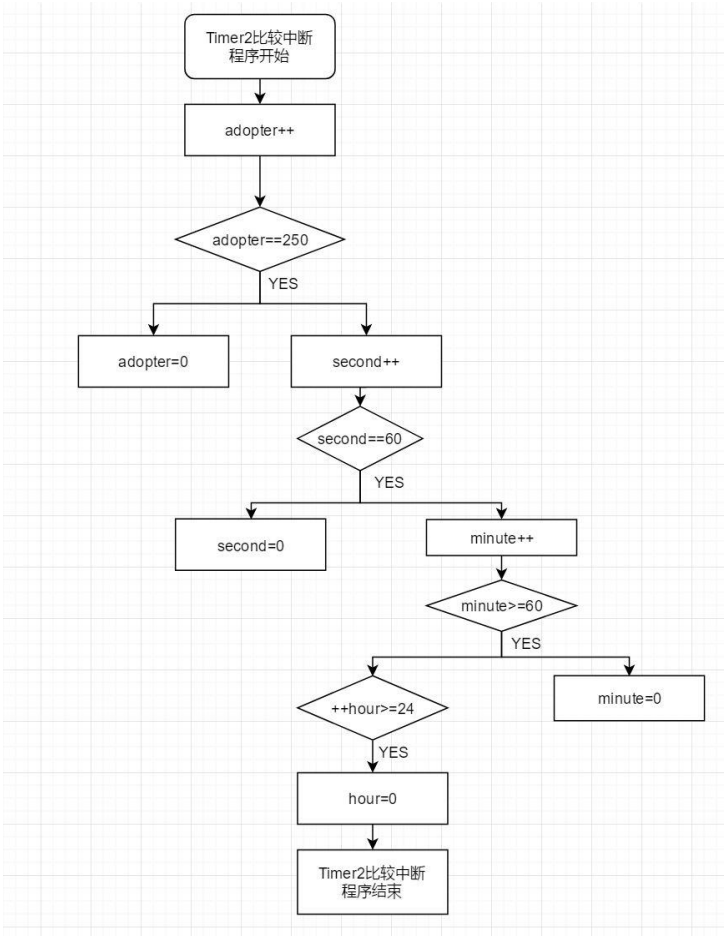
（4-2）有色发光二极管显示流程框图



（5-1）Timer0 中断服务程序流程框图



（5-2）Timer2 中断服务程序流程框图



（5-3）ADC 模数转换器中断服务程序流程框图

4.3 变量定义表

变量类型	变量名称	变量含义
unsigned char	LED_Disbuf[12]	数码管七段码显示
unsigned char	position[8]	八片数码管位置选择
unsigned int	second=0	秒
unsigned int	minute=55	分钟
unsigned int	hour=20	小时
unsigned long int	adc_rel	ADC 参考量
unsigned long int	adc_val	ADC 转换输出量
int	flag_a=0	数码管显示模式切换 flag
int	flag_b=0	屏蔽错误称量购买 flag
int	spa=2	A 零食单价
int	spb=4	B 零食单价
int	tpa=0	A 零食累计价格
int	tpb=0	B 零食累计价格
int	adopter=0	Timer2 调整 n 值参数

4.4 编写详细程序

(globals)

工控1课程设计ZLB.c

```
#include "iom16v.h"
// Common Cathode
unsigned char LED_Disbuf[12]={0X3F,0X06,0X5B,0X4F,0X66,0X6D,0X7D,0X07,0X7F,0X6F,0X7C,0X77};
// Selection Bit---Common Cathode
unsigned char position[8]={0XFE,0XFD,0XFB,0XF7,0XEF,0XDF,0XBF,0X7F};

unsigned int second=0;          // 秒
unsigned int minute=55;         // 分 初始值为55
unsigned int hour=20;           // 时 初始值为20


unsigned long int adc_rel;
unsigned long int adc_val;          // ADC模数转换器的输出显示值

int flag_a=0;                       // 用于五种显示模式切换的flag
int flag_b=0;                       // 用于当称重不满足要求时屏蔽累计总价格的flag
int spa=2;                          // A零食的单价 初始值为2
int spb=4;                          // B零食的单价 初始值为4
int tpa=0;                          // A零食的总价 初始值为0
int tpb=0;                          // B零食的总价 初始值为0
int adopter=0;

// 延时功能
void delayms(unsigned int i)
{
    unsigned int j;
    while (i--)
    {
        for (j=0;j<665;j++);
    }
}
```

