

May 03, 13 19:56

jdefo002_lab7_part1.c

Page 1/3

```

1  /*      jdefo002_lab7_part1.c - 4/29/13
2  *      Name: Joshua DeForest-Williams Email: Jdefo002@ucr.edu
3  *      CS Login: jdefo002
4  *      Partner(s) Name: Ariana DeJaco E-mail: Adeja001@ucr.edu
5  *      Lab Section: 022
6  *      Assignment: Lab #7 Exercise #1
7  *      Exercise Description: Connect LEDs to PB0, PB1, PB2, and PB3. Light
8  *      the three LEDs PB0, PB1, and PB2 in sequence for 1 second each.
9  *      Meanwhile, blink PB3 on 1 second and off 1 second.
10 *      I acknowledge all content contained herein, excluding template or exampl
11 e
12 *      code, is my own original work.
13 */
14
15
16 #include <avr/io.h>
17 #include <avr/interrupt.h>
18 volatile unsigned char TimerFlag=0; // TimerISR() sets this to 1. C programmer s
19   hould clear to 0.
20 // Internal variables for mapping AVR's ISR to our cleaner TimerISR model.
21 unsigned long _avr_timer_M=1; // Start count from here, down to 0. Default 1 ms.
22 unsigned long _avr_timer_cntcurr=0; // Current internal count of lms ticks
23 void TimerOn() {
24     // AVR timer/counter controller register TCCR0
25     TCCR0 = 0x0B; // bit3bit6=10: CTC mode (clear timer on compare)
26     // bit2bit1bit0=011: pre-scaler /64
27     // 00001011: 0x0B
28     // SO, 8 MHz clock or 8,000,000 /64 = 125,000 ticks/s
29     // Thus, TCNT0 register will count at 125,000 ticks/s
30     // AVR output compare register OCR0.
31     OCR0 = 125; // Timer interrupt will be generated when TCNT0==OCR0
32     // We want a 1 ms tick. 0.001 s * 125,000 ticks/s = 125
33     // So when TCNT0 register equals 125,
34     // 1 ms has passed. Thus, we compare to 125.
35     // AVR timer interrupt mask register
36     TIMSK = 0x02; // bit1: OCIE0 -- enables compare match interrupt
37     //Initialize avr counter
38     TCNT0=0;
39     _avr_timer_cntcurr = _avr_timer_M;
40     // TimerISR will be called every _avr_timer_cntcurr milliseconds
41     //Enable global interrupts
42     SREG |= 0x80; // 0x80: 10000000
43 }
44 void TimerOff() {
45     TCCR0 = 0x00; // bit2bit1bit0=000: timer off
46 }
47 void TimerISR(){
48     TimerFlag = 1;
49 }
50 // In our approach, the C programmer does not touch this ISR, but rather TimerIS
51 R()
52 ISR(TIMER0_COMP_vect)
53 {
54     // CPU automatically calls when TCNT0 == OCR0 (every 1 ms per TimerOn setting
55 s)
56     _avr_timer_cntcurr--; // Count down to 0 rather than up to TOP
57     if (_avr_timer_cntcurr == 0) { // results in a more efficient compare
58 TimerISR(); // Call the ISR that the user uses
59     _avr_timer_cntcurr = _avr_timer_M;
60 }
61 }
62 // Set TimerISR() to tick every M ms
63 void TimerSet(unsigned long M) {
64     _avr_timer_M = M;
65     _avr_timer_cntcurr = _avr_timer_M;
66 }
67 volatile unsigned char B = 0x01;
68 enum Counter_States { InitState, SecState, ThirdState } CounterState;
69 unsigned char TckFct_Counter()
70 {
71     // Variable we are returning

