

Apr 26, 13 21:23

jdefo002\_lab5\_part1.c

Page 1/2

```

1  /* jdefo002_lab4_part1.c - April 22, 2013
2  * Name: Joshua Deforest-Williams E-mail jdefo002@ucr.edu
3  * CS Login: jdefo002
4  * Partner Name: Ariana DeJaco E-mail adeja001@ucr.edu
5  * Lab Section: 022
6  * Assignment: Lab#5 Exercise#1
7  * Exercise Description: Complete the following program by defining
8  * the function "void Write7Seg(unsigned char x)" that sets the
9  * 7-segment display to display x for values of 0 to 9. So if x is 0,
10 * illuminate segments A, B, C, D, E, and F (but not G). If x is 1,
11 * illuminate segments B and C. If x > 9, illuminate nothing.
12 * Carefully examine the above photo, the ATmega32 pinout, and the
13 * seven-segment pinout, to properly define the function.
14 */
15
16 // Use for debouncing the switch (How many Hz)
17 #define F_CPU 1000000
18
19 #include <avr/io.h>
20 #include <util/delay.h>
21 #include <avr/sfr_defs.h>
22
23 // Bit-access function
24 unsigned char GetBit(unsigned char x, unsigned char k)
25 {
26     return ((x & (0x01 << k)) != 0);
27 }
28
29 // Bit-set function
30 unsigned char SetBit(unsigned char x, unsigned char k, unsigned char b)
31 {
32     return (b ? x | (0x01 << k) : x & ~(0x01 << k));
33 }
34
35 // Current Port Definitions
36 #define LED_DDR          DDRC
37 #define LED_INPORT       PINC
38 #define LED_OUTPORT      PORTC
39 #define SENSOR_DDR      DDRA
40 #define SENSOR_INPORT    PINA
41 #define SENSOR_OUTPORT   PORTA
42 #define UNSEDEB_DDR      DDRB
43 #define UNSEDEB_PIN      PINB
44 #define UNSEDEB_PORT     PORTB
45 #define UNSEDD_DDR       DDRD
46 #define UNSEDD_PIN       PIND
47 #define UNSEDD_PORT      PORTD
48
49 // Additional macros not defines in sfr_defs.h
50 #define SET_PORT_BIT(OUTPORT, BIT)    OUTPORT |= (1 << BIT)
51 #define CLEAR_PORT_BIT(OUTPORT, BIT)  OUTPORT &= ~(1 << BIT)
52
53 //DDRA: Configures each of port A's physical pins to input (0) or output(1)
54 //PORTA: Writing to this register writes the port's physical pins
55 // (Write only)
56 //PINA: Reading this register reads the values of the port's physical pins
57 // (Read only)
58
59 // Creating a type "statetype" and making 2 variables of this type and initializi
60 ng them to InitReset
61 typedef enum { InitReset, Increment, Decrement, WaitForButtonPress, WaitForButto
62 nRelease, ShiftLeft, ShiftRight, ErrorState } statetype;
63 statetype CurrentState = InitReset;
64 statetype ActionState = InitReset;
65 #include <avr/io.h>
66
67 unsigned char Write7Seg(unsigned char x)
68 {
69     unsigned char segvalue = 0;
70     switch(x)
71     {
72         case 0:

```

Apr 26, 13 21:23

jdefo002\_lab5\_part1.c

Page 2/2

```

72         segvalue = 0x3F;
73         break;
74     case 1:
75         segvalue = 0x0C;
76         break;
77     case 2:
78         segvalue = 0x5B;
79         break;
80     case 3:
81         segvalue = 0x5E;
82         break;
83     case 4:
84         segvalue = 0x6C;
85         break;
86     case 5:
87         segvalue = 0x76;
88         break;
89     case 6:
90         segvalue = 0x77;
91         break;
92     case 7:
93         segvalue = 0x1C;
94         break;
95     case 8:
96         segvalue = 0x7F;
97         break;
98     case 9:
99         segvalue = 0x7C;
100        break;
101    default:
102        segvalue = 0x73;
103        break;
104    }
105    return segvalue;
106 }
107
108 int rand(void);
109
110 int main(void)
111 {
112     unsigned char cnt=0;
113     DDRA = 0x00; PORTA = 0xFF; // Config port A as inputs, init 1s
114     DDRC = 0xFF; PORTC = 0x00; // Config port C as outputs, init 0s
115     while(1) {
116         if (GetBit(PINA, 0)==0) { // Button pressed
117             cnt = rand() % 10; // cnt is rand num from 0-9
118             PORTC = Write7Seg(cnt);
119         }
120     }
121 }
122

```

Apr 26, 13 21:24

jdefo002\_lab5\_part2.c

Page 1/3

```

1  /* jdefo002_lab5_part2.c - April 22, 2013
2  * Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
3  * CS Login: jdefo002
4  * Partner Name: Ariana DeJaco Email adeja001@ucr.edu
5  * Lab Section: 022
6  * Assignment: Lab#5 Exercise#2
7  * Exercise Description: (From an earlier lab) Buttons are connected
8  * to PA0 and PA1. Output PORTC drives the 7-segment display initially
9  * displaying 0. Pressing PA0 increments the display (stopping at 9).
10 * Pressing PA1 decrements the display (stopping at 0). If both buttons
11 * are depressed (even if not initially simultaneously), the display
12 * resets to 0. Use a state machine (not synchronous) captured in C.
13 */
14
15 // Used for debouncing the button (How many Hz)
16 #define F_CPU 1000000
17
18 #include <avr/io.h>
19 #include <util/delay.h>
20 #include <avr/sfr_defs.h>
21
22 // Bit-access function
23 unsigned char GetBit(unsigned char x, unsigned char k)
24 {
25     return ((x & (0x01 << k)) != 0);
26 }
27
28 // Current Port Definitions
29 #define LED_DDR          DDRC
30 #define LED_INPORT       PINC
31 #define LED_OUTPORT      PORTC
32 #define SENSOR_DDR       DDRA
33 #define SENSOR_INPORT    PINA
34 #define SENSOR_OUTPORT   PORTA
35 #define UNSEDB_DDR       DDRB
36 #define UNSEDB_PIN       PINB
37 #define UNSEDB_PORT      PORTB
38 #define UNSEDD_DDR       DDRD
39 #define UNSEDD_PIN       PIND
40 #define UNSEDD_PORT      PORTD
41
42 // Additional macros not defines in sfr_defs.h
43 #define SET_PORT_BIT(OUTPORT, BIT)    OUTPORT |= (1 << BIT)
44 #define CLEAR_PORT_BIT(OUTPORT, BIT)  OUTPORT &= ~(1 << BIT)
45
46 //DDRA: Configures each of port A's physical pins to input (0) or output(1)
47 //PORTA: Writing to this register writes the port's physical pins
48 // (Write only)
49 //PINA: Reading this register reads the values of the port's physical pins
50 // (Read only)
51
52 enum Counter_States { InitReset, Increment, Decrement, WaitForButtonPress, WaitF
orButtonRelease, ErrorState } CounterState;
53
54 unsigned char TckFct_Counter(unsigned char inputData, unsigned char LedValue)
55 {
56     // Variable we are returning
57     unsigned char tempLedValue = LedValue;
58
59     switch(CounterState)
60     {
61         //Transitions
62         case InitReset: // Initial Transition
63             CounterState = WaitForButtonRelease;
64             break;
65         case Increment:
66             CounterState = WaitForButtonRelease;
67             break;
68         case Decrement:
69             CounterState = WaitForButtonRelease;
70             break;
71         case WaitForButtonPress:
72             if(inputData == 0x00)

