

Apr 17, 13 19:20

jdefo002_lab3_part1.c

Page 1/2

```

1  /*
2  * jdefo002_lab3_part1.c - April 15, 2013
3  * Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
4  * CS Login: jdefo002
5  * Partner Name: Ariana DeJaco E-mail:adeja001@ucr.edu
6  * Lab Section: 022
7  * Assignment: Lab#3 Exercise#1
8  * Exercise Description: PB0 and PB1 each connect to an LED, and PB0's
9  * LED is initially on. Pressing a button connected to PGetBit (INPUT_INPORT,0)
10 turns off
11 * PB0's LED and turns on PB1's LED, staying that way after button release.
12 * Pressing the button again turns off PB1's LED and turns on PB0's LED.
13 */
14
15 #include <avr/io.h>
16 #include <avr/sfr_defs.h>
17
18 // Current Port Definitions
19 #define INPUT_DDR          DDRA
20 #define INPUT_INPORT      PINA
21 #define INPUT_OUTPORT     PORTA
22 #define LED_DDR           DDRB
23 #define LED_INPORT        PINB
24 #define LED_OUTPORT       PORTB
25
26 // Additional macros not defines in sfr_defs.h
27 #define SET_PORT_BIT(OUTPORT, BIT)    OUTPORT |= (1 << BIT)
28 #define CLEAR_PORT_BIT(OUTPORT, BIT)  OUTPORT &= ~(1 << BIT)
29
30 unsigned char flag = 0;
31 unsigned char GetBit(unsigned char x, unsigned char k) {
32     return ((x & (0x01 << k)) != 0);
33 }
34 //volatile unsigned char TimerFlag=0; //raised by ISD, lowered by main code.
35 //void TimerISR() {
36 //    // TimerFlag=1;
37 //}
38 enum LED_States{Init, LED0,LED1, reset_flag_led1, reset_flag_led0}
39     LED_State;
40
41 void TickFct_LED(){
42     switch (LED_State){ // Transitions
43     case Init:
44         LED_State = LED0; //Initial State
45         break;
46     case LED0:
47         if (GetBit (INPUT_INPORT,0) && !flag){ // a=0
48             LED_State = LED0;
49         }
50         else if(!GetBit (INPUT_INPORT,0)){ //a =1
51             LED_State = reset_flag_led0;
52         }
53     }
54     break;
55     case LED1:
56         if (GetBit (INPUT_INPORT,0) && flag) // a=1
57         {
58             LED_State = LED1;
59         }
60         else if (!flag)
61         {
62             LED_State= LED0;
63         }
64     }
65     /*
66     else if (!GetBit (INPUT_INPORT,0)) //a =0
67     {
68         LED_State = reset_flag_led1;
69     }
70     }
71     break;
72     case reset_flag_led1:

```

Apr 17, 13 19:20

jdefo002_lab3_part1.c

Page 2/2

```

73         if (GetBit (INPUT_INPORT,0))
74         {
75             LED_State = LED0;
76         }
77         break;
78     case reset_flag_led0:
79         if (GetBit (INPUT_INPORT, 0))
80         {
81             LED_State = LED1;
82         }
83         break;
84     default:
85         LED_State =Init;
86         break;
87     }
88     switch (LED_State){ //State Actions
89     case LED0:
90         SET_PORT_BIT(LED_OUTPORT,0);
91         CLEAR_PORT_BIT(LED_OUTPORT,1);
92         flag = 0;
93         break;
94     case LED1:
95         SET_PORT_BIT(LED_OUTPORT,1);
96         CLEAR_PORT_BIT(LED_OUTPORT,0);
97         flag = 1;
98         break;
99     case reset_flag_led1:
100         flag = 0;
101         break;
102     case reset_flag_led0:
103         flag = 1;
104         break;
105     default:
106         break;
107     }
108 }
109 }
110
111 //DDRA: Configures each of port A's physical pins to input (0) or output (1)
112 //PORTA: Writing to this register writes the port's physical pins (Write only)
113 //PINA: Reading this register reads the values of the port's physical pins (Read
114 only)
115 int main(void)
116 {
117     LED_DDR = 0xFF; //Configures port B's 8 pins as outputs.
118     LED_OUTPORT = 0x00; //Initialize output on port C to 0x00;
119     INPUT_DDR = 0x00; // Configure Port A's 8 pins as inputs.
120     INPUT_OUTPORT = 0xFF; // Configure Port A's 8 pins as inputs.
121     LED_OUTPORT=0;
122     //TimerSet(500);
123     //TimerOn();
124     LED_State = Init;
125     while(1)
126     {
127         TickFct_LED ();
128         //while(!TimerFlag) {}
129         //TimerFlag = 0;
130     }
131 }

