```c
/*
 * FileName: main.c
 * Version: 1
 *
 * Created: 9/19/2022 8:47:49 AM
 * Author: Ethan Zeronik
 *
 * Operations: barebones io testing
 *
 * Hardware:
 *   Atmega2560          micro controller
 *   PORTA.4             hot water switch
 *   PORTA.5             warm water switch
 *   PORTA.6             cold water switch
 *   PORTA.7             door open switch
 *   PORTK.0             start pushbutton
 *   PORTA.0             motor out 1
 *   PORTA.1             motor out 2
 *   PORTA.2             motor out 3
 *   PORTA.3             motor out 4
 *   PORTC.0             wash done led
 *   PORTC.1             agitate led
 *   PORTC.2             spin led
 *   PORTC.4             drain valve
 *   PORTC.5             hot water valve
 *   PORTC.6             cold water valve
 */

#include <avr/io.h>

#include "Debugger.h"

/* NOTE: Custom Macros */
// TODO: None

/* NOTE: Global Variables */
// TODO: None

/* NOTE: Function prototypes */
// inits IO ports
void IO_init(void);

/* NOTE: Application implementation */
// the main loop of the function, provided to us
int main(void)
{
    initDebug();

    IO_init();

    while(1)
    {
    }
}

/* NOTE: Function implementations */
void IO_init(void)
```

```c
{
    // bottom nibble is motor and top is input switches
    DDRA = 0x0f;
    // turn on switch pullup resistors
    PORTA = 0xf0;

    // the start button
    DDRK = 0x00;
    PORTK = 0x01;

    // the led output port
    DDRC = 0xff;
    PORTC = 0x00;
}

```

```c
/*
 * FileName: main.c
 * Version: 1
 *
 * Created: 9/19/2022 8:49:04 AM
 * Author: Ethan Zeronik
 *
 * Operations: basic washing machine functions
 *
 * Hardware:
 *   Atmega2560          micro controller
 *   PORTA.4             hot water switch
 *   PORTA.5             warm water switch
 *   PORTA.6             cold water switch
 *   PORTA.7             door open switch
 *   PORTK.0             start pushbutton
 *   PORTA.0             motor out 1
 *   PORTA.1             motor out 2
 *   PORTA.2             motor out 3
 *   PORTA.3             motor out 4
 *   PORTC.0             wash done led
 *   PORTC.1             agitate led
 *   PORTC.2             spin led
 *   PORTC.4             drain valve
 *   PORTC.5             hot water valve
 *   PORTC.6             cold water valve
 */

#include <avr/io.h>

#define F_CPU 16000000UL
#include <util/delay.h>

#include "Debugger.h"

/* NOTE: Custom Macros */
#define startButton (PINK & 0x01)

#define hotButton  (PINA & 0x10)
#define warmButton (PINA & 0x20)
#define coldButton (PINA & 0x40)
#define doorSwitch (PINA & 0x80)

#define outPort (PORTC)

#define doneLed(S)    ((PORTC & ~0x01) | (S << 0))
#define agitateLed(S) ((PORTC & ~0x02) | (S << 1))
#define spinLed(S)    ((PORTC & ~0x04) | (S << 2))

#define drainValve(S) ((PORTC & ~0x10) | (S << 4))
#define hotValve(S)   ((PORTC & ~0x20) | (S << 5))
#define coldValve(S)  ((PORTC & ~0x40) | (S << 6))

/* NOTE: Global Variables */
// TODO: None

/* NOTE: Function prototypes */
```

```c
// inits IO ports
void IO_init(void);
// returns what valves need to be opened
void WASH_setValvesToInput(void);

/* NOTE: Application implementation */
// the main loop of the function, provided to us
int main(void)
{
    initDebug();

    IO_init();

    while(1)
    {
        // while we have not started the washing machine
        // and the door is open
        while(!((startButton == 0x01) && (doorSwitch == 0x00)))
        {
            // do nothing
        }

        // NOTE: fill cycle
        WASH_setValvesToInput();

        _delay_ms(4000);

        outPort = hotValve(0);
        outPort = coldValve(0);

        // NOTE: wash cycle
        outPort = agitateLed(1);

