

Apr 24, 13 17:05

adeja001_lab4_part1.c

Page 1/2

```

1  /* adeja001_lab4_part1.c - April 22, 2013
2  * Name: Ariana DeJaco E-mail adeja001@ucr.edu
3  * CS Login: adeja001
4  * Partner Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
5  * Lab Section: 022
6  * Assignment: Lab#4 Exercise#1
7  * Exercise Description: A car has a fuel-level sensor that sets PA3..PA0 to a v
8  * A series of LEDs connected to PC5..PC0 should light to graphically indicate t
9  * he fuel level. If the fuel level is 1 or 2,
10 * PC5 lights. If the level is 3 or 4, PC5 and PC4 light. Level 5-6 lights PC5..
11 * PC3. 7-9 lights PC5..PC2. 10-12
12 * lights PC5..PC1. 13-15 lights PC5..PC0. Also, PC6 connects to a "Low fuel" ic
13 on, which should light if the level is 4 or less.
14 * (The example below shows the display for a fuel level of 3).
15 */
16
17 #include <avr/io.h>
18 #include <avr/sfr_defs.h>
19
20 // Bit-access function
21 unsigned char GetBit(unsigned char x, unsigned char k)
22 {
23     return ((x & (0x01 << k)) != 0);
24 }
25
26 // Current Port Definitions
27 #define LED_DDR          DDRC
28 #define LED_INPORT       PINC
29 #define LED_OUTPORT      PORTC
30 #define SENSOR_DDR       DDRA
31 #define SENSOR_INPORT    PINA
32 #define SENSOR_OUTPORT   PORTA
33 #define UNSEDB_DDR       DDRB
34 #define UNSEDB_PIN       PINB
35 #define UNSEDB_PORT      PORTB
36 #define UNSEDD_DDR       DDRD
37 #define UNSEDD_PIN       PIND
38 #define UNSEDD_PORT      PORTD
39
40 // All "fuel levels" defined as macros, makes easy to change
41 #define LEVEL1 2
42 #define LEVEL2 4
43 #define LEVEL3 6
44 #define LEVEL4 9
45 #define LEVEL5 12
46
47 // Additional macros not defines in sfr_defs.h
48 #define SET_PORT_BIT(OUTPORT, BIT)    OUTPORT |= (1 << BIT)
49 #define CLEAR_PORT_BIT(OUTPORT, BIT)  OUTPORT &= ~(1 << BIT)
50
51 // Function to set all the LED lights according to the Fuel Level.
52 unsigned char TickFct_gas(char SensorValue)
53 {
54     unsigned char ledValue;
55
56     if (SensorValue > LEVEL5)
57     {
58         ledValue = 0x3F;
59     }
60     else if (SensorValue > LEVEL4)
61     {
62         ledValue = 0x3E;
63     }
64     else if (SensorValue > LEVEL3)
65     {
66         ledValue = 0x3C;
67     }
68     else if (SensorValue > LEVEL2)
69     {
70         ledValue = 0x38;
71     }

```

Apr 24, 13 17:05

adeja001_lab4_part1.c

Page 2/2

```

70     else if (SensorValue > LEVEL1)
71     {
72         ledValue = 0x30;
73     }
74     else if (SensorValue > 0)
75     {
76         ledValue = 0x20;
77     }
78     else
79     {
80         ledValue = 0;
81     }
82
83     if (SensorValue <= LEVEL2)
84     {
85         ledValue |= 0x40;
86     }
87     return ledValue;
88 }
89
90 //DDRA: Configures each of port A's physical pins to input (0) or output(1)
91 //PORTA: Writing to this register writes the port's physical pins
92 // (Write only)
93 //PINA: Reading this register reads the values of the port's physical pins
94 // (Read only)
95 int main(void)
96 {
97     SENSOR_DDR = 0x00;    // Configure port A's 8 pins as inputs
98     LED_DDR = 0xFF;       // Configure port C's 8 pins as outputs,
99     UNSEDB_DDR = 0x00;
100    UNSEDD_DDR = 0x00;
101
102    unsigned char SensorValue = 0;
103    unsigned char SavedSensorValue = 0;
104
105    while(1)
106    {
107        // We are working with inverted logic so we need to use the complimen
108        t so
109        // we use the tilda.. Must also mask out bits we are not using
110        // (in this case A4-A7)
111        SensorValue = (~SENSOR_INPORT & 0x0F);
112
113        // Do not need to use the tilda (compliment) when putting into the fu
114        nction
115        // because we need it to output a 1 in order for LED to light up.
116        LED_OUTPORT = TickFct_gas(SensorValue);
117    }
118
119
120
121
122