```

Apr 17, 13 19:20

jdefo002_lab3_part2.c

Page 1/3

```

1  /*
2  * jdefo002_lab3_part2.c - April 12, 2013
3  * Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
4  * CS Login: jdefo002
5  * Partner Name: Ariana DeJaco E-mail:adeja001@ucr.edu
6  * Lab Section: 022
7  * Assignment: Lab#3 Exercise#2
8  * Exercise Description: Buttons are connected to PA0 and PA1.
9  * Output for PORTC is initially 0. Pressing PA0 increments PORTC
10 * (stopping at 9). Pressing PA1 decrements PORTC (stopping at 0).
11 * If both buttons are depressed (even if not initially simultaneously),
12 * PORTC resets to 0.
13 */
14
15
16 #include <avr/io.h>
17 #include <avr/sfr_defs.h>
18
19
20 // Current Port Definitions
21 #define INPUT_DDR      DDRA
22 #define INPUT_INPORT    PINA
23 #define INPUT_OUTPORT   PORTA
24 #define COUNT_DDR      DDRC
25 #define COUNT_INPORT    PINC
26 #define COUNT_OUTPORT   PORTC
27
28 // Additional macros not defines in sfr_defs.h
29 #define SET_PORT_BIT(OUTPORT, BIT)    OUTPORT |= (1 << BIT)
30 #define CLEAR_PORT_BIT(OUTPORT, BIT)  OUTPORT &= ~(1 << BIT)
31
32 unsigned char count = 0;
33 unsigned char flag = 0;
34 unsigned char GetBit(unsigned char x, unsigned char k) {
35     return ((x & (0x01 << k)) != 0);
36 }
37
38 enum COUNT_States{Init, Increment ,Decrement, waiting, both}
39     COUNT_State;
40
41 void TickFct_CNT(){
42     switch (COUNT_State){ // Transitions
43     case Init:
44         if (GetBit (INPUT_INPORT,0) && count < 9){
45             COUNT_State = Increment; //Initial State
46         }
47         else if (GetBit (INPUT_INPORT,1)&& count > 0)
48         {
49             COUNT_State= Decrement;
50         }
51         else
52         {
53             COUNT_State = Init;
54         }
55         break;
56     case Increment:
57         if (GetBit (INPUT_INPORT,0)&& GetBit (INPUT_INPORT,1))
58         {
59             COUNT_State = both;
60         }
61         else if (GetBit(INPUT_INPORT,0) && !flag)
62         {
63             COUNT_State = Increment;
64         }
65         else
66         {
67             COUNT_State = waiting;
68         }
69         break;
70     case Decrement:
71         if (GetBit (INPUT_INPORT,0)&& GetBit (INPUT_INPORT,1))
72         {
73