        for(size_t i = 0; i < 2; i++)
        {
            // move cw for 2 seconds
            _delay_ms(2000);

            // move ccw for 2 seconds
            _delay_ms(2000);
        }

        outPort = agitateLed(0);

        // NOTE: drain cycle
        outPort = drainValve(1);

        _delay_ms(4000);

        outPort = drainValve(0);

        // NOTE: fill again cycle
        WASH_setValvesToInput();

        _delay_ms(4000);

        outPort = hotValve(0);
        outPort = coldValve(0);

        // NOTE: rinse cycle
        // agitate 12 seconds

        // NOTE: rinse again cycle
        outPort = drainValve(1);

        // wait 15 s
        // spin for 9s

        outPort = drainValve(0);

        // NOTE: done with the wash
        outPort = doneLed(1);

        while(doorSwitch != 0x80)
        {
            // do nothing
        }

        outPort = doneLed(0);
    }
}

/* NOTE: Function implementations */
void IO_init(void)
{
    // bottom nibble is motor and top is input switches
    DDRA  = 0x0f;
    // turn on switch pullup resistors
    PORTA = 0xf0;

    // the start button
    DDRK  = 0x00;
    PORTK = 0x01;

    // the led output port
    DDRC  = 0xff;
    PORTC = 0x00;
}

void WASH_setValvesToInput(void)
{
    if(hotButton != 0x00)
    {
        // hot on, cold off
        outPort = hotValve(1);
        outPort = coldValve(0);
    }

    if(warmButton != 0x00)
    {
        // hot on, cold on
        outPort = hotValve(1);
        outPort = coldValve(1);
    }

    if(coldButton != 0x00)
    {
        // hot off, cold on
        outPort = hotValve(0);
        outPort = coldValve(1);
    }
}
```

```c
/*
 * FileName: main.c
 * Version: 1
 *
 * Created: 9/19/2022 8:50:04 AM
 * Author: Ethan Zeronik
 *
 * Operations: full featured washing machine functions
 *
 * Hardware:
 *   Atmega2560          micro controller
 *   PORTA.4             hot water switch
 *   PORTA.5             warm water switch
 *   PORTA.6             cold water switch
 *   PORTA.7             door open switch
 *   PORTK.0             start pushbutton
 *   PORTA.0             motor out 1
 *   PORTA.1             motor out 2
 *   PORTA.2             motor out 3
 *   PORTA.3             motor out 4
 *   PORTC.0             wash done led
 *   PORTC.1             agitate led
 *   PORTC.2             spin led
 *   PORTC.4             drain valve
 *   PORTC.5             hot water valve
 *   PORTC.6             cold water valve
 */

#include <avr/io.h>

#define F_CPU 16000000UL
#include <util/delay.h>

#include "Debugger.h"
#include "StepperMotor.h"

/* NOTE: Custom Macros */
#define startButton (PINK & 0x01)

#define hotButton  (PINA & 0x10)
#define warmButton (PINA & 0x20)
#define coldButton (PINA & 0x40)
#define doorSwitch (PINA & 0x80)

#define outPort (PORTC)

#define doneLed(S)    ((PORTC & ~0x01) | (S << 0))
#define agitateLed(S) ((PORTC & ~0x02) | (S << 1))
#define spinLed(S)    ((PORTC & ~0x04) | (S << 2))

#define drainValve(S) ((PORTC & ~0x10) | (S << 4))
#define hotValve(S)   ((PORTC & ~0x20) | (S << 5))
#define coldValve(S)  ((PORTC & ~0x40) | (S << 6))

/* NOTE: Global Variables */
// TODO: None

```

```c
/* NOTE: Function prototypes */
// inits IO ports
void IO_init(void);
// returns what valves need to be opened
void WASH_setValvesToInput(void);

/* NOTE: Application implementation */
// the main loop of the function, provided to us
int main(void)
{
    initDebug();

    IO_init();
    SM_init(&DDRA, &PORTA);

    while(1)
    {
        // while we have not started the washing machine
        // and the door is open
        while(!((startButton == 0x01) && (doorSwitch == 0x00)))
        {
            // do nothing