工控1课程设计ZLB.c

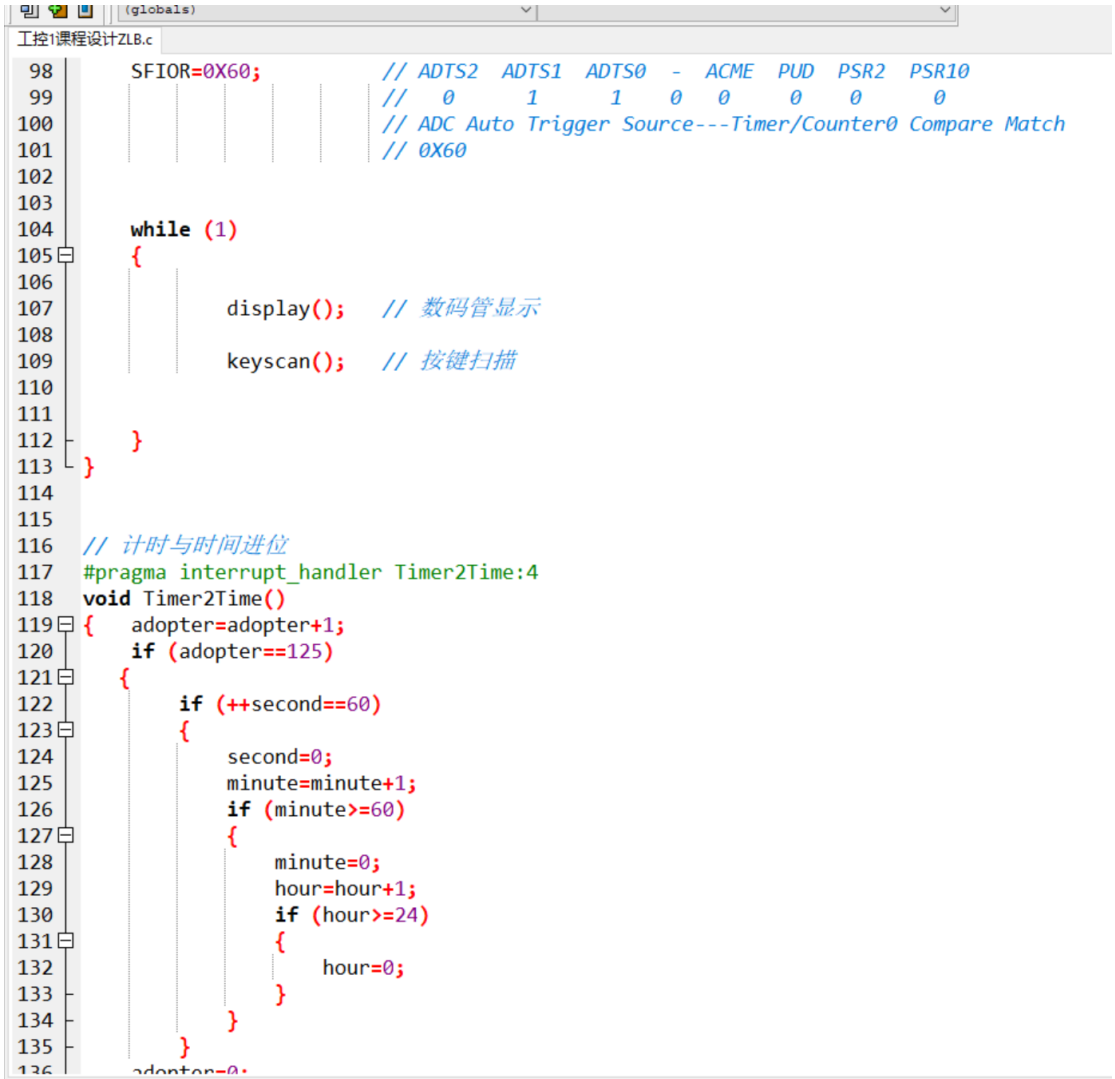
```
34
35
36  // 主程序
37  void main()
38  {
39      DDRA=0X7F;              // PA0-PA6作七段码数值显示  PA7作ADC模数转换器输入
40      PORTA=0X80;
41      DDRB=0XFF;              // 八片数码管position显示
42      PORTB=0X00;
43      DDRC=0XFF;              // PC0绿色发光二极管输出
44                              // PC1黄色发光二极管输出
45                              // PC6红色发光二极管输出
46      PORTC=0X7F;
47
48      DDRD=0XE0;              // PD0-PD4五个管脚作为按键输入
49      PORTD=0XFF;
50
51      ASSR=0X00;              // Timer2关闭异步计数 开启内部计数 时钟频率为8MHz
52                              //   -    -    -    -    AS2  TCN2UB  OCR2UB  TCR2UB
53                              //   0    0    0    0    0      0       0       0
54                              // 0X00
55
56      TCCR2=0X0E;             // FOC2  WGM20  COM21  COM20  WGM21  CS22  CS21  CS20
57                              //  0      0      0      0      1      1     1     0
58                              // 0X0F
59                              // 选用CTC模式   预分频系数选择256
60
61      OCR2=124;                // n=(f-clock-cpu)*(t)/(N)=(8MHz)*(1)/(256)=31250  31250/250=125=n
62                              // OCR2=n-1=125-1=124
63                              // Timer2用于时间计数 因此要作为秒脉冲发生器 t=1s
64                              // 由于软仿真时内部时钟频率为4MHz，硬件调试时内部时钟频率为8MHz
65                              // 因此软仿真时adopter=125 硬件调试时adopter=250
66
67
68      TCCR0=0X0B;             // Timer0 CTC模式作为模数转换器ADC的 Auto Trigger Source
69                              // FOC0  WGM00  COM01  COM00  WGM01  CS02  CS01  CS00
70                              //   0     0      0      0      1     0     1     1
71                              // 0X0B
72                              // 预分频系数选择64
73
74      OCR0=124;               // n=(f-clock-cpu)*(t)/(N)=(8MHz)*(2ms)/(64)=250
75                              // OCR0=n-1=250-1=249
76
77      TIMSK=0X82;             // 开启Timer2的CTC模式与Timer0的CTC模式
78                              // OCIE2  TOIE2  TICIE1  OCIE1A  OCIE1B  TOIE1  OCIE0  TOIE0
79                              //   1      0      0       0       0       0      1      0
80                              // 0X82
81
82      SREG=0X80;              // 总开关打开 I-bit置1
83
84      ADMUX=0X47;             //  REFS1  REFS0  ADLAR  MUX4  MUX3  MUX2  MUX1  MUX0
85                              //    0      1      0     0     0     1     1     1
86                              // 0X47
87                              // AVCC with ecternal capacitor at AREF pin
88                              // Right Adjusted
89                              // Single Ended Input---ADC7
90
91      ADCSRA=0XBD;            // ADEN  ADSC  ADATE  ADIF  ADIE  ADPS2  ADPS1  ADPS0
92                              //  1     0      1     1     1     1      0      1
93                              // 0XBD
94                              // Enable the ADC
95                              // Auto Triggering
96                              // ADC Prescaler Select---Division Factor---32
97
98      SFIOR=0X60;             // ADTS2  ADTS1  ADTS0  -  ACME  PUD  PSR2  PSR10
99                              //   0      1      1    0   0     0    0     0
100                             // ADC Auto Trigger Source---Timer/Counter0 Compare Match
101                             // 0X60
102
103
104     while (1)
105     {
```