```

Apr 26, 13 21:24

jdefo002\_lab5\_part2.c

Page 2/3

```

73     {
74         CounterState = WaitForButtonPress;
75     }
76     else if(inputData == 0x01)
77     {
78         if(tempLedValue < 9)
79         {
80             CounterState = Increment;
81         }
82         // Will stay in wait if not < 9
83     }
84     else if(inputData == 0x02)
85     {
86         if(tempLedValue > 0)
87         {
88             CounterState = Decrement;
89         }
90         // Will stay in wait if not > 0
91     }
92     else if(inputData == 0x03)
93     {
94         CounterState = InitReset;
95     }
96     else
97     {
98         CounterState = ErrorState;
99     }
100    break;
101    case WaitForButtonRelease:
102        if(inputData == 0x00)
103        {
104            CounterState = WaitForButtonPress;
105        }
106        break;
107    case ErrorState:
108        break;
109    default:
110        CounterState = ErrorState;
111        break;
112    }
113
114    switch(CounterState)
115    { // Actions
116        case Increment:
117            tempLedValue++;
118            break;
119        case Decrement:
120            tempLedValue--;
121            break;
122        case InitReset:
123            tempLedValue = 0x00;
124            break;
125        case WaitForButtonPress:
126            break;
127        case WaitForButtonRelease:
128            break;
129        case ErrorState:
130            tempLedValue = 0xFF;
131            break;
132        default:
133            break;
134    }
135    return tempLedValue;
136 }
137
138 unsigned char Write7Seg(unsigned char x)
139 {
140     unsigned char segvalue = 0;
141
142     switch(x)
143     {
144         case 0:
145             segvalue = 0x3F;

```

Apr 26, 13 21:24

jdefo002\_lab5\_part2.c

Page 3/3

```

146         break;
147     case 1:
148         segvalue = 0x0C;
149         break;
150     case 2:
151         segvalue = 0x5B;
152         break;
153     case 3:
154         segvalue = 0x5E;
155         break;
156     case 4:
157         segvalue = 0x6C;
158         break;
159     case 5:
160         segvalue = 0x76;
161         break;
162     case 6:
163         segvalue = 0x77;
164         break;
165     case 7:
166         segvalue = 0x1C;
167         break;
168     case 8:
169         segvalue = 0x7F;
170         break;
171     case 9:
172         segvalue = 0x7C;
173         break;
174     default:
175         segvalue = 0x73;
176         break;
177     }
178
179     return segvalue;
180 }
181
182 int main(void)
183 {
184     SENSOR_DDR = 0x00;
185     LED_DDR    = 0xFF;
186     UNSEDB_DDR= 0x00;
187     UNSEDD_DDR= 0x00;
188
189     // Initialize LEDs to off
190     LED_OUTPORT = 0x00;
191     unsigned char ButtonValue = 0;
192     unsigned char CurrentLEDValue = 0;
193     unsigned char SevenSegValue = 0;
194
195     while(1)
196     {
197         // Code to debounce the switch.
198         while (ButtonValue != (~SENSOR_INPORT & 0x03))
199         {
200             _delay_ms(50);
201             ButtonValue = (~SENSOR_INPORT & 0x03);
202         }
203
204         CurrentLEDValue = TckFct_Counter(ButtonValue, CurrentLEDValue);
205         SevenSegValue = Write7Seg(CurrentLEDValue);
206         LED_OUTPORT = SevenSegValue;
207     }
208 }