        }

        // NOTE: fill cycle
        WASH_setValvesToInput();

        _delay_ms(4000);

        outPort = hotValve(0);
        outPort = coldValve(0);

        // NOTE: wash cycle
        outPort = agitateLed(1);

        for(size_t i = 0; i < 2; i++)
        {
            // move cw for 2 seconds
            SM_moveTime((StepperMotorRunMode_t)2, true, 2000, 5);

            // move ccw for 2 seconds
            SM_moveTime((StepperMotorRunMode_t)2, false, 2000, 5);
        }

        outPort = agitateLed(0);

        // NOTE: drain cycle
        outPort = drainValve(1);

        _delay_ms(4000);

        outPort = drainValve(0);

        // NOTE: fill again cycle
        WASH_setValvesToInput();

        _delay_ms(4000);

        outPort = hotValve(0);
        outPort = coldValve(0);

        // NOTE: rinse cycle
        outPort = agitateLed(1);

        for(size_t i = 0; i < 3; i++)
        {
            // move cw for 2 seconds
            SM_moveTime((StepperMotorRunMode_t)2, true, 2000, 5);

            // move ccw for 2 seconds
            SM_moveTime((StepperMotorRunMode_t)2, false, 2000, 5);
        }

        outPort = agitateLed(0);

        // NOTE: spin cycle
        outPort = drainValve(1);
        outPort = spinLed(1);

        // spin for 9s
        SM_moveTime((StepperMotorRunMode_t)1, true, 9000, 3);

        outPort = drainValve(0);
        outPort = spinLed(0);


        // NOTE: done with the wash
        outPort = doneLed(1);

        while(doorSwitch != 0x80)
        {
            // do nothing
        }

        outPort = doneLed(0);
    }
}

/* NOTE: Function implementations */
void IO_init(void)
{
    // bottom nibble is motor and top is input switches
    DDRA  = 0x0f;
    // turn on switch pullup resistors
    PORTA = 0xf0;

    // the start button
    DDRK  = 0x00;
    PORTK = 0x01;

    // the led output port
    DDRC  = 0xff;
    PORTC = 0x00;
}
```

```c

void WASH_setValvesToInput(void)
{
    if(hotButton != 0x00)
    {
        // hot on, cold off
        outPort = hotValve(1);
        outPort = coldValve(0);
    }

    if(warmButton != 0x00)
    {
        // hot on, cold on
        outPort = hotValve(1);
        outPort = coldValve(1);
    }

    if(coldButton != 0x00)
    {
        // hot off, cold on
        outPort = hotValve(0);
        outPort = coldValve(1);
    }
}

```

```c
/*
 * FileName: StepperMotor.h
 * Version: 1
 *
 * Created: 9/14/2022 2:00 PM
 * Author: Ethan Zeronik
 *
 * Operations: header for the stepper motor submobule
 */

#ifndef StepperMotor_h_INCLUDED
#define StepperMotor_h_INCLUDED

#if defined(__cplusplus)
extern "C" {
#endif

#include <stdbool.h>
#include <stdint.h>
#include <stdio.h>

/* NOTE: Custom Types */
// typing for the stepper motor enum
typedef enum StepperMotorRunMode_t
{
    // wave step mode
    Wave = 0,
    // wave step mode
    Full = 1,
    // wave step mode
    Half = 2,
} StepperMotorRunMode_t;

/* NOTE: Function prototypes */
// inits IO for the stepper motor
// takes a pointer to the port to use, assumes botom nibble
void SM_init(volatile uint8_t * pRegister, volatile uint8_t * pPort);

// moves the motor in the given mode to the given distance
// distance is in units of rotation
void SM_move(StepperMotorRunMode_t mode, double distance);

// moves the motor in the given mode to the given position
// distance is in units of degrees
void SM_movePosition(StepperMotorRunMode_t mode, uint16_t distance);

// moves the motor in the given mode and the given direction for the given time
// 1 is CW and 0 is CCW
// both times are in ms
void SM_moveTime(StepperMotorRunMode_t mode, bool direction, double time, double stepTime);

#if defined(__cplusplus)