```

May 03, 13 19:56

jdefo002_lab7_part1.c

Page 2/3

```

70
71
72     switch(CounterState)
73     {
74         //Transitions
75         case InitState: // Initial Transition
76             B=0x01;
77             CounterState = SecState;
78         break;
79         case SecState: // Initial Transition
80             B=0x02;
81             CounterState = ThirdState;
82         break;
83         case ThirdState: // Initial Transition
84             B=0x04;
85             CounterState = InitState;
86         break;
87         default:
88             break;
89     }
90
91     switch(CounterState)
92     { // Actions
93         case InitState:
94             break;
95         default:
96             break;
97     }
98 }
99 enum Blink_State { InitState, Blink_two } BlinkState;
100 unsigned char TckFct_Blink()
101 {
102     // Variable we are returning
103
104
105     switch(BlinkState)
106     {
107         //Transitions
108         case InitState: // Initial Transition
109             B = B | 0x08;
110             BlinkState = Blink_two;
111             break;
112         case Blink_two:
113             B = B & 0x07;
114             BlinkState = InitState;
115             break;
116         default:
117             break;
118     }
119
120     switch(CounterState)
121     { // Actions
122         case InitState:
123             break;
124         default:
125             break;
126     }
127 }
128 void main()
129 {
130     DDRB = 0xFF; // Set port B to output
131     PORTB = 0x00; // Init port B to 0s
132     TimerSet(1000);
133     TimerOn();
134     CounterState = InitState;
135     BlinkState = InitState;
136     while(1) {
137         // User code
138         TckFct_Counter();
139         TckFct_Blink();
140         PORTB = B;
141         while (!TimerFlag); // Wait 1 sec
142         TimerFlag = 0;

```

May 03, 13 19:56

jdefo002_lab7_part1.c

Page 3/3

```
143      // Note: For the above a better style would use a synchSM with TickSM()  
144      // This example just illustrates the use of the ISR and flag  
145      }  
146  }
```

May 03, 13 20:44

jdefo002_lab7_part2.c

Page 1/3

```

1  /*      jdefo002_lab7_part2.c - 4/29/13
2  *      Name: Joshua DeForest-Williams Email: Jdefo002@ucr.edu
3  *      CS Login: jdefo002
4  *      Partner(s) Name: Ariana DeJaco E-mail: Adeja001@ucr.edu
5  *      Lab Section: 022
6  *      Assignment: Lab #7 Exercise #2
7  *      Exercise Description: Connect LEDs to PB0, PB1, PB2, and PB3. Light
8  *      the three LEDs PB0, PB1, and PB2 in sequence for 1 second each.
9  *      Meanwhile, blink PB3 on 1 second and off 1 second.
10 *      EXTENSION: Modify the above example so the three LEDs light for
11 *      300 ms, while PB3's LED still blinks 1 second on and 1 second off.
12 *      I acknowledge all content contained herein, excluding template or exampl
13 e
14 *      code, is my own original work.
15 */
16
17 #include <avr/io.h>
18 #include <avr/interrupt.h>
19 volatile unsigned char TimerFlag=0; // TimerISR() sets this to 1. C programmer s
20 hould clear to 0.
21 // Internal variables for mapping AVR's ISR to our cleaner TimerISR model.
22 unsigned long _avr_timer_M=1; // Start count from here, down to 0. Default 1 ms.
23 unsigned long _avr_timer_cntcurr=0; // Current internal count of 1ms ticks
24 volatile unsigned char B = 0x01;
25 volatile unsigned char Blink = 0x08;
26 volatile unsigned char Count_Time = 3;
27 volatile unsigned char Blink_Time = 10;
28 void TimerOn() {
29     // AVR timer/counter controller register TCCR0
30     TCCR0 = 0x0B; // bit3bit6=10: CTC mode (clear timer on compare)
31     // bit2bit1bit0=011: pre-scaler /64
32     // 00001011: 0x0B
33     // SO, 8 MHz clock or 8,000,000 /64 = 125,000 ticks/s
34     // Thus, TCNT0 register will count at 125,000 ticks/s
35     // AVR output compare register OCR0.
36     OCR0 = 125; // Timer interrupt will be generated when TCNT0==OCR0
37     // We want a 1 ms tick. 0.001 s * 125,000 ticks/s = 125
38     // So when TCNT0 register equals 125,
39     // 1 ms has passed. Thus, we compare to 125.
40     // AVR timer interrupt mask register
41     TIMSK = 0x02; // bit1: OCIE0 -- enables compare match interrupt
42     //Initialize avr counter
43     TCNT0=0;
44     _avr_timer_cntcurr = _avr_timer_M;
45     // TimerISR will be called every _avr_timer_cntcurr milliseconds
46     //Enable global interrupts
47     SREG |= 0x80; // 0x80: 10000000
48 }
49 void TimerOff() {
50     TCCR0 = 0x00; // bit2bit1bit0=000: timer off
51 }
52 void TimerISR(){
53     TimerFlag = 1;
54     Count_Time++;
55     Blink_Time++;
56 }
57 // In our approach, the C programmer does not touch this ISR, but rather TimerIS
58 R()
59 ISR(TIMER0_COMP_vect)
60 {
61     // CPU automatically calls when TCNT0 == OCR0 (every 1 ms per TimerOn setting
62 s)
63     _avr_timer_cntcurr--; // Count down to 0 rather than up to TOP
64     if (_avr_timer_cntcurr == 0) { // results in a more efficient compare
65         TimerISR(); // Call the ISR that the user uses
66         _avr_timer_cntcurr = _avr_timer_M;
67     }
68 }
69 // Set TimerISR() to tick every M ms
70 void TimerSet(unsigned long M) {
71     _avr_timer_M = M;