```

Apr 17, 13 19:20

jdefo002_lab3_part2.c

Page 2/3

```

74         COUNT_State = both;
75     }
76     else if (GetBit(INPUT_INPORT,1) && !flag)
77     {
78         COUNT_State = Decrement;
79     }
80     else
81     {
82         COUNT_State = waiting;
83     }
84
85     break;
86
87 case waiting:
88     if (GetBit (INPUT_INPORT,0)&& GetBit (INPUT_INPORT,1))
89     {
90         COUNT_State = both;
91     }
92
93     if (GetBit (INPUT_INPORT, 0))
94     {
95         COUNT_State = waiting;
96     }
97     else if (GetBit (INPUT_INPORT,1))
98     {
99         COUNT_State = waiting;
100    }
101    else if (!GetBit (INPUT_INPORT,0) && !GetBit (INPUT_INPORT,1))
102    {
103        COUNT_State = Init;
104    }
105    break;
106 case both:
107     COUNT_State = both;
108     break;
109 default:
110     COUNT_State =Init;
111     break;
112 }
113 switch (COUNT_State){ //State Actions
114 case Increment:
115     count = count +1;
116     COUNT_State = waiting;
117     flag = 1;
118     break;
119 case Decrement:
120     count = count - 1;
121     COUNT_State= waiting;
122     flag = 1;
123     break;
124 case waiting:
125     COUNT_State = waiting;
126     flag = 0;
127     break;
128 case both:
129     count = 0;
130     flag = 0;
131     COUNT_State = Init;
132     break;
133 default:
134     break;
135 }
136
137 }
138
139
140 //DDRA: Configures each of port A's physical pins to input (0) or output (1)
141 //PORTA: Writing to this register writes the port's physical pins (Write only)
142 //PINA: Reading this register reads the values of the port's physical pins (Read
143 only)
144 int main(void)
145 {
146     COUNT_DDR = 0xFF; //Configures port B's 8 pins as outputs.

```

Apr 17, 13 19:20

jdefo002_lab3_part2.c

Page 3/3

```
146     COUNT_OUTPORT = 0x00; //Initialize output on port C to 0x00;
147     INPUT_DDR = 0x00; // Configure Port A's 8 pins as inputs.
148     INPUT_OUTPORT = 0xFF; // Configure Port A's 8 pins as inputs.
149     COUNT_State = Init;
150     while(1)
151     {
152         TickFct_CNT ();
153         COUNT_OUTPORT = count;
154     }
155
156 }
```

Apr 17, 13 19:20

jdefo002_lab3_part3.c

Page 1/3

```

1  /*
2  * jdefo002_lab3_part3.c - April 15, 2013
3  * Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
4  * CS Login: Jdefo002
5  * Partner Name: Ariana DeJaco E-mail:adeja001@ucr.edu
6  * Lab Section: 022
7  * Assignment: Lab#3 Exercise#2
8  * Exercise Description: A household has a digital combination deadbolt
9  * lock system on the doorway. The system has buttons on a keypad. Button
10 * 'X' connects to PA0, 'Y' to PA1, and '#' to PA2. Pressing and
11 * releasing '#', then pressing 'Y', should unlock the door by setting
12 * PB0 to 1. Any other sequence fails to unlock. Pressing a button from
13 * inside the house (PA7) locks the door (PB0=0). For debugging purposes,
14 * give each state a number, and always write the current state to PORTC
15 * (consider using the enum state variable). Also, be sure to check that
16 * only one button is pressed at a time.
17 */
18 #include <avr/io.h>
19 #include <avr/sfr_defs.h>
20 // Current Port Definitions
21 #define INPUT_DDR DDRA
22 #define INPUT_INPORT PINA
23 #define INPUT_OUTPORT PORTA
24 #define DOOR_DDR DDRB
25 #define DOOR_INPORT PINB
26 #define DOOR_OUTPORT PORTB
27 #define DEBUG_DDR DDRC
28 #define DEBUG_INPORT PINC
29 #define DEBUG_OUTPORT PORTC
30 // Additional macros not defines in sfr_defs.h
31 #define SET_PORT_BIT(OUTPORT, BIT) OUTPORT |= (1 << BIT)
32 #define CLEAR_PORT_BIT(OUTPORT, BIT) OUTPORT &= ~(1 << BIT)
33 unsigned char GetBit(unsigned char x_bit, unsigned char k) {
34     return ((x_bit & (0x01 << k)) != 0);
35 }
36 char state_num = 0;
37 char x;
38 char y;
39 char pound;
40 char lock;
41 char flag = 0;
42 enum Door_States{Initial,button_p, button_Pound_wait, wait_y, Release, button_Y,
43     button_lock} Door_State;
44 void TickFct_DOOR(){
45     switch (Door_State){ // Transitions
46     case Initial: // state 0
47         if ((x&&y)|| (x&&pound)|| (x&&lock)|| (y&&pound)|| (y&&lock)|| (pound
48             &&lock) ){
49             Door_State = Release;
50         }
51         else if (pound&&!y)
52         {
53             Door_State = button_p;
54         }
55         else if (lock){
56             Door_State = button_lock;
57         }
58         break;
59     case button_p: // state 1
60         if (!pound && y) {
61             Door_State = button_Y;
62         }
63         else if (!pound && !y) {
64             Door_State = wait_y;
65         }
66         else
67         {
68             Door_State = button_Pound_wait;
69         }
70         break;
71     case button_Pound_wait: // state 2