Compiler | Resources | Compile Log | Debug | Find Results

工控1课程设计ZLB.c


```
67
68      TCCR0=0X0B;             // Timer0 CTC模式作为模数转换器ADC的 Auto Trigger Source
69                              // FOC0  WGM00  COM01  COM00  WGM01  CS02  CS01  CS00
70                              //   0     0      0      0      1     0     1     1
71                              // 0X0B
72                              // 预分频系数选择64
73
74      OCR0=124;               // n=(f-clock-cpu)*(t)/(N)=(8MHz)*(2ms)/(64)=250
75                              // OCR0=n-1=250-1=249
76
77      TIMSK=0X82;             // 开启Timer2的CTC模式与Timer0的CTC模式
78                              // OCIE2  TOIE2  TICIE1  OCIE1A  OCIE1B  TOIE1  OCIE0  TOIE0
79                              //   1      0      0       0       0       0      1      0
80                              // 0X82
81
82      SREG=0X80;              // 总开关打开 I-bit置1
83
84      ADMUX=0X47;             //  REFS1  REFS0  ADLAR  MUX4  MUX3  MUX2  MUX1  MUX0
85                              //    0      1      0     0     0     1     1     1
86                              // 0X47
87                              // AVCC with ecternal capacitor at AREF pin
88                              // Right Adjusted
89                              // Single Ended Input---ADC7
90
91      ADCSRA=0XBD;            // ADEN  ADSC  ADATE  ADIF  ADIE  ADPS2  ADPS1  ADPS0
92                              //  1     0      1     1     1     1      0      1
93                              // 0XBD
94                              // Enable the ADC
95                              // Auto Triggering
96                              // ADC Prescaler Select---Division Factor---32
97
98      SFIOR=0X60;             // ADTS2  ADTS1  ADTS0  -  ACME  PUD  PSR2  PSR10
99                              //   0      1      1    0   0     0    0     0
100                             // ADC Auto Trigger Source---Timer/Counter0 Compare Match
101                             // 0X60
102
103
104     while (1)
105     {
```