```

May 03, 13 20:44

jdefo002_lab7_part2.c

Page 2/3

```

70     _avr_timer_cntcurr = _avr_timer_M;
71 }
72
73 enum Counter_States { InitState, SecState, ThirdState } CounterState;
74
75 unsigned char TckFct_Counter()
76 {
77     // Variable we are returning
78
79     switch(CounterState)
80     {
81         //Transitions
82         case InitState: // Initial Transition
83             B=0x01;
84             CounterState = SecState;
85             break;
86         case SecState: // Initial Transition
87             B=0x02;
88             CounterState = ThirdState;
89             break;
90         case ThirdState: // Initial Transition
91             B=0x04;
92             CounterState = InitState;
93             break;
94         default:
95             break;
96     }
97
98     switch(CounterState)
99     { // Actions
100         case InitState:
101             break;
102         default:
103             break;
104     }
105 }
106 enum Blink_State { InitialState, Blink_two } BlinkState;
107
108 unsigned char TckFct_Blink()
109 {
110     // Variable we are returning
111
112     switch(BlinkState)
113     {
114         //Transitions
115         case InitialState: // Initial Transition
116             Blink =0x08;
117             BlinkState = Blink_two;
118             break;
119         case Blink_two:
120             Blink = 0x00;
121             BlinkState = InitialState;
122             break;
123         default:
124             break;
125     }
126
127     switch(BlinkState)
128     { // Actions
129         case InitState:
130             break;
131         default:
132             break;
133     }
134 }
135
136 void main()
137 {
138     DDRB = 0xFF; // Set port B to output
139     PORTB = 0x00; // Init port B to 0s
140     TimerSet(100);
141     TimerOn();
142     CounterState = InitState;

```

May 03, 13 20:44

jdefo002_lab7_part2.c

Page 3/3

```
143 BlinkState = InitialState;
144 while(1) {
145     // User code
146     if (Count_Time == 3){
147         TckFct_Counter();
148         Count_Time = 0;
149     }
150     if (Blink_Time == 10){
151         TckFct_Blink();
152         Blink_Time = 0;
153     }
154     PORTB = B|Blink;
155     while (!TimerFlag);    // Wait 1 sec
156     TimerFlag = 0;
157     // Note: For the above a better style would use a synchSM with TickSM()
158     // This example just illustrates the use of the ISR and flag
159 }
160 }
```