```

Apr 26, 13 21:25

jdefo002\_lab5\_part3.c

Page 1/3

```

1  /* jdefo002_lab5_part3.c - April 22, 2013
2  * Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
3  * CS Login: jdefo002
4  * Partner Name: Ariana DeJaco Email adejaoo1@ucr.edu
5  * Lab Section: 022
6  * Assignment: Lab#5 Exercise#3
7  * Exercise Description: Using the seven-segment display, buttons,
8  * and LEDs, create a very simple game. It can be any game, and it
9  * can be very simple -- but each student pair must create and
10 * implement their own game. Use a state machine (not synchronous)
11 * as appropriate.
12 */
13
14 // Used for debouncing the button (How many Hz)
15 #define F_CPU 1000000
16
17 #include <avr/io.h>
18 #include <util/delay.h>
19 #include <avr/sfr_defs.h>
20
21 // Bit-access function
22 unsigned char GetBit(unsigned char x, unsigned char k)
23 {
24     return ((x & (0x01 << k)) != 0);
25 }
26
27 // Current Port Definitions
28 #define LED_DDR          DDRC
29 #define LED_INPORT       PINC
30 #define LED_OUTPORT      PORTC
31 #define SENSOR_DDR      DDRA
32 #define SENSOR_INPORT    PINA
33 #define SENSOR_OUTPORT   PORTA
34 #define UNSEDB_DDR       DDRB
35 #define UNSEDB_PIN       PINB
36 #define UNSEDB_PORT      PORTB
37 #define UNSEDD_DDR       DDRD
38 #define UNSEDD_PIN       PIND
39 #define UNSEDD_PORT      PORTD
40
41 // Additional macros not defines in sfr_defs.h
42 #define SET_PORT_BIT(OUTPORT, BIT)      OUTPORT |= (1 << BIT)
43 #define CLEAR_PORT_BIT(OUTPORT, BIT)    OUTPORT &= ~(1 << BIT)
44
45 //DDRA: Configures each of port A's physical pins to input (0) or output(1)
46 //PORTA: Writing to this register writes the port's physical pins
47 // (Write only)
48 //PINA: Reading this register reads the values of the port's physical pins
49 // (Read only)
50
51 enum Counter_States { InitReset, LEDLight, WaitForButtonPress, WaitForButtonRelease, ErrorState } CounterState;
52
53 unsigned char Write7SegNumber(unsigned char x)
54 {
55     unsigned char segvalue = 0;
56
57     switch(x)
58     {
59         case 0:
60             segvalue = 0x3F;
61             break;
62         case 1:
63             segvalue = 0x0C;
64             break;
65         case 2:
66             segvalue = 0x5B;
67             break;
68         case 3:
69             segvalue = 0x5E;
70             break;
71         case 4:

```

Apr 26, 13 21:25

jdefo002\_lab5\_part3.c

Page 2/3

```

73     segvalue = 0x6C;
74     break;
75     case 5:
76         segvalue = 0x76;
77         break;
78     case 6:
79         segvalue = 0x77;
80         break;
81     case 7:
82         segvalue = 0x1C;
83         break;
84     case 8:
85         segvalue = 0x7F;
86         break;
87     case 9:
88         segvalue = 0x7C;
89         break;
90     default:
91         segvalue = 0x73;
92         break;
93 }
94
95 return segvalue;
96 }
97
98 unsigned char Write7SegLetter(unsigned char randNum, unsigned char guess)
99 {
100     unsigned char segvalue = 0;
101     unsigned char ranvalue = randNum;
102
103     /*if(randNum > guess)
104     {
105         segvalue = 0x6D;
106     }
107     else if(randNum < guess)
108     {
109         segvalue = 0x23;
110     }
111     else*/ if(randNum == guess)
112     {
113         segvalue = Write7SegNumber(ranvalue);
114     }
115     else
116     {
117         segvalue = 0x00;
118     }
119
120     return segvalue;
121 }
122
123 unsigned char TckFct_Counter(unsigned char inputData, unsigned char randNum, unsigned char sevSegVal)
124 {
125     // Variable we are returning
126     unsigned char tempLedValue = sevSegVal;
127
128     switch(CounterState)
129     {
130         //Transitions
131         case InitReset: // Initial Transition
132             CounterState = WaitForButtonRelease;
133             break;
134         case LEDLight:
135             CounterState = WaitForButtonRelease;
136             break;
137         case WaitForButtonPress:
138             if(inputData == 0x00)
139             {
140                 CounterState = WaitForButtonPress;
141             }
142             else
143             {
144                 CounterState = LEDLight;