```

Apr 24, 13 17:05

adeja001_lab4_part2.c

Page 1/3

```

1  /* adeja001_lab4_part2.c - April 22, 2013
2  * Name: Ariana DeJaco Email adeja001@ucr.edu
3  * CS Login: adeja001
4  * Partner Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
5  * Lab Section: 022
6  * Assignment: Lab#4 Exercise#2
7  * Exercise Description: Buttons are connected to PA0 and PA1. Output for PORTC
  is initially 0.
8  * Pressing PA0 increments PORTC. Pressing PA1 decrements. If both are pressed,
  PORTC is reset.
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 enum Counter_States { InitReset, Increment, Decrement, WaitForButtonPress, WaitF
  orButtonRelease, ErrorState } CounterState;
49
50 unsigned char TckFct_Counter(unsigned char inputData, unsigned char LedValue)
51 {
52     // Variable we are returning
53     unsigned char tempLedValue = LedValue;
54
55     switch(CounterState)
56     {
57         //Transitions
58         case InitReset: // Initial Transition
59             CounterState = WaitForButtonRelease;
60             break;
61         case Increment:
62             CounterState = WaitForButtonRelease;
63             break;
64         case Decrement:
65             CounterState = WaitForButtonRelease;
66             break;
67         case WaitForButtonPress:
68             if(inputData == 0x00)
69             {
70                 CounterState = WaitForButtonPress;

```

Apr 24, 13 17:05

adeja001_lab4_part2.c

Page 2/3

```

71     }
72     else if(inputData == 0x01)
73     {
74         if(tempLedValue < 9)
75         {
76             CounterState = Increment;
77         }
78         // Will stay in wait if not < 9
79     }
80     else if(inputData == 0x02)
81     {
82         if(tempLedValue > 0)
83         {
84             CounterState = Decrement;
85         }
86         // Will stay in wait if not > 0
87     }
88     else if(inputData == 0x03)
89     {
90         CounterState = InitReset;
91     }
92     else
93     {
94         CounterState = ErrorState;
95     }
96     break;
97 case WaitForButtonRelease:
98     if(inputData == 0x00)
99     {
100         CounterState = WaitForButtonPress;
101     }
102     break;
103 case ErrorState:
104     break;
105 default:
106     CounterState = ErrorState;
107     break;
108 }
109
110 switch(CounterState)
111 { // Actions
112     case Increment:
113         tempLedValue++;
114         break;
115     case Decrement:
116         tempLedValue--;
117         break;
118     case InitReset:
119         tempLedValue = 0x00;
120         break;
121     case WaitForButtonPress:
122         break;
123     case WaitForButtonRelease:
124         break;
125     case ErrorState:
126         tempLedValue = 0xFF;
127         break;
128     default:
129         break;
130 }
131 return tempLedValue;
132 }
133
134 int main(void)
135 {
136     SENSOR_DDR = 0x00;
137     LED_DDR = 0xFF;
138     UNSEDB_DDR = 0x00;
139     UNSEDD_DDR = 0x00;
140
141     // Initialize LEDs to off
142     LED_OUTPORT = 0x00;
143     unsigned char ButtonValue = 0;

```

Apr 24, 13 17:05

adeja001_lab4_part2.c

Page 3/3

```
144     unsigned char CurrentLEDValue = 0;
145
146     while(1)
147     {
148         // Code to debounce the switch.
149         while (ButtonValue != (~SENSOR_INPORT & 0x03))
150         {
151             _delay_ms(50);
152             ButtonValue = (~SENSOR_INPORT & 0x03);
153         }
154         CurrentLEDValue = TckFct_Counter(ButtonValue, CurrentLEDValue);
155         LED_OUTPORT = CurrentLEDValue;
156     }
157 }
158 }
```