May 03, 13 20:46

jdefo002_lab7_part3.c

Page 1/3

```

1  /* jdefo002_lab7_part3.c - 4/29/13
2  * Name: Joshua DeForest-Williams Email: Jdefo002@ucr.edu
3  * CS Login: jdefo002
4  * Partner(s) Name: Ariana DeJaco E-mail: Adeja001@ucr.edu
5  * Lab Section:022
6  * Assignment: Lab #7 Exercise #3
7  * Exercise Description:Connect LEDs to PB0, PB1, PB2, and PB3. Light
8  * the three LEDs PB0, PB1, and PB2 in sequence for 1 second each.
9  * Meanwhile, blink PB3 on 1 second and off 1 second.
10 * EXTENSION: Modify the above example so the three LEDs light for
11 * 300 ms, while PB3's LED still blinks 1 second on and 1 second off.
12 * EXTENSION 2: To the previous exercise's implementation, connect your
13 * speaker's red wire to PB4 and black wire to ground. Add a third task
14 * that toggles PB4 on for 2 ms and off for 2 ms as long as a switch on
15 * PA2 is in the on position.
16 * I acknowledge all content contained herein, excluding template or exampl
17 e
18 */
19
20 #include <avr/io.h>
21 #include <avr/interrupt.h>
22 volatile unsigned char TimerFlag=0; // TimerISR() sets this to 1. C programmer s
23 hould clear to 0.
24 // Internal variables for mapping AVR's ISR to our cleaner TimerISR model.
25 unsigned long _avr_timer_M=1; // Start count from here, down to 0. Default 1 ms.
26 unsigned long _avr_timer_cntcurr=0; // Current internal count of lms ticks
27 volatile unsigned char A;
28 volatile unsigned char B = 0x01;
29 volatile unsigned char Blink = 0x08;
30 volatile unsigned char Sound = 0x00;
31 volatile unsigned long Count_Time = 300;
32 volatile unsigned long Blink_Time = 1000;
33 volatile unsigned long Sound_time = 2;
34 void TimerOn() {
35     // AVR timer/counter controller register TCCR0
36     TCCR0 = 0x0B; // bit3bit6=10: CTC mode (clear timer on compare)
37     // bit2bit1bit0=011: pre-scaler /64
38     // 00001011: 0x0B
39     // SO, 8 MHz clock or 8,000,000 /64 = 125,000 ticks/s
40     // Thus, TCNT0 register will count at 125,000 ticks/s
41     // AVR output compare register OCR0.
42     OCR0 = 125; // Timer interrupt will be generated when TCNT0==OCR0
43     // We want a 1 ms tick. 0.001 s * 125,000 ticks/s = 125
44     // So when TCNT0 register equals 125,
45     // 1 ms has passed. Thus, we compare to 125.
46     // AVR timer interrupt mask register
47     TIMSK = 0x02; // bit1: OCIE0 -- enables compare match interrupt
48     //Initialize avr counter
49     TCNT0=0;
50     _avr_timer_cntcurr = _avr_timer_M;
51     // TimerISR will be called every _avr_timer_cntcurr milliseconds
52     //Enable global interrupts
53     SREG |= 0x80; // 0x80: 10000000
54 }
55 void TimerOff() {
56     TCCR0 = 0x00; // bit2bit1bit0=000: timer off
57 }
58 void TimerISR(){
59     TimerFlag = 1;
60     Count_Time++;
61     Blink_Time++;
62     Sound_time++;
63 }
64 // In our approach, the C programmer does not touch this ISR, but rather TimerIS
65 R()
66 ISR(TIMER0_COMP_vect)
67 {
68     // CPU automatically calls when TCNT0 == OCR0 (every 1 ms per TimerOn setting
69     s)
70     _avr_timer_cntcurr--; // Count down to 0 rather than up to TOP
71     if (_avr_timer_cntcurr == 0) { // results in a more efficient compare
72         TimerISR(); // Call the ISR that the user uses