Compiler | Resources | Compile Log | Debug | Find Results




```
                if (hour>=24)
                {
                    hour=0;
                }
            }
        }
        adopter=0;
    }
}



// Timer0 CTC模式
#pragma interrupt_handler Timer0_CTC:20
void Timer0_CTC ()
{

}

```

工控1课程设计ZLB.c

```
146
147 }
148
149
150
151 // 绿色、黄色、红色发光二极管显示控制
152 void Diode_Light()
153 {
154     if ((adc_val>550)&&(adc_val)<=605)
155     {
156         PORTC=0XFE;      // 使绿色发光二极管亮
157         flag_b=0;
158     }
159     if ((adc_val>280)&&(adc_val<400))
160     {
161         PORTC=0XFD;      // 使黄色发光二极管亮
162         flag_b=1;
163
164     }
165     if (((adc_val>=0)&&(adc_val<=280))||((adc_val>=400)&&(adc_val<=550)))
166     {
167         PORTC=0XBF;      // 使红色发光二极管亮
168         flag_b=2;
169
170     }
171 }
172
173
174
175 // ADC模数转换器
176 #pragma interrupt_handler adc_isr:15
177 void adc_isr()
178 {
179       adc_rel=ADCL;
180       adc_rel|=(unsigned int)(ADCH<<8);
181       adc_val=(unsigned long)adc_rel*606/1024;    // 数字量显示范围控制在0-605
182
183 }
184
```

Compiler | Resources | Compile Log | Debug | Find Results

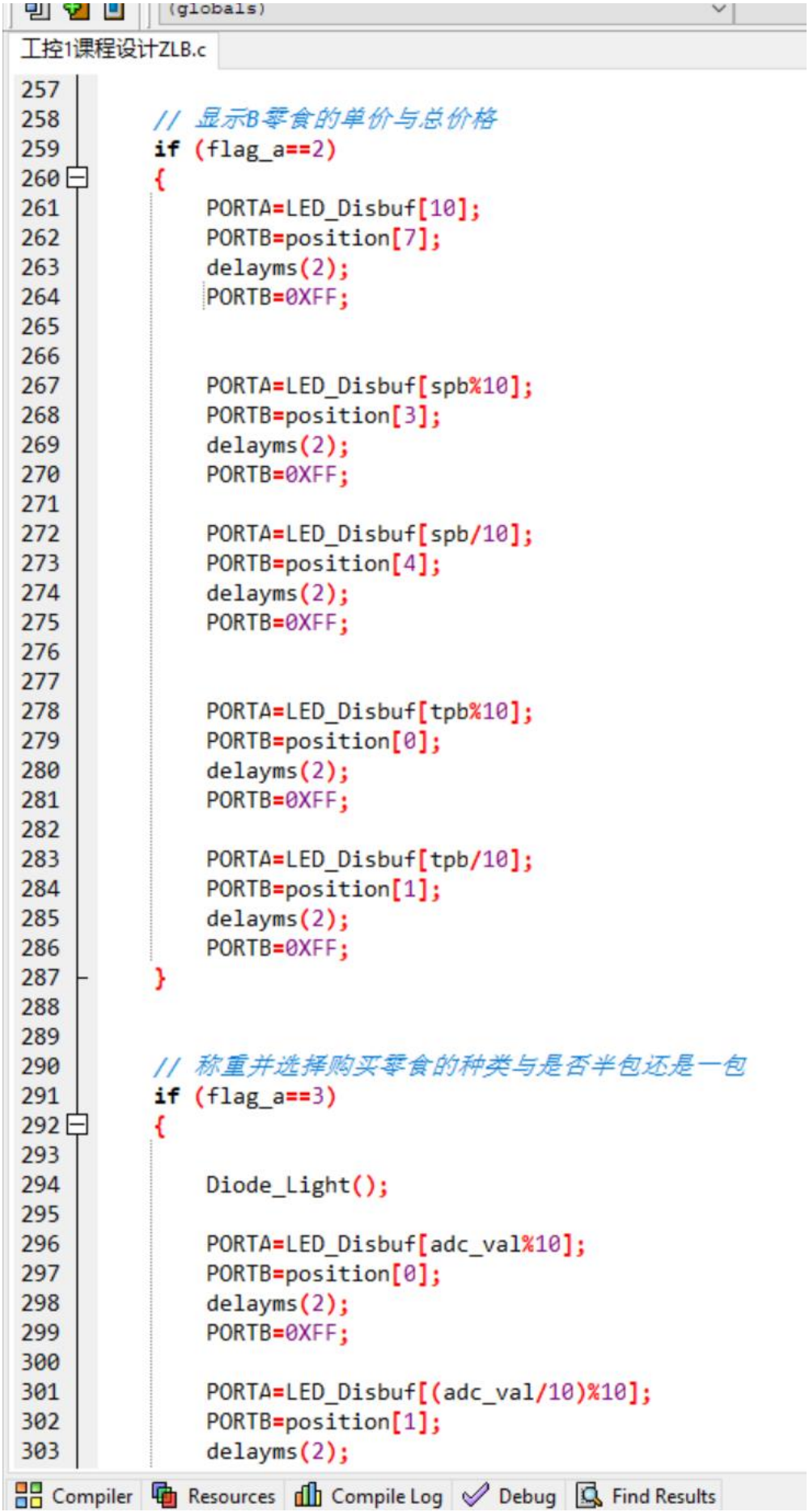
工控1课程设计ZLB.c

```
182
183 }
184
185
186
187
188 // 数码管显示子程序
189 void display()
190 {
191     // 显示时间
192     if (flag_a==0)
193     {
194         PORTA=LED_Disbuf[second%10];
195         PORTB=position[0];
196         delayms(2);
197         PORTB=0XFF;
198
199         PORTA=LED_Disbuf[second/10];
200         PORTB=position[1];
201         delayms(2);
202         PORTB=0XFF;
203
204         PORTA=LED_Disbuf[minute%10];
205         PORTB=position[4];
206         delayms(2);
207         PORTB=0XFF;
208
209         PORTA=LED_Disbuf[minute/10];
210         PORTB=position[5];
211         delayms(2);
212         PORTB=0XFF;
213
214
215         PORTA=LED_Disbuf[hour%10];
216         PORTB=position[6];
217         delayms(2);
218         PORTB=0XFF;
219
220         PORTA=LED_Disbuf[hour/10];
221         PORTB=position[7];
222         delayms(2);
223         PORTB=0XFF;
224     }
225
226     // 显示A零食的单价与总价格
227     if (flag_a==1)
228     {
```

