

```
1  /*
2   * FileName: main.c
3   * Version: 1
4   *
5   * Created: 9/19/2022 8:47:49 AM
6   * Author: Ethan Zeronik
7   *
8   * Operations: barebones io testing
9   *
10  * Hardware:
11  *   Atmega2560          micro controller
12  *   PORTA.4             hot water switch
13  *   PORTA.5             warm water switch
14  *   PORTA.6             cold water switch
15  *   PORTA.7             door open switch
16  *   PORTK.0             start pushbutton
17  *   PORTA.0             motor out 1
18  *   PORTA.1             motor out 2
19  *   PORTA.2             motor out 3
20  *   PORTA.3             motor out 4
21  *   PORTC.0             wash done led
22  *   PORTC.1             agitate led
23  *   PORTC.2             spin led
24  *   PORTC.4             drain valve
25  *   PORTC.5             hot water valve
26  *   PORTC.6             cold water valve
27  */
28
29  #include <avr/io.h>
30
31  #include "Debugger.h"
32
33  /* NOTE: Custom Macros */
34  // TODO: None
35
36  /* NOTE: Global Variables */
37  // TODO: None
38
39  /* NOTE: Function prototypes */
40  // inits IO ports
41  void IO_init(void);
42
43  /* NOTE: Application implementation */
44  // the main loop of the function, provided to us
45  int main(void)
46  {
47      initDebug();
48
49      IO_init();
50
51      while(1)
52      {
53      }
54  }
55
56  /* NOTE: Function implementations */
57  void IO_init(void)
```

```
58 {
59     // bottom nibble is motor and top is input switches
60     DDRA = 0x0f;
61     // turn on switch pullup resistors
62     PORTA = 0xf0;
63
64     // the start button
65     DDRK = 0x00;
66     PORTK = 0x01;
67
68     // the led output port
69     DDRC = 0xff;
70     PORTC = 0x00;
71 }
72
```

```
1  /*
2  * FileName: main.c
3  * Version: 1
4  *
5  * Created: 9/19/2022 8:49:04 AM
6  * Author: Ethan Zeronik
7  *
8  * Operations: basic washing machine functions
9  *
10 * Hardware:
11 *   Atmega2560          micro controller
12 *   PORTA.4             hot water switch
13 *   PORTA.5             warm water switch
14 *   PORTA.6             cold water switch
15 *   PORTA.7             door open switch
16 *   PORTK.0             start pushbutton
17 *   PORTA.0             motor out 1
18 *   PORTA.1             motor out 2
19 *   PORTA.2             motor out 3
20 *   PORTA.3             motor out 4
21 *   PORTC.0             wash done led
22 *   PORTC.1             agitate led
23 *   PORTC.2             spin led
24 *   PORTC.4             drain valve
25 *   PORTC.5             hot water valve
26 *   PORTC.6             cold water valve
27 */
28
29 #include <avr/io.h>
30
31 #define F_CPU 16000000UL
32 #include <util/delay.h>
33
34 #include "Debugger.h"
35
36 /* NOTE: Custom Macros */
37 #define startButton (PINK & 0x01)
38
39 #define hotButton (PINA & 0x10)
40 #define warmButton (PINA & 0x20)
41 #define coldButton (PINA & 0x40)
42 #define doorSwitch (PINA & 0x80)
43
44 #define outPort (PORTC)
45
46 #define doneLed(S) ((PORTC & ~0x01) | (S << 0))
47 #define agitateLed(S) ((PORTC & ~0x02) | (S << 1))
48 #define spinLed(S) ((PORTC & ~0x04) | (S << 2))
49
50 #define drainValve(S) ((PORTC & ~0x10) | (S << 4))
51 #define hotValve(S) ((PORTC & ~0x20) | (S << 5))
52 #define coldValve(S) ((PORTC & ~0x40) | (S << 6))
53
54 /* NOTE: Global Variables */
55 // TODO: None
56
57 /* NOTE: Function prototypes */
```

```
58 // inits IO ports
59 void IO_init(void);
60 // returns what valves need to be opened
61 void WASH_setValvesToInput(void);
62
63 /* NOTE: Application implementation */
64 // the main loop of the function, provided to us
65 int main(void)
66 {
67     initDebug();
68
69     IO_init();
70
71     while(1)
72     {
73         // while we have not started the washing machine
74         // and the door is open
75         while(!((startButton == 0x01) && (doorSwitch == 0x00)))
76         {
77             // do nothing
78         }
79
80         // NOTE: fill cycle
81         WASH_setValvesToInput();
82
83         _delay_ms(4000);
84
85         outPort = hotValve(0);
86         outPort = coldValve(0);
87
88         // NOTE: wash cycle
89         outPort = agitateLed(1);
90
91         for(size_t i = 0; i < 2; i++)
92         {