Apr 24, 13 17:05

adeja001_lab4_partchallenge.c

Page 1/4

```

1  /* adeja001_lab4_partchallenge.c - April 22, 2013
2  * Name: Ariana DeJaco E-mail adeja001@ucr.edu
3  * CS Login: adeja001
4  * Partner Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
5  * Lab Section: 022
6  * Assignment: Lab#4 Exercise#Challenge
7  * Exercise Description: Create your own festive lights display with 6 LEDs conn
8  ected to PC5-PC0
9  * lighting in some attractive sequence. Pressing the button on PA0 changes the
10 lights to the next
11 * configuration in the sequence.
12 */
13 // Use for debouncing the switch (How many Hz)
14 #define F_CPU 1000000
15 #include <avr/io.h>
16 #include <util/delay.h>
17 #include <avr/sfr_defs.h>
18
19 // Bit-access function
20 unsigned char GetBit(unsigned char x, unsigned char k)
21 {
22     return ((x & (0x01 << k)) != 0);
23 }
24
25 // Current Port Definitions
26 #define LED_DDR          DDRC
27 #define LED_INPORT       PINC
28 #define LED_OUTPORT      PORTC
29 #define SENSOR_DDR       DDRA
30 #define SENSOR_INPORT    PINA
31 #define SENSOR_OUTPORT   PORTA
32 #define UNSEDEB_DDR      DDRB
33 #define UNSEDEB_PIN       PINB
34 #define UNSEDEB_PORT      PORTB
35 #define UNSEDD_DDR       DDRD
36 #define UNSEDD_PIN        PIND
37 #define UNSEDD_PORT       PORTD
38
39 // Additional macros not defines in sfr_defs.h
40 #define SET_PORT_BIT(OUTPORT, BIT)      OUTPORT |= (1 << BIT)
41 #define CLEAR_PORT_BIT(OUTPORT, BIT)    OUTPORT &= ~(1 << BIT)
42
43 //DDRA: Configures each of port A's physical pins to input (0) or output(1)
44 //PORTA: Writing to this register writes the port's physical pins
45 // (Write only)
46 //PINA: Reading this register reads the values of the port's physical pins
47 // (Read only)
48
49 // Creating a type "statetype" and making 2 variables of this type and initializi
50 ng them to InitReset
51 typedef enum { InitReset, Increment, Decrement, WaitForButtonPress, WaitForButto
52 nRelease, ShiftLeft, ShiftRight, ErrorState } statetype;
53 statetype CurrentState = InitReset;
54 statetype ActionState = InitReset;
55
56 unsigned char TckFct_Counter(unsigned char inputData, unsigned char LedValue)
57 {
58     unsigned char TempLedValue = LedValue;
59
60     switch(CurrentState)
61     {
62         //Transitions
63         case InitReset: // Initial Transition
64             CurrentState = WaitForButtonRelease;
65             break;
66         case Increment:
67             if(inputData == 0x00)
68             {
69                 CurrentState = WaitForButtonPress;
70             }
71             else
72             {
73                 break;
74             }
75         case Decrement:
76             if(inputData == 0x00)
77             {
78                 CurrentState = WaitForButtonPress;
79             }
80             else
81             {
82                 break;
83             }
84         case ShiftRight:
85             if(inputData == 0x00)
86             {
87                 CurrentState = WaitForButtonPress;
88             }
89             else
90             {
91                 CurrentState = WaitForButtonRelease;
92                 ActionState = ShiftRight;
93             }
94             break;
95         case ShiftLeft:
96             if(inputData == 0x00)
97             {
98                 CurrentState = WaitForButtonPress;
99             }
100            else
101            {
102                CurrentState = WaitForButtonRelease;
103                ActionState = ShiftLeft;
104            }
105            break;
106        case WaitForButtonPress:
107            if(inputData == 0x00)
108            {
109                CurrentState = WaitForButtonPress;
110            }
111            else
112            {
113                switch(ActionState)
114                {
115                    case InitReset:
116                        CurrentState = ShiftLeft;
117                        break;
118                    case ShiftLeft:
119                        CurrentState = ShiftRight;
120                        break;
121                    case ShiftRight:
122                        CurrentState = Increment;
123                        break;
124                    case Increment:
125                        CurrentState = Decrement;
126                        break;
127                    case Decrement:
128                        CurrentState = ShiftLeft;
129                        break;
130                    default:
131                        CurrentState = ErrorState;
132                        break;
133                }
134            }
135        case WaitForButtonRelease:
136            if(inputData == 0x00)
137            {
138                CurrentState = WaitForButtonPress;
139            }
140            else
141            {
142                break;
143            }
144        }
145     }