工控1课程设计ZLB.c

```
222         delayms(2);
223         PORTB=0XFF;
224     }
225
226     // 显示A零食的单价与总价格
227     if (flag_a==1)
228     {
229         PORTA=LED_Disbuf[11];
230         PORTB=position[7];
231         delayms(2);
232         PORTB=0XFF;
233
234         PORTA=LED_Disbuf[spa%10];
235         PORTB=position[3];
236         delayms(2);
237         PORTB=0XFF;
238
239         PORTA=LED_Disbuf[spa/10];
240         PORTB=position[4];
241         delayms(2);
242         PORTB=0XFF;
243
244
245         PORTA=LED_Disbuf[tpa%10];
246         PORTB=position[0];
247         delayms(2);
248         PORTB=0XFF;
249
250         PORTA=LED_Disbuf[tpa/10];
251         PORTB=position[1];
252         delayms(2);
253         PORTB=0XFF;
254
255     }
256
257
258     // 显示B零食的单价与总价格
259     if (flag_a==2)
260     {
261         PORTA=LED_Disbuf[10];
262         PORTB=position[7];
263         delayms(2);
264         PORTB=0XFF;
265
266
267         PORTA=LED_Disbuf[spb%10];
268         PORTB=position[3];
```

Compiler | Resources | Compile Log | Debug | Find Results


(globals)
工控1课程设计ZLB.c
257
258 // 显示B零食的单价与总价格
259 if (flag_a==2)
260 {
261 PORTA=LED_Disbuf[10];
262 PORTB=position[7];
263 delayms(2);
264 PORTB=0XFF;
265
266
267 PORTA=LED_Disbuf[spb%10];
268 PORTB=position[3];
269 delayms(2);
270 PORTB=0XFF;
271
272 PORTA=LED_Disbuf[spb/10];
273 PORTB=position[4];
274 delayms(2);
275 PORTB=0XFF;
276
277
278 PORTA=LED_Disbuf[tpb%10];
279 PORTB=position[0];
280 delayms(2);
281 PORTB=0XFF;
282
283 PORTA=LED_Disbuf[tpb/10];
284 PORTB=position[1];
285 delayms(2);
286 PORTB=0XFF;
287 }
288
289
290 // 称重并选择购买零食的种类与是否半包还是一包
291 if (flag_a==3)
292 {
293
294 Diode_Light();
295
296 PORTA=LED_Disbuf[adc_val%10];
297 PORTB=position[0];
298 delayms(2);
299 PORTB=0XFF;
300
301 PORTA=LED_Disbuf[(adc_val/10)%10];
302 PORTB=position[1];
303 delayms(2);
Compiler
Resources
Compile Log
Debug
Find Results