```

Apr 17, 13 19:20

jdefo002_lab3_part3.c

Page 2/3

```

72     if ((x&&y)|| (x&&pound)|| (x&&lock)|| (y&&pound)|| (y&&lock)|| (pound
73         &&lock) ){
74         Door_State = Release;
75     }
76     else if (pound && flag)
77     {
78         Door_State = button_Pound_wait;
79     }
80     else if (!pound && y) {
81         Door_State = button_Y;
82     }
83     else if (!pound && !y) {
84         Door_State = wait_y;
85     }
86     break;
87     case wait_y: // state 3
88         if ((pound && !y) || (pound && y))
89         {
90             Door_State = Initial;
91         }
92         else if (!pound && !y)
93         {
94             Door_State = wait_y;
95         }
96         else if (!pound && y)
97         {
98             Door_State = button_Y;
99         }
100    case Release: // state 4
101        if ((x&&y)|| (x&&pound)|| (x&&lock)|| (y&&pound)|| (y&&lock)|| (pound
102            &&lock) ){
103            Door_State = Release;
104        }
105        else
106        {
107            Door_State = Initial;
108        }
109        break;
110    case button_Y: // state 5
111        Door_State = Initial;
112        break;
113    case button_lock: // state 6
114        break;
115    default:
116        break;
117    }
118 }
119 switch (Door_State){ //State Actions
120     case button_p:
121         Door_State = button_Pound_wait;
122         flag = 1;
123         state_num = 1;
124         break;
125     case button_Pound_wait:
126         state_num = 2;
127         break;
128     case wait_y:
129         flag = 0;
130         state_num = 3;
131         break;
132     case Release:
133         state_num = 4;
134         break;
135     case button_Y:
136         SET_PORT_BIT(DOOR_OUTPORT, 0);
137         flag = 0;
138         state_num = 5;
139         break;
140     case button_lock:
141         CLEAR_PORT_BIT(DOOR_OUTPORT,0);
142         Door_State = Initial;

```

Apr 17, 13 19:20

jdefo002_lab3_part3.c

Page 3/3

```

143         state_num = 6;
144         break;
145     default:
146         state_num= 0;
147         break;
148     }
149 }
150 //DDRA: Configures each of port A's physical pins to input (0) or output (1)
151 //PORTA: Writing to this register writes the port's physical pins (Write only)
152 //PINA: Reading this register reads the values of the port's physical pins (Read
    only)
153 int main(void)
154 {
155     DOOR_DDR = 0xFF; //Configures port B's 8 pins as outputs.
156     DOOR_OUTPORT = 0x00; //Initialize output on port B to 0x00;
157     DEBUG_DDR = 0xFF; //Configures port C's 8 pins as outputs.
158     DEBUG_OUTPORT = 0x00; //Initialize output on port C to 0x00;
159     INPUT_DDR = 0x00; // Configure Port A's 8 pins as inputs.
160     INPUT_OUTPORT = 0xFF; // Configure Port A's 8 pins as inputs.
161     Door_State = Initial;
162     while(1)
163     {
164         x = GetBit(INPUT_INPORT, 0);
165         y = GetBit(INPUT_INPORT, 1);
166         pound = GetBit(INPUT_INPORT, 2);
167         lock = GetBit(INPUT_INPORT, 7);
168         TickFct_DOOR ();
169         DEBUG_OUTPORT = state_num;
170     }
171 }