```

May 03, 13 20:46

jdefo002_lab7_part3.c

Page 2/3

```

70     _avr_timer_cntcurr = _avr_timer_M;
71 }
72 }
73 // Set TimerISR() to tick every M ms
74 void TimerSet(unsigned long M) {
75     _avr_timer_M = M;
76     _avr_timer_cntcurr = _avr_timer_M;
77 }
78
79 enum Counter_States { InitState, SecState, ThirdState } CounterState;
80
81 void TckFct_Counter()
82 {
83     switch(CounterState)
84     {
85         //Transitions
86         case InitState:
87             B=0x01;
88             CounterState = SecState;
89             break;
90         case SecState:
91             B=0x02;
92             CounterState = ThirdState;
93             break;
94         case ThirdState:
95             B=0x04;
96             CounterState = InitState;
97             break;
98         default:
99             break;
100     }
101     switch(CounterState)
102     { // Actions
103         case InitState:
104             break;
105         default:
106             break;
107     }
108 }
109 enum Blink_State { BInitialState, Blink_two } BlinkState;
110
111 void TckFct_Blink()
112 {
113     switch(BlinkState)
114     {
115         //Transitions
116         case BInitialState: // Initial Transition
117             Blink =0x08;
118             BlinkState = Blink_two;
119             break;
120         case Blink_two:
121             Blink = 0x00;
122             BlinkState = BInitialState;
123             default:
124                 break;
125     }
126     switch(BlinkState)
127     { // Actions
128         case BInitialState:
129             break;
130         default:
131             break;
132     }
133 }
134 enum Sound_State { SInitialState, Sound_two } SoundState;
135
136 void TckFct_Sound()
137 {
138     switch(SoundState)
139     {
140         //Transitions
141         case SInitialState: // Initial Transition
142             if (A)

```

May 03, 13 20:46

jdefo002_lab7_part3.c

Page 3/3

```

143     {
144         Sound = 0x10;
145     }
146     else{
147         Sound = 0x00;
148     }
149     SoundState = Sound_two;
150     break;
151 case Sound_two:
152     Sound = 0x00;
153     SoundState = SInitialState;
154     default:
155         break;
156 }
157 switch(SoundState)
158 { // Actions
159     case SInitialState:
160         break;
161     default:
162         break;
163 }
164 }
165 int main(void)
166 {
167     DDRA = 0x00;
168     PORTA = 0xFF;
169     DDRB = 0xFF; // Set port B to output
170     PORTB = 0x00; // Init port B to 0s
171     TimerSet(1);
172     TimerOn();
173     CounterState = InitState;
174     BlinkState = BInitialState;
175     SoundState = SInitialState;
176     while(1) {
177         // User code
178         A = PINA;
179         A = (A&0x04)>>2;
180         if (Count_Time == 300){
181             TckFct_Counter();
182             Count_Time = 0;
183         }
184         if (Blink_Time == 1000){
185             TckFct_Blink();
186             Blink_Time = 0;
187         }
188         if (Sound_time == 2)
189         {
190             TckFct_Sound();
191             Sound_time = 0;
192         }
193         PORTB = B|Blink|Sound;
194         //while (!TimerFlag);
195         TimerFlag = 0;
196         // Note: For the above a better style would use a synchSM with TickSM()
197         // This example just illustrates the use of the ISR and flag
198     }
199 }