工控1课程设计ZLB.c

```
300
301         PORTA=LED_Disbuf[(adc_val/10)%10];
302         PORTB=position[1];
303         delayms(2);
304         PORTB=0XFF;
305
306         PORTA=LED_Disbuf[adc_val/100];
307         PORTB=position[2];
308         delayms(2);
309         PORTB=0XFF;
310
311
312
313     }
314
315     // 同时显示A零食与B零食各自的总价格
316     if (flag_a==4)
317     {
318         PORTC=0XFF;       // 关闭发光二极管
319
320         PORTA=LED_Disbuf[11];
321         PORTB=position[7];
322         delayms(2);
323         PORTB=0XFF;
324
325         PORTA=LED_Disbuf[10];
326         PORTB=position[3];
327         delayms(2);
328         PORTB=0XFF;
329
330         PORTA=LED_Disbuf[tpa%10];
331         PORTB=position[4];
332         delayms(2);
333         PORTB=0XFF;
334
335         PORTA=LED_Disbuf[tpa/10];
336         PORTB=position[5];
337         delayms(2);
338         PORTB=0XFF;
339
340         PORTA=LED_Disbuf[tpb%10];
341         PORTB=position[0];
342         delayms(2);
343         PORTB=0XFF;
344
345         PORTA=LED_Disbuf[tpb/10];
346         PORTB=position[1];
```

Compiler | Resources | Compile Log | Debug | Find Results

工控1课程设计ZLB.c

```
342             delayms(2);
343             PORTB=0XFF;
344
345             PORTA=LED_Disbuf[tpb/10];
346             PORTB=position[1];
347             delayms(2);
348             PORTB=0XFF;
349
350
351         }
352
353
354
355
356     }
357
358
359     // 按键扫描子程序
360     void keyscan()
361     {
362         if ((PIND&0XFF)!=0XFF)
363         {
364             delayms(20);                    // 按键防抖消振
365             if ((PIND&0XFF)!=0XFF)
366             {
367
368
369
370                 if (PIND==0XFE)
371                 {
372                     flag_a=flag_a+1;
373                     if   (flag_a==5)
374                        {
375                          flag_a=0;
376                        }
377                 }
378
379
380                 if (flag_a==0)
381             {
382
383                 if (PIND==0XFD)         // 修改分钟
384                 {
385                     if (++minute==60)
386                     {
387                         minute=0;
388                         hour=hour+1;
```

Compiler | Resources | Compile Log | Debug | Find Results

工控1课程设计ZLB.c

```
381         {
382
383             if (PIND==0XFD)         // 修改分钟
384             {
385                 if (++minute==60)
386                 {
387                     minute=0;
388                     hour=hour+1;
389                     if(hour==24) hour=0;
390                 }
391             }
392             if (PIND==0XFB)         // 修改时钟
393             {
394                 if (++hour>=24) hour=0;
395             }
396         }
397
398             if (flag_a==1)
399             {
400                 if (PIND==0XFD)  // 修改A零食单价
401                 {
402                     spa=spa+2;
403                     if (spa==14) spa=2;
404                 }
405                 if (PIND==0XFB)  // A零食总价格清零
406                 {
407                     tpa=0;
408                 }
409             }
410
411             if (flag_a==2)
412             {
413                 if (PIND==0XFD)  // 修改B零食单价
414                 {
415                     spb=spb+2;
416                     if (spb==16) spb=2;
417                 }
418                 if (PIND==0XFB)  // B零食总价格清零
419                 {
420                     tpb=0;
421                 }
422             }
423
424             if ((flag_a==3)&&(flag_b==1))
425             {
426
427
```

Compiler | Resources | Compile Log | Debug | Find Results

工控1课程设计ZLB.c

```
420                 tpb=0;
421             }
422         }
423
424         if ((flag_a==3)&&(flag_b==1))
425         {
426
427
428             if (PIND==0XFD)  // 购买半包A零食
429             {
430                 tpa=tpa+(spa/2);
431                 if (tpa>99) tpa=0;
432             }
433
434
435             if (PIND==0XFB)  // 购买半包B零食
436             {
437                 tpb=tpb+(spb/2);
438                 if (tpb>99) tpb=0;
439             }
440
441         }
442
443             if ((flag_a==3)&&(flag_b==0))
444         {
445
446             if (PIND==0XF7)  // 购买一包A零食
447                 {
448                     tpa=tpa+spa;
449                     if (tpa>99) tpa=0;
450                 }
451             if (PIND==0XEF)  // 购买一包B零食
452                 {
453                     tpb=tpb+spb;
454                     if (tpb>99) tpb=0;
455                 }
456
457         }
458
459
460         if (flag_a==4)
461         {
462             if (PIND==0XFD)  // A零食总价格清零
463             {
464                 tpa=0;
465             }
466             if (PIND==0XFB)  // B零食总价格清零
```

Compiler | Resources | Compile Log | Debug | Find Results

(globals)