```

Apr 17, 13 19:20

jdefo002_lab3_part4.c

Page 1/3

```

1  /*
2  * jdefo002_lab3_part4.c - April 17, 2013
3  * Name: Joshua DeForest-Williams E-mail jdefo002@ucr.edu
4  * CS Login: Jdefo002
5  * Partner Name: Ariana DeJaco E-mail:adeja001@ucr.edu
6  * Lab Section: 022
7  * Assignment: Lab#3 Exercise#4
8  * Exercise Description: A household has a digital combination deadbolt
9  * lock system on the doorway. The system has buttons on a keypad. Button
10 * 'X' connects to PA0, 'Y' to PA1, and '#' to PA2. Pressing and
11 * releasing '#', then pressing 'Y', should unlock the door by setting
12 * PB0 to 1. Any other sequence fails to unlock. Pressing a button from
13 * inside the house (PA7) locks the door (PB0=0). For debugging purposes,
14 * give each state a number, and always write the current state to PORTC
15 * (consider using the enum state variable). Also, be sure to check that
16 * only one button is pressed at a time.
17 * (Challenge) Extend the above door so that it can also be locked by entering t
18 * he earlier code.
19 */
20 #include <avr/io.h>
21 #include <avr/sfr_defs.h>
22 // Current Port Definitions
23 #define INPUT_DDR DDRA
24 #define INPUT_INPORT PINA
25 #define INPUT_OUTPORT PORTA
26 #define DOOR_DDR DDRB
27 #define DOOR_INPORT PINB
28 #define DOOR_OUTPORT PORTB
29 #define DEBUG_DDR DDRC
30 #define DEBUG_INPORT PINC
31 #define DEBUG_OUTPORT PORTC
32 // Additional macros not defines in sfr_defs.h
33 #define SET_PORT_BIT(OUTPORT, BIT) OUTPORT |= (1 << BIT)
34 #define CLEAR_PORT_BIT(OUTPORT, BIT) OUTPORT &= ~(1 << BIT)
35 unsigned char GetBit(unsigned char x_bit, unsigned char k) {
36     return ((x_bit & (0x01 << k)) != 0);
37 }
38 char state_num= 0;
39 char x;
40 char y;
41 char pound;
42 char lock;
43 char flag = 0;
44 char locked_flag = 0; // locked = 0, unlocked = 1;
45 enum Door_States{Initial,button_p, button_Pound_wait, wait_y, Release, button_Y,
46     button_lock,
47     locked_y_wait, locked_y} Door_State;
48 void TickFct_DOOR(){
49     switch (Door_State){ // Transitions
50     case Initial: // state 0
51         if ((x&y)|| (x&pound)|| (x&lock)|| (y&pound)|| (y&lock)|| (pound
52         &&lock) ){
53             Door_State = Release;
54         }
55         else if (pound&&y)
56         {
57             Door_State = button_p;
58         }
59         else if (lock){
60             Door_State = button_lock;
61         }
62         break;
63     case button_p: // state 1
64         if (!pound && y) {
65             Door_State = button_Y;
66         }
67         else if (!pound && !y) {
68             Door_State = wait_y;
69         }
70         else
71         {
72             Door_State = button_Pound_wait;
73         }
74     }
75 }