```

Apr 26, 13 21:25

jdefo002\_lab5\_part3.c

Page 3/3

```

145         }
146         break;
147     case WaitForButtonRelease:
148         if(inputData == 0x00)
149         {
150             CounterState = WaitForButtonPress;
151         }
152         break;
153     /*case ErrorState:
154         break;*/
155     default:
156         CounterState = InitReset;//ErrorState;
157         break;
158 }
159
160 switch(CounterState)
161 { // Actions
162     case LEDLight:
163         tempLedValue = Write7SegLetter(inputData, randNum);
164         break;
165     case InitReset:
166         tempLedValue = 0x00;
167         break;
168     case WaitForButtonPress:
169         break;
170     case WaitForButtonRelease:
171         break;
172     /*case ErrorState:
173         tempLedValue = 0xFF;
174         break;*/
175     default:
176         break;
177 }
178 return tempLedValue;
179 }
180
181 int rand(void);
182
183 int main(void)
184 {
185     SENSOR_DDR = 0x00;
186     LED_DDR = 0xFF;
187     UNUSEDDB_DDR= 0x00;
188     UNUSEDDB_DDR= 0x00;
189
190     // Initialize LEDs to off
191     LED_OUTPORT = 0x00;
192     unsigned char ButtonValue = 0;
193     unsigned char SevenSegValue = 0;
194
195     unsigned char RandNumber = 4;
196
197     while(1)
198     {
199         // Code to debounce the switch.
200         while (ButtonValue != (~SENSOR_INPORT & 0x0F))
201         {
202             _delay_ms(50);
203             ButtonValue = (~SENSOR_INPORT & 0x0F);
204         }
205
206         SevenSegValue = TckFct_Counter(ButtonValue, RandNumber, SevenSeg
207 Value);
208         LED_OUTPORT = SevenSegValue;
209     }
210 }

```

Apr 26, 13 21:26

jdefo002\_lab5\_partchallenge.c

Page 1/3

```

1  /* jdefo002_lab5_partchallenge.c - April 22, 2013
2  * Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
3  * CS Login: jdefo002
4  * Partner Name: Ariana DeJaco Email adejaoo1@ucr.edu
5  * Lab Section: 022
6  * Assignment: Lab#5 Exercise#challenge
7  * Exercise Description: Create a more complex game -- make it fun!
8  * However, still use basic (not synchronous) state machines -- so no time-inter
  val behavior in your game.
9  */
10
11 // Used for debouncing the button (How many Hz)
12 #define F_CPU 1000000
13
14 #include <avr/io.h>
15 #include <util/delay.h>
16 #include <avr/sfr_defs.h>
17
18 // Bit-access function
19 unsigned char GetBit(unsigned char x, unsigned char k)
20 {
21     return ((x & (0x01 << k)) != 0);
22 }
23
24 // Current Port Definitions
25 #define LED_DDR          DDRC
26 #define LED_INPORT       PINC
27 #define LED_OUTPORT      PORTC
28 #define SENSOR_DDR       DDRA
29 #define SENSOR_INPORT     PINA
30 #define SENSOR_OUTPORT    PORTA
31 #define UNSEDB_DDR        DDRB
32 #define UNSEDB_PIN        PINB
33 #define UNSEDB_PORT       PORTB
34 #define UNSEDD_DDR        DDRD
35 #define UNSEDD_PIN        PIND
36 #define UNSEDD_PORT       PORTD
37
38 // Additional macros not defines in sfr_defs.h
39 #define SET_PORT_BIT(OUTPORT, BIT)    OUTPORT |= (1 << BIT)
40 #define CLEAR_PORT_BIT(OUTPORT, BIT)  OUTPORT &= ~(1 << BIT)
41
42 //DDRA: Configures each of port A's physical pins to input (0) or output(1)
43 //PORTA: Writing to this register writes the port's physical pins
44 // (Write only)
45 //PINA: Reading this register reads the values of the port's physical pins
46 // (Read only)
47
48
49 enum Counter_States { InitReset, LEDLight, WaitForButtonPress, WaitForButtonRelease } CounterState;
50
51 unsigned char Write7SegNumber(unsigned char x)
52 {
53     unsigned char segvalue = 0;
54
55     switch(x)
56     {
57         case 0:
58             segvalue = 0x3F;
59             break;
60         case 1:
61             segvalue = 0x0C;
62             break;
63         case 2:
64             segvalue = 0x5B;
65             break;
66         case 3:
67             segvalue = 0x5E;
68             break;
69         case 4:
70             segvalue = 0x6C;
71             break;