工控1课程设计ZLB.c

```
439                 }
440
441             }
442
443             if ((flag_a==3)&&(flag_b==0))
444             {
445
446                 if (PIND==0XF7)  // 购买一包A零食
447                 {
448                     tpa=tpa+spa;
449                     if (tpa>99) tpa=0;
450                 }
451                 if (PIND==0XEF)  // 购买一包B零食
452                 {
453                     tpb=tpb+spb;
454                     if (tpb>99) tpb=0;
455                 }
456
457             }
458
459
460             if (flag_a==4)
461             {
462                 if (PIND==0XFD)  // A零食总价格清零
463                 {
464                     tpa=0;
465                 }
466                 if (PIND==0XFB)  // B零食总价格清零
467                 {
468                     tpb=0;
469                 }
470             }
471
472         }
473     }
474     while ((PIND&0XFF)!=0XFF);
475 }
476
477
478
479
480
481
```

5 软件仿真调试及修改

当我们小组在进行软件仿真时，遇到并解决了以下的问题。

（1）在时间显示里面，我们发现计时虽然可以自动的从 23：59 调节到 00:00，但是我们在手动修改分钟和时钟时，分钟可以由 59 走到 00，但是时钟会继续增加 23,24,25。。。后来我们在按键扫描子程序里面加了判断语句，当手动调节小时时间增加到 24 时，它会自动清零，以此问题得到了解决。

（2）在软仿真的数码管显示过程中，我们发现当开启软仿真时，数码管显示的并不是我们最初设定的时间，而是称重模式中的重量。按下数码管显示模式切换按键不能够实现显示模式的切换，经过仔细检查代码发现我们将重量即 ADC 输出值的显示放在了 ADC 触发源 Timer0 的中断服务程序里面，既没有添加 flag 使其屏蔽，也没有另 TIMSK 关断。我们将重量的显示放到主程序的 display 里面并且添加 flag_a 使其选择性的屏蔽，从而使该问题得到了解决。

6 硬件调试及修改

当我们小组顺利完成软仿真之后在进行硬件调试时，遇到并解决了以下主要的问题。

（1） 我们在模式切换时，原始的设定为首先显示时间，之后显示 A 零食可修改单价与总价，然后显示 B 零食，然后显示称重重量与购买选择。但是当我们将显示模式切换至称重模式时，数码管瞬间亮了一下然后熄灭，而且所有按键失灵，硬件调试直接卡死。由于我们的软仿真没有任何问题，因此很难发现问题到底出现在什么地方。经过仔细检查后发现，或许是我们将五个按键与三个有色发光二极管同时接在了 PD 八个管脚上的原因。我们认为这种同时令 PD0-PD4 作为按键输入与 PD5-PD7 作为有色发光二极管输出的管脚设置方式会导致硬件内部错乱，从而导致调试卡死，但是这在原理上又是合理的，因为我们已经通过了软仿真，具体原因我们无法理解。对于该问题，我们将三个有色发光二极管接在了 PC 的三个管脚上，PC0、PC1、PC6 分别接绿色、黄色与红色发光二极管，PD5-PD7 悬空不使用，从而使该问题得到了解决。

（2） 在硬件调试时，我们发现我们的时间显示走得很慢，分钟并不能每一分钟实现加一。询问了其他小组的同学后，我们了解到实验室硬件的异步晶振老化，导致计时可能出现延缓甚至停滞的现象。因此，我们决定对负责时间计时与进位的 Timer2 进

行调试修改。此前，我们使用的是异步晶振 32.768kHz，现在，我们令 ASSR=0X00,关闭异步计数，改为内部 8MHz 计数，并且重新计算了 OCR2 的参数，从而使该问题得到了解决。

（3） 在老师对我们进行硬件考核时，老师指出我们遗漏了一项功能，即当称重为半包时不能选择一包，当称重为一包时不能选择半包。对于该问题，我对称重数值的 flag 确立与按键扫描子程序里面的 flag 判断条件进行了简单的修改，从而使该问题得到了解决。

7 操作说明/使用说明

零食称重售卖机

本产品拥有五种数码管显示模式：

1. 时间显示与修改模式；

2. A 零食单价可修改与总价可清零模式；

3. B 零食单价可修改与总价可清零模式；

4. 称重与选择购买模式；

5. A 零食与 B 零食总价同时显示并均可清零模式。

本产品配备五个按键与三个有色发光二极管。其中，三个有色发光二极管只有在模式四即称重与选择购买模式下才会发光，其余模式下皆关闭。

机器打开后，数码管自动显示初始设定时间 20 时 55 分 00 秒。首先声明，按键 1 负责模式切换，每按下一次按键 1 数码管会切换不同的显示模式，并且会 1-2-3-4-5-1-2-3-4-5-1。。。不断循环。在时间显示模式中，按下按键 2 会使分钟加一，按下按键 3 会使小时加一，按键 4 与按键 5 无效。