```

May 03, 13 20:48

jdefo002_lab7_part4.c

Page 1/4

```

1  /*      jdefo002_lab7_part4.c - 4/29/13
2  *      Name: Joshua DeForest-Williams Email: Jdefo002@ucr.edu
3  *      CS Login: jdefo002
4  *      Partner(s) Name: Ariana DeJaco E-mail: Adeja001@ucr.edu
5  *      Lab Section:022
6  *      Assignment: Lab #7 Exercise #4
7  *      Exercise Description:Connect LEDs to PB0, PB1, PB2, and PB3. Light
8  *      the three LEDs PB0, PB1, and PB2 in sequence for 1 second each.
9  *      Meanwhile, blink PB3 on 1 second and off 1 second.
10 *      EXTENSION: Modify the above example so the three LEDs light for
11 *      300 ms, while PB3's LED still blinks 1 second on and 1 second off.
12 *      EXTENSION 2: To the previous exercise's implementation, connect your
13 *      speaker's red wire to PB4 and black wire to ground. Add a third task
14 *      that toggles PB4 on for 2 ms and off for 2 ms as long as a switch on
15 *      PA2 is in the on position.
16 *      I acknowledge all content contained herein, excluding template or exampl
17 e
18 *      code, is my own original work.
19 */
20 #include <avr/io.h>
21 #include <avr/interrupt.h>
22 volatile unsigned char TimerFlag=0; // TimerISR() sets this to 1. C programmer s
23 hould clear to 0.
24 // Internal variables for mapping AVR's ISR to our cleaner TimerISR model.
25 unsigned long _avr_timer_M=1; // Start count from here, down to 0. Default 1 ms.
26 unsigned long _avr_timer_cntcurr=0; // Current internal count of lms ticks
27 volatile unsigned char A;
28 volatile unsigned char B = 0x01;
29 volatile unsigned char Blink = 0x08;
30 volatile unsigned char Sound = 0x00;
31 volatile unsigned long Sound_var = 1;
32 volatile unsigned long Count_Time = 300;
33 volatile unsigned long Blink_Time = 1000;
34 volatile unsigned long Sound_time = 1;
35 volatile unsigned long button_press = 20;
36 void TimerOn() {
37     // AVR timer/counter controller register TCCR0
38     TCCR0 = 0x0B; // bit3bit6=10: CTC mode (clear timer on compare)
39     // bit2bit1bit0=011: pre-scaler /64
40     // 00001011: 0x0B
41     // SO, 8 MHz clock or 8,000,000 /64 = 125,000 ticks/s
42     // Thus, TCNT0 register will count at 125,000 ticks/s
43     // AVR output compare register OCR0.
44     OCR0 = 125; // Timer interrupt will be generated when TCNT0==OCR0
45     // We want a 1 ms tick. 0.001 s * 125,000 ticks/s = 125
46     // So when TCNT0 register equals 125,
47     // 1 ms has passed. Thus, we compare to 125.
48     // AVR timer interrupt mask register
49     TIMSK = 0x02; // bit1: OCIE0 -- enables compare match interrupt
50     //Initialize avr counter
51     TCNT0=0;
52     _avr_timer_cntcurr = _avr_timer_M;
53     // TimerISR will be called every _avr_timer_cntcurr milliseconds
54     //Enable global interrupts
55     SREG |= 0x80; // 0x80: 10000000
56 }
57 void TimerOff() {
58     TCCR0 = 0x00; // bit2bit1bit0=000: timer off
59 }
60 void TimerISR(){
61     TimerFlag = 1;
62     Count_Time++;
63     Blink_Time++;
64     Sound_time++;
65     button_press++;
66 }
67 // In our approach, the C programmer does not touch this ISR, but rather TimerIS
68 R()
69 ISR(TIMER0_COMP_vect)
70 {
71     // CPU automatically calls when TCNT0 == OCR0 (every 1 ms per TimerOn setting
72 s)