```

Apr 17, 13 19:20

jdefo002_lab3_part4.c

Page 2/3

```

71     }
72 }
73 break;
74 case button_Pound_wait: // state 2
75     if ((x&y)|| (x&pound)|| (x&lock)|| (y&pound)|| (y&lock)|| (pound
76     &&lock) ){
77         Door_State = Release;
78     }
79     else if (pound && flag && locked_flag)
80     {
81         Door_State = locked_y_wait;
82     }
83     else if (pound && flag && !locked_flag)
84     {
85         Door_State = button_Pound_wait;
86     }
87     else if (!pound && y) {
88         Door_State = button_Y;
89     }
90     else if (!pound && !y) {
91         Door_State = wait_y;
92     }
93 break;
94 case wait_y: // state 3
95     if ((pound && !y) || (pound && y))
96     {
97         Door_State = Initial;
98     }
99     else if (!pound && !y)
100     {
101         Door_State = wait_y;
102     }
103     else if (!pound && y)
104     {
105         Door_State = button_Y;
106     }
107 case Release: // state 4
108     if ((x&y)|| (x&pound)|| (x&lock)|| (y&pound)|| (y&lock)|| (pound
109     &&lock) ){
110         Door_State = Release;
111     }
112     else
113     {
114         Door_State = Initial;
115     }
116 break;
117 case button_Y: // state 5
118     Door_State = Initial;
119 break;
120 case button_lock: // state 6
121     break;
122 case locked_y_wait: // state 7
123     if ((pound && !y) || (pound && y))
124     {
125         Door_State = Initial;
126     }
127     else if (!pound && !y)
128     {
129         Door_State = locked_y_wait;
130     }
131     else if (!pound && y)
132     {
133         Door_State = locked_y;
134     }
135     break;
136 case locked_y:
137     break;
138 default:
139     break;
140 }
141 }

```

Apr 17, 13 19:20

jdefo002_lab3_part4.c

Page 3/3

```

142 switch (Door_State){ //State Actions
143     case button_p:
144         Door_State = button_Pound_wait;
145         flag = 1;
146         state_num = 1;
147         break;
148     case button_Pound_wait:
149         state_num = 2;
150         break;
151     case wait_y:
152         flag = 0;
153         state_num = 3;
154         break;
155     case Release:
156         state_num = 4;
157         break;
158     case button_Y:
159         SET_PORT_BIT(DOOR_OUTPORT, 0);
160         flag = 0;
161         state_num = 5;
162         locked_flag = 1;
163         break;
164     case button_lock:
165         CLEAR_PORT_BIT(DOOR_OUTPORT,0);
166         Door_State = Initial;
167         state_num = 6;
168         locked_flag = 0;
169         break;
170     case locked_y_wait:
171         state_num = 7;
172         Door_State = locked_y_wait;
173         break;
174     case locked_y:
175         state_num = 8;
176         CLEAR_PORT_BIT(DOOR_OUTPORT, 0);
177         locked_flag = 0;
178         Door_State = Initial;
179         break;
180     default:
181         state_num= 0;
182         break;
183 }
184 }
185 //DDRA: Configures each of port A's physical pins to input (0) or output (1)
186 //PORTA: Writing to this register writes the port's physical pins (Write only)
187 //PINA: Reading this register reads the values of the port's physical pins (Read
188 only)
189 int main(void)
190 {
191     DOOR_DDR = 0xFF; //Configures port B's 8 pins as outputs.
192     DOOR_OUTPORT = 0x00; //Initialize output on port B to 0x00;
193     DEBUG_DDR = 0xFF; //Configures port C's 8 pins as outputs.
194     DEBUG_OUTPORT = 0x00; //Initialize output on port C to 0x00;
195     INPUT_DDR = 0x00; // Configure Port A's 8 pins as inputs.
196     INPUT_OUTPORT = 0xFF; // Configure Port A's 8 pins as inputs.
197     Door_State = Initial;
198     while(1)
199     {
200         x = GetBit(INPUT_INPORT, 0);
201         y = GetBit(INPUT_INPORT, 1);
202         pound = GetBit(INPUT_INPORT, 2);
203         lock = GetBit(INPUT_INPORT, 7);
204         TickFct_DOOR ();
205         DEBUG_OUTPORT = state_num;
206     }

```