若按下按键 1 使数码管显示模式 2，此时会显示 A 零食的种类、单价与总价格。为 A 零食设置的初始单价为 2 元，按下按键 2 后可以修改 A 零食的单价，每次按下会使 A 零食单价加 2，A 零食单价最大值为 12，达到最大值后再次按下按键 2 会使 A 零食的单价置回 2；为 A 零食设置的初始累计价格为 0 元，按下按键 3 可以使 A 零食的累积价格清零，按键 4 与按键 5 无效。

若按下按键 1 使数码管显示模式 3，此时会显示 B 零食的种类、单价与总价格。为 B 零食设置的初始单价为 4 元，按下按键 2 后可以修改 B 零食的单价，每次按下会使 B 零食单价加 2，B 零食单价最大值为 14 元，达到最大值后再次按下按键 2 会使 B 零食单价置回 2；为 B 零食设置的初始累计价格即总价格为 0 元，按下按键 3 可以使 B 零食累计价格清零，按键 4 与按键 5 无效。

若按下按键 1 使数码管显示模式 4，此时数码管会显示滑动变阻器调节模拟量通过 ADC 转换所得到的相应数字量，即零食的重量。本产品设置的零食的称重范围为 0 到 605。在该模式下，有色发光二极管也会根据重量的变化进行选择性的打开。对于某一个重量值，只会有一个有色二极管发光，其详细的范围是：

当重量值>550g，绿灯亮；

当 280g<重量值<400g，黄灯亮；

当重量值小于 280g 或 400g<重量值<550g，红灯亮。

在该模式下，消费者可以完成对 A、B 零食种类和一包半包的选择与购买。而该产品的特点是先称重，称重合适后再通过按键进行购买，具体的购买操作是：

按下按键 2，购买一份半包 A 零食；

按下按键 3，购买一份半包 B 零食；

按下按键 4，购买一份一包 A 零食；

按下按键 5，购买一份一包 B 零食。

由于该产品的特点是先称重再购买，因此称重与购买的选择紧密相关，体现在：

（1）当有色二极管红灯亮时，无论按下上述四个购买按键的哪一个，A 与 B 零食的累计价格都不会发生变化。

（2）当有色二极管黄灯亮时，此时只允许购买半包 A 零食或者半包 B 零食。按下按键 4 与按键 5 的操作无效，不会使 A 零食或 B 零食的总价格变化。

（3）当有色二极管绿灯亮时，此时只允许购买一包 A 零食或一包 B 零食。按下按键 2 与按键 3 的操作无效，不会使 A 零食或 B 零食的总价格变化。

此外，该产品仍有一个特点，即可以在该模式下重复购买，例如在绿灯亮时按下按键 4 两次，即相当于购买了两份一包 A 零食，即两包 A

零食。

若按下按键 1 使数码管显示模式 5，此时会显示结束购买后的 A 零食与 B 零食各自的累计价格。其中，按下按键 2 会使 A 零食累计价格清零，按下按键 3 会使 B 零食累计价格清零，按键 4 与按键 5 无效。

至此结束零食称重售卖机的售卖过程，按下按键 1 可以返回模式 1 即时间显示模式，并可以开始新一轮的售卖过程。如果此前不对 A 零食与 B 零食的累计价格清零，那么在新的循环里面它们各自的累计价格将会是上次购买完成后各自的累计价格。

8 课程设计总结

工业控制器原理及应用（1）课程设计是我们上大学以来第一次所接受的课程设计课程。对于这样比较特殊的课程，首先我们对它是比较陌生的，因此课程设计开始的第一天我们毫无头绪，不知道该如何设计、如何画仿真图以及如何编程。因此第一天我们仅仅完成了时间的显示与修改，对于题目复杂功能要求的实现十分陌生。但是，随着时间的推移，我们逐渐对题目变得熟悉与了解，并且能够一步步将题目的要求条理清晰、化繁为简，并最终解决所有问题。这个过程中，诚然会遇到很多或大或小的麻烦。也经常让我们十分焦躁和苦恼。但最终发现这些或大或小的挫折也培养了我们的设计能力与动手能力，并且提高了我们对新事物的理解与学习能力。

而且，完成工控 1 的课程设计当然离不开 C 语言，我们经过该门课的课程设计，编程能力也有了一定的提升，这对今后我们在研究生学习与研究或者在工作实践中有十分重要的帮助。

完成工控 1 课程设计，我们学习与收获了很多，也对自己的产品较为满意与自豪。