```

Apr 26, 13 21:26

jdefo002\_lab5\_partchallenge.c

Page 2/3

```

72     case 5:
73         segvalue = 0x76;
74         break;
75     case 6:
76         segvalue = 0x77;
77         break;
78     case 7:
79         segvalue = 0x1C;
80         break;
81     case 8:
82         segvalue = 0x7F;
83         break;
84     case 9:
85         segvalue = 0x7C;
86         break;
87     default:
88         segvalue = 0x73;
89         break;
90 }
91
92 return segvalue;
93 }
94
95 unsigned char Write7SegLetter(unsigned char randNum, unsigned char guess)
96 {
97     unsigned char segvalue = 0;
98     unsigned char ranvalue = randNum;
99
100     if(randNum < guess)
101     {
102         segvalue = 0x6D;
103     }
104     else if(randNum > guess)
105     {
106         segvalue = 0x23;
107     }
108     else if(randNum == guess)
109     {
110         segvalue = Write7SegNumber(ranvalue);
111     }
112     else
113     {
114         segvalue = 0x00;
115     }
116
117     return segvalue;
118 }
119
120 unsigned char TckFct_Counter(unsigned char inputData, unsigned char randNum, unsigned char sevSegVal)
121 {
122     // Variable we are returning
123     unsigned char tempLedValue = sevSegVal;
124
125     switch(CounterState)
126     {
127         //Transitions
128         case InitReset: // Initial Transition
129             CounterState = WaitForButtonRelease;
130             break;
131         case LEDLight:
132             CounterState = WaitForButtonRelease;
133             break;
134         case WaitForButtonPress:
135             if(inputData == 0x00)
136             {
137                 CounterState = WaitForButtonPress;
138             }
139             else
140             {
141                 CounterState = LEDLight;
142             }
143             break;

```

Apr 26, 13 21:26

jdefo002\_lab5\_partchallenge.c

Page 3/3

```

144         case WaitForButtonRelease:
145             if(inputData == 0x00)
146             {
147                 CounterState = WaitForButtonPress;
148             }
149             break;
150         default:
151             CounterState = InitReset;//ErrorState;
152             break;
153     }
154
155     switch(CounterState)
156     { // Actions
157         case LEDLight:
158             tempLedValue = Write7SegLetter(inputData, randNum);
159             break;
160         case InitReset:
161             tempLedValue = 0x00;
162             break;
163         case WaitForButtonPress:
164             break;
165         case WaitForButtonRelease:
166             break;
167         default:
168             break;
169     }
170     return tempLedValue;
171 }
172
173 int rand(void);
174
175 int main(void)
176 {
177     SENSOR_DDR = 0x00;
178     LED_DDR    = 0xFF;
179     UNUSED_B_DDR = 0x00;
180     UNUSED_D_DDR = 0x00;
181
182     // Initialize LEDs to off
183     LED_OUTPORT = 0x00;
184     unsigned char ButtonValue = 0;
185     unsigned char SevenSegValue = 0;
186
187     unsigned char RandNumber = 4;
188
189     while(1)
190     {
191         // Code to debounce the switch.
192         while (ButtonValue != (~SENSOR_INPORT & 0x0F))
193         {
194             _delay_ms(50);
195             ButtonValue = (~SENSOR_INPORT & 0x0F);
196         }
197
198         SevenSegValue = TckFct_Counter(ButtonValue, RandNumber, SevenSeg
199 Value);
200         LED_OUTPORT = SevenSegValue;
201     }

```