} /* extern "C" */
#endif

#endif // StepperMotor_h_INCLUDED
```

```c
/*
 * FileName: StepperMotor.c
 * Version: 1
 *
 * Created: 9/14/2022 2:00 PM
 * Author: Ethan Zeronik
 *
 * Operations: run the stepper motor in one of three modes
 */

/* NOTE: Includes */
#include "StepperMotor.h"

// TODO: move this
#define __DELAY_BACKWARD_COMPATIBLE__
#define F_CPU 16000000UL
#include <util/delay.h>

/* NOTE: Local declarations */
typedef struct StepperMotorModeData_t
{
    // size of the array
    size_t                arraySize;
    // pointer to the array
    uint8_t const * const pArray;
    // number of steps to take for desired rotation
    uint32_t              steps;
} StepperMotorModeData_t;

// returns the amount of steps needed for the given mode
// rotation is in radians (I think)
StepperMotorModeData_t getModeAndSteps(StepperMotorRunMode_t mode, double rotation);

/* NOTE: Global Variables */
// implementation of the wave step map
static uint8_t sWaveStepMap[4] = {
    0x01,
    0x02,
    0x04,
    0x08,
};

// implementation of the full step map
static uint8_t sFullStepMap[4] = {
    0x03,
    0x06,
    0x0c,
    0x09,
};

// implementation of the wave step map
static uint8_t sHalfStepMap[8] = {
    0x09,
    0x01,
    0x03,
    0x02,
    0x06,
    0x04,
    0x0c,
    0x08,
};

// instance pointer to the motor port
static volatile uint8_t * sMotorPort;

/* NOTE: Function implementations */
void SM_init(volatile uint8_t * pRegister, volatile uint8_t * pPort)
{
    // configure port register
    *pRegister = (*pRegister & 0xf0) | 0x0f;

    // turn on pullup resisitors on the bottom nibble
    *pPort = 0x00;

    // save the port pointer to the static var
    sMotorPort = pPort;
}

void SM_move(StepperMotorRunMode_t mode, double distance)
{
    StepperMotorModeData_t data = getModeAndSteps(mode, distance);

    for(uint32_t i = 0, j = 0; i < data.steps; i++)
    {
        *sMotorPort = data.pArray[j++];

        if(j >= data.arraySize)
        {
            j = 0;
        }

        _delay_ms(3);
    }

    *sMotorPort = 0x00;
}

void SM_movePosition(StepperMotorRunMode_t mode, uint16_t distance)
{
    SM_move(mode, ((double)distance / 360));
}

void SM_moveTime(StepperMotorRunMode_t mode, bool direction, double time, double stepTime)
{
    StepperMotorModeData_t data = getModeAndSteps(mode, 0);

    for(uint32_t i = 0, j = (direction ? data.arraySize : 0); i < (time / stepTime); i++)
    {
        *sMotorPort = data.pArray[(direction ? j-- : j++)];

        if(j >= data.arraySize || j <= 0)
        {
            j = (direction ? data.arraySize : 0);
        }

        _delay_ms(stepTime);
    }

    *sMotorPort = 0x00;
}

/* NOTE: Local function implementations */
StepperMotorModeData_t getModeAndSteps(StepperMotorRunMode_t mode, double rotation)
{
    uint8_t * pArray = NULL;
    uint8_t   size   = 0;
    uint32_t  steps  = 0;

    switch(mode)
    {
        case Wave:
        {
            pArray = sWaveStepMap;
            size   = sizeof(sWaveStepMap) / sizeof(sWaveStepMap[0]);
            steps  = (rotation * 2048);
        }
        break;
        case Full:
        {
            pArray = sFullStepMap;
            size   = sizeof(sFullStepMap) / sizeof(sFullStepMap[0]);
            steps  = (rotation * 2048);
        }
        break;
        case Half:
        {
            pArray = sHalfStepMap;
            size   = sizeof(sHalfStepMap) / sizeof(sHalfStepMap[0]);
            steps  = (rotation * 4096);
        }
        break;
        default:
            break;
    };

    return (StepperMotorModeData_t){
        .pArray    = pArray,
        .steps     = steps,
        .arraySize = size,
    };
}
```