```

Apr 24, 13 17:05

adeja001_lab4_partchallenge.c

Page 2/4

```

70     {
71         CurrentState = WaitForButtonRelease;
72         ActionState = Increment;
73     }
74     break;
75 case Decrement:
76     if(inputData == 0x00)
77     {
78         CurrentState = WaitForButtonPress;
79     }
80     else
81     {
82         CurrentState = WaitForButtonRelease;
83         ActionState = Decrement;
84     }
85     break;
86 case ShiftRight:
87     if(inputData == 0x00)
88     {
89         CurrentState = WaitForButtonPress;
90     }
91     else
92     {
93         CurrentState = WaitForButtonRelease;
94         ActionState = ShiftRight;
95     }
96     break;
97 case ShiftLeft:
98     if(inputData == 0x00)
99     {
100         CurrentState = WaitForButtonPress;
101     }
102     else
103     {
104         CurrentState = WaitForButtonRelease;
105         ActionState = ShiftLeft;
106     }
107     break;
108 case WaitForButtonPress:
109     if(inputData == 0x00)
110     {
111         CurrentState = WaitForButtonPress;
112     }
113     else
114     {
115         switch(ActionState)
116         {
117             case InitReset:
118                 CurrentState = ShiftLeft;
119                 break;
120             case ShiftLeft:
121                 CurrentState = ShiftRight;
122                 break;
123             case ShiftRight:
124                 CurrentState = Increment;
125                 break;
126             case Increment:
127                 CurrentState = Decrement;
128                 break;
129             case Decrement:
130                 CurrentState = ShiftLeft;
131                 break;
132             default:
133                 CurrentState = ErrorState;
134                 break;
135         }
136     }
137 case WaitForButtonRelease:
138     if(inputData == 0x00)
139     {
140         CurrentState = WaitForButtonPress;
141     }
142     else
143     {
144         break;
145     }

```

Apr 24, 13 17:05

adeja001_lab4_partchallenge.c

Page 3/4

```

143     case ErrorState:
144         break;
145     default:
146         CurrentState = ErrorState;
147         break;
148 }
149
150 switch(CurrentState)
151 {
152     case InitReset:
153         TempLedValue = 0x00;
154         break;
155     case Decrement:
156         TempLedValue = 0x3F;
157         break;
158     case Increment:
159         TempLedValue = 0x00;
160         break;
161     case ShiftLeft:
162         TempLedValue = 0x01;
163         break;
164     case ShiftRight:
165         TempLedValue = 0x20;
166         break;
167     case WaitForButtonRelease: //If button pressed or button release
168         d do the action state
169     case WaitForButtonPress:
170         switch(ActionState)
171         {
172             case Increment:
173                 if(TempLedValue < 0x3F)
174                 {
175                     TempLedValue++;
176                 }
177                 else
178                 {
179                     TempLedValue = 0x00;
180                 }
181                 break;
182             case Decrement:
183                 if(TempLedValue > 0x00)
184                 {
185                     TempLedValue--;
186                 }
187                 else
188                 {
189                     TempLedValue = 0x3F;
190                 }
191                 break;
192             case InitReset:
193                 TempLedValue = 0x00;
194                 break;
195             case ShiftLeft:
196                 if(TempLedValue < 0x20)
197                 {
198                     TempLedValue <= 1;
199                 }
200                 else
201                 {
202                     TempLedValue = 0x01;
203                 }
204                 break;
205             case ShiftRight:
206                 if(TempLedValue > 0x01)
207                 {
208                     TempLedValue >= 1;
209                 }
210                 else
211                 {
212                     TempLedValue = 0x20;
213                 }
214                 break;

```

Apr 24, 13 17:05

adeja001_lab4_partchallenge.c

Page 4/4

```

214     case WaitForButtonPress:
215         TempLedValue = 0xFF; // Indicate Error
216         break;
217     case WaitForButtonRelease:
218         TempLedValue = 0xFF; // Indicate Error
219         break;
220     case ErrorState:
221         TempLedValue = 0xFF; // Indicate Error
222         break;
223     default:
224         break;
225 }
226 break;
227 }
228 return TempLedValue;
229 }
230
231 int main(void)
232 {
233     SENSOR_DDR = 0x00;
234     LED_DDR = 0xFF;
235     UNSEDB_DDR= 0x00;
236     UNSEDD_DDR= 0x00;
237
238     LED_OUTPORT = 0x00;
239     unsigned char ButtonValue = 0;
240     unsigned char CurrentLEDValue = 0;
241
242     while(1)
243     {
244         // De-bounce the switch
245         while (ButtonValue != (~SENSOR_INPORT & 0x01))
246         {
247             _delay_ms(50);
248             ButtonValue = (~SENSOR_INPORT & 0x01);
249         }
250
251         CurrentLEDValue = TckFct_Counter(ButtonValue, CurrentLEDValue);
252         LED_OUTPORT = CurrentLEDValue;
253
254         // Delay for the blinking lights.
255         _delay_ms(250);
256     }
257 }

```