93             // move cw for 2 seconds
94             _delay_ms(2000);
95
96             // move ccw for 2 seconds
97             _delay_ms(2000);
98         }
99
100         outPort = agitateLed(0);
101
102         // NOTE: drain cycle
103         outPort = drainValve(1);
104
105         _delay_ms(4000);
106
107         outPort = drainValve(0);
108
109         // NOTE: fill again cycle
110         WASH_setValvesToInput();
111
112         _delay_ms(4000);
113
114         outPort = hotValve(0);
```

```
115     outPort = coldValve(0);
116
117     // NOTE: rinse cycle
118     // agitate 12 seconds
119
120     // NOTE: rinse again cycle
121     outPort = drainValve(1);
122
123     // wait 15 s
124     // spin for 9s
125
126     outPort = drainValve(0);
127
128     // NOTE: done with the wash
129     outPort = doneLed(1);
130
131     while(doorSwitch != 0x80)
132     {
133         // do nothing
134     }
135
136     outPort = doneLed(0);
137 }
138 }
139
140 /* NOTE: Function implementations */
141 void IO_init(void)
142 {
143     // bottom nibble is motor and top is input switches
144     DDRA = 0x0f;
145     // turn on switch pullup resistors
146     PORTA = 0xf0;
147
148     // the start button
149     DDRK = 0x00;
150     PORTK = 0x01;
151
152     // the led output port
153     DDRC = 0xff;
154     PORTC = 0x00;
155 }
156
157 void WASH_setValvesToInput(void)
158 {
159     if(hotButton != 0x00)
160     {
161         // hot on, cold off
162         outPort = hotValve(1);
163         outPort = coldValve(0);
164     }
165
166     if(warmButton != 0x00)
167     {
168         // hot on, cold on
169         outPort = hotValve(1);
170         outPort = coldValve(1);
171     }
```

```
172 |  
173 |     if(coldButton != 0x00)  
174 |     {  
175 |         // hot off, cold on  
176 |         outPort = hotValve(0);  
177 |         outPort = coldValve(1);  
178 |     }  
179 | }
```

```
1  /*
2  * FileName: main.c
3  * Version: 1
4  *
5  * Created: 9/19/2022 8:50:04 AM
6  * Author: Ethan Zeronik
7  *
8  * Operations: full featured washing machine functions
9  *
10 * Hardware:
11 *   Atmega2560          micro controller
12 *   PORTA.4             hot water switch
13 *   PORTA.5             warm water switch
14 *   PORTA.6             cold water switch
15 *   PORTA.7             door open switch
16 *   PORTK.0             start pushbutton
17 *   PORTA.0             motor out 1
18 *   PORTA.1             motor out 2
19 *   PORTA.2             motor out 3
20 *   PORTA.3             motor out 4
21 *   PORTC.0             wash done led
22 *   PORTC.1             agitate led
23 *   PORTC.2             spin led
24 *   PORTC.4             drain valve
25 *   PORTC.5             hot water valve
26 *   PORTC.6             cold water valve
27 */
28
29 #include <avr/io.h>
30
31 #define F_CPU 16000000UL
32 #include <util/delay.h>
33
34 #include "Debugger.h"
35 #include "StepperMotor.h"
36
37 /* NOTE: Custom Macros */
38 #define startButton (PINK & 0x01)
39
40 #define hotButton (PINA & 0x10)
41 #define warmButton (PINA & 0x20)
42 #define coldButton (PINA & 0x40)
43 #define doorSwitch (PINA & 0x80)
44
45 #define outPort (PORTC)
46
47 #define doneLed(S) ((PORTC & ~0x01) | (S << 0))
48 #define agitateLed(S) ((PORTC & ~0x02) | (S << 1))
49 #define spinLed(S) ((PORTC & ~0x04) | (S << 2))
50
51 #define drainValve(S) ((PORTC & ~0x10) | (S << 4))
52 #define hotValve(S) ((PORTC & ~0x20) | (S << 5))
53 #define coldValve(S) ((PORTC & ~0x40) | (S << 6))
54
55 /* NOTE: Global Variables */
56 // TODO: None
57
```

```
58  /* NOTE: Function prototypes */
59  // inits IO ports
60  void IO_init(void);
61  // returns what valves need to be opened
62  void WASH_setValvesToInput(void);
63
64  /* NOTE: Application implementation */
65  // the main loop of the function, provided to us
66  int main(void)
67  {
68      initDebug();
69
70      IO_init();
71      SM_init(&DDRA, &PORTA);
72
73      while(1)
74      {
75          // while we have not started the washing machine
76          // and the door is open
77          while(!((startButton == 0x01) && (doorSwitch == 0x00)))
78          {
79              // do nothing
80          }
81
82          // NOTE: fill cycle
83          WASH_setValvesToInput();