```

May 03, 13 20:48

jdefo002_lab7_part4.c

Page 2/4

```

70     _avr_timer_cntcurr--; // Count down to 0 rather than up to TOP
71     if (_avr_timer_cntcurr == 0) { // results in a more efficient compare
72         TimerISR(); // Call the ISR that the user uses
73         _avr_timer_cntcurr = _avr_timer_M;
74     }
75 }
76 // Set TimerISR() to tick every M ms
77 void TimerSet(unsigned long M) {
78     _avr_timer_M = M;
79     _avr_timer_cntcurr = _avr_timer_M;
80 }
81
82 enum Counter_States { InitState, SecState, ThirdState } CounterState;
83
84 void TckFct_Counter()
85 {
86     switch(CounterState)
87     {
88         //Transitions
89         case InitState:
90             B=0x01;
91             CounterState = SecState;
92             break;
93         case SecState:
94             B=0x02;
95             CounterState = ThirdState;
96             break;
97         case ThirdState:
98             B=0x04;
99             CounterState = InitState;
100             break;
101         default:
102             break;
103     }
104     switch(CounterState)
105     { // Actions
106         case InitState:
107             break;
108         default:
109             break;
110     }
111 }
112 enum Blink_State { BInitialState, Blink_two } BlinkState;
113
114 void TckFct_Blink()
115 {
116     switch(BlinkState)
117     {
118         //Transitions
119         case BInitialState: // Initial Transition
120             Blink =0x08;
121             BlinkState = Blink_two;
122             break;
123         case Blink_two:
124             Blink = 0x00;
125             BlinkState = BInitialState;
126             default:
127                 break;
128     }
129     switch(BlinkState)
130     { // Actions
131         case BInitialState:
132             break;
133         default:
134             break;
135     }
136 }
137 enum Sound_State { SInitialState, Sound_two } SoundState;
138
139 void TckFct_Sound()
140 {
141     switch(SoundState)
142     {

```

May 03, 13 20:48

jdefo002_lab7_part4.c

Page 3/4

```

143 //Transitions
144 case SInitialState: // Initial Transition
145     if (!A)
146     {
147         Sound = 0x30;
148     }
149     else{
150         Sound = 0x00;
151     }
152     SoundState = Sound_two;
153     break;
154 case Sound_two:
155     Sound = 0x00;
156     SoundState = SInitialState;
157     default:
158         break;
159 }
160 switch(SoundState)
161 { // Actions
162     case SInitialState:
163         break;
164     default:
165         break;
166 }
167 }
168 enum Frequency_State { FInitialState} FrequencyState;
169
170 void TckFct_Freq()
171 {
172     switch(FrequencyState)
173     {
174     case FInitialState:
175     {
176         if (A & 0x01)
177         {
178             Sound_var++;
179         }
180         else if ((A & 0x02)&& Sound_var > 1){
181             Sound_var--;
182         }
183         FrequencyState = FInitialState;
184         break;
185     }
186     default:
187         break;
188     }
189     switch(SoundState)
190     { // Actions
191     case SInitialState:
192         break;
193     default:
194         break;
195     }
196 }
197 int main(void)
198 {
199     DDRA = 0x00;PORTA = 0xFF;
200
201     DDRB = 0xFF; // Set port B to output
202     PORTB = 0x00; // Init port B to 0s
203     TimerSet(1);
204     TimerOn();
205     CounterState = InitState;
206     BlinkState = BInitialState;
207     SoundState = SInitialState;
208     FrequencyState = FInitialState;
209     while(1) {
210         // User code
211         A = ~PINA;
212         if (Count_Time >= 300){
213             TckFct_Counter();
214             Count_Time = 0;
215         }

```

May 03, 13 20:48

jdefo002_lab7_part4.c

Page 4/4

```

216     if (Blink_Time >= 1000){
217         TckFct_Blink();
218         Blink_Time = 0;
219     }
220     if (button_press >= 20){
221         TckFct_Freq ();
222         button_press = 0;
223     }
224     if (Sound_time >= Sound_var)
225     {
226         TckFct_Sound();
227         Sound_time = 0;
228     }
229     PORTB = B|Blink|Sound;
230     while (!TimerFlag);
231     TimerFlag = 0;
232     // Note: For the above a better style would use a synchSM with TickSM()
233     // This example just illustrates the use of the ISR and flag
234 }
235 }

```