84
85          _delay_ms(4000);
86
87          outPort = hotValve(0);
88          outPort = coldValve(0);
89
90          // NOTE: wash cycle
91          outPort = agitateLed(1);
92
93          for(size_t i = 0; i < 2; i++)
94          {
95              // move cw for 2 seconds
96              SM_moveTime((StepperMotorRunMode_t)2, true, 2000, 5);
97
98              // move ccw for 2 seconds
99              SM_moveTime((StepperMotorRunMode_t)2, false, 2000, 5);
100          }
101
102          outPort = agitateLed(0);
103
104          // NOTE: drain cycle
105          outPort = drainValve(1);
106
107          _delay_ms(4000);
108
109          outPort = drainValve(0);
110
111          // NOTE: fill again cycle
112          WASH_setValvesToInput();
113
114          _delay_ms(4000);
```



```
115
116     outPort = hotValve(0);
117     outPort = coldValve(0);
118
119     // NOTE: rinse cycle
120     outPort = agitateLed(1);
121
122     for(size_t i = 0; i < 3; i++)
123     {
124         // move cw for 2 seconds
125         SM_moveTime((StepperMotorRunMode_t)2, true, 2000, 5);
126
127         // move ccw for 2 seconds
128         SM_moveTime((StepperMotorRunMode_t)2, false, 2000, 5);
129     }
130
131     outPort = agitateLed(0);
132
133     // NOTE: spin cycle
134     outPort = drainValve(1);
135     outPort = spinLed(1);
136
137     // spin for 9s
138     SM_moveTime((StepperMotorRunMode_t)1, true, 9000, 3);
139
140     outPort = drainValve(0);
141     outPort = spinLed(0);
142
143
144     // NOTE: done with the wash
145     outPort = doneLed(1);
146
147     while(doorSwitch != 0x80)
148     {
149         // do nothing
150     }
151
152     outPort = doneLed(0);
153 }
154
155
156 /* NOTE: Function implementations */
157 void IO_init(void)
158 {
159     // bottom nibble is motor and top is input switches
160     DDRA = 0x0f;
161     // turn on switch pullup resistors
162     PORTA = 0xf0;
163
164     // the start button
165     DDRK = 0x00;
166     PORTK = 0x01;
167
168     // the led output port
169     DDRC = 0xff;
170     PORTC = 0x00;
171 }
```

```
172
173 void WASH_setValvesToInput(void)
174 {
175     if(hotButton != 0x00)
176     {
177         // hot on, cold off
178         outPort = hotValve(1);
179         outPort = coldValve(0);
180     }
181
182     if(warmButton != 0x00)
183     {
184         // hot on, cold on
185         outPort = hotValve(1);
186         outPort = coldValve(1);
187     }
188
189     if(coldButton != 0x00)
190     {
191         // hot off, cold on
192         outPort = hotValve(0);
193         outPort = coldValve(1);
194     }
195 }
196
```

```
1  /*
2   * FileName: StepperMotor.h
3   * Version: 1
4   *
5   * Created: 9/14/2022 2:00 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: header for the stepper motor submobule
9   */
10
11 #ifndef StepperMotor_h_INCLUDED
12 #define StepperMotor_h_INCLUDED
13
14 #if defined(__cplusplus)
15 extern "C" {
16 #endif
17
18 #include <stdbool.h>
19 #include <stdint.h>
20 #include <stdio.h>
21
22 /* NOTE: Custom Types */
23 // typing for the stepper motor enum
24 typedef enum StepperMotorRunMode_t
25 {
26     // wave step mode
27     Wave = 0,
28     // wave step mode
29     Full = 1,
30     // wave step mode
31     Half = 2,
32 } StepperMotorRunMode_t;
33
34 /* NOTE: Function prototypes */
35 // inits IO for the stepper motor
36 // takes a pointer to the port to use, assumes botom nibble
37 void SM_init(volatile uint8_t * pRegister, volatile uint8_t * pPort);
38
39 // moves the motor in the given mode to the given distance
40 // distance is in units of rotation
41 void SM_move(StepperMotorRunMode_t mode, double distance);
42
43 // moves the motor in the given mode to the given position
44 // distance is in units of degrees
45 void SM_movePosition(StepperMotorRunMode_t mode, uint16_t distance);
46
47 // moves the motor in the given mode and the given direction for the given time
48 // 1 is CW and 0 is CCW
49 // both times are in ms
50 void SM_moveTime(StepperMotorRunMode_t mode, bool direction, double time, double stepTime);
51
52 #if defined(__cplusplus)
53 } /* extern "C" */
54 #endif
55
56 #endif // StepperMotor_h_INCLUDED
```

```
1  /*
2  * FileName: StepperMotor.c
3  * Version: 1
4  *
5  * Created: 9/14/2022 2:00 PM
6  * Author: Ethan Zeronik
7  *
8  * Operations: run the stepper motor in one of three modes
9  */
10
11 /* NOTE: Includes */
12 #include "StepperMotor.h"
13
14 // TODO: move this
15 #define __DELAY_BACKWARD_COMPATIBLE__
16 #define F_CPU 16000000UL
17 #include <util/delay.h>
18
19 /* NOTE: Local declarations */
20 typedef struct StepperMotorModeData_t
21 {
22     // size of the array
23     size_t      arraySize;
24     // pointer to the array
25     uint8_t const * const pArray;
26     // number of steps to take for desired rotation
27     uint32_t      steps;
28 } StepperMotorModeData_t;
29
30 // returns the amount of steps needed for the given mode
31 // rotation is in radians (I think)
32 StepperMotorModeData_t getModeAndSteps(StepperMotorRunMode_t mode, double rotation);
33
34 /* NOTE: Global Variables */
35 // implementation of the wave step map
36 static uint8_t sWaveStepMap[4] = {
37     0x01,
38     0x02,
39     0x04,
40     0x08,
41 };
42
43 // implementation of the full step map
44 static uint8_t sFullStepMap[4] = {
45     0x03,
46     0x06,
47     0x0c,
48     0x09,
49 };
50
51 // implementation of the wave step map
52 static uint8_t sHalfStepMap[8] = {
53     0x09,
54     0x01,
55     0x03,
56     0x02,
57     0x06,
```

```
58     0x04,  
59     0x0c,  
60     0x08,  
61 };  
62  
63 // instance pointer to the motor port  
64 static volatile uint8_t * sMotorPort;  
65  
66 /* NOTE: Function implementations */  
67 void SM_init(volatile uint8_t * pRegister, volatile uint8_t * pPort)  
68 {  
69     // configure port register  
70     *pRegister = (*pRegister & 0xf0) | 0x0f;  
71  
72     // turn on pullup resisitors on the bottom nibble  
73     *pPort = 0x00;  
74  
75     // save the port pointer to the static var  
76     sMotorPort = pPort;  
77 }  
78  
79 void SM_move(StepperMotorRunMode_t mode, double distance)  
80 {  
81     StepperMotorModeData_t data = getModeAndSteps(mode, distance);  
82  
83     for(uint32_t i = 0, j = 0; i < data.steps; i++)  
84     {  
85         *sMotorPort = data.pArray[j++];  
86  
87         if(j >= data.arraySize)  
88         {  
89             j = 0;  
90         }  
91  
92         _delay_ms(3);  
93     }  
94  
95     *sMotorPort = 0x00;  
96 }  
97  
98 void SM_movePosition(StepperMotorRunMode_t mode, uint16_t distance)  
99 {  
100     SM_move(mode, ((double)distance / 360));  
101 }  
102  
103 void SM_moveTime(StepperMotorRunMode_t mode, bool direction, double time, double stepTime)  
104 {  
105     StepperMotorModeData_t data = getModeAndSteps(mode, 0);  
106  
107     for(uint32_t i = 0, j = (direction ? data.arraySize : 0); i < (time / stepTime); i++)  
108     {  
109         *sMotorPort = data.pArray[(direction ? j-- : j++)];  
110  
111         if(j >= data.arraySize || j <= 0)  
112         {  
113             j = (direction ? data.arraySize : 0);  
114         }
```

```
115 |
116 |     _delay_ms(stepTime);
117 | }
118 |
119 | *sMotorPort = 0x00;
120 | }
121 |
122 | /* NOTE: Local function implementations */
123 | StepperMotorModeData_t getModeAndSteps(StepperMotorRunMode_t mode, double rotation)
124 | {
125 |     uint8_t * pArray = NULL;
126 |     uint8_t size = 0;
127 |     uint32_t steps = 0;
128 |
129 |     switch(mode)
130 |     {
131 |     case Wave:
132 |     {
133 |         pArray = sWaveStepMap;
134 |         size = sizeof(sWaveStepMap) / sizeof(sWaveStepMap[0]);
135 |         steps = (rotation * 2048);
136 |     }
137 |     break;
138 |     case Full:
139 |     {
140 |         pArray = sFullStepMap;
141 |         size = sizeof(sFullStepMap) / sizeof(sFullStepMap[0]);
142 |         steps = (rotation * 2048);
143 |     }
144 |     break;
145 |     case Half:
146 |     {
147 |         pArray = sHalfStepMap;
148 |         size = sizeof(sHalfStepMap) / sizeof(sHalfStepMap[0]);
149 |         steps = (rotation * 4096);
150 |     }
151 |     break;
152 |     default:
153 |         break;
154 |     };
155 |
156 |     return (StepperMotorModeData_t){
157 |         .pArray = pArray,
158 |         .steps = steps,
159 |         .arraySize = size,
160 |     };
161 | }
```