```
//
// SegaController.h
//
// Author:
//       Jon Thysell <thysell@gmail.com>
//
// Copyright (c) 2017 Jon Thysell <http://jonthysell.com>
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in
// all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
// THE SOFTWARE.

#ifndef SegaController_h
#define SegaController_h

enum
{
    SC_CTL_ON    = 1, // The controller is connected
    SC_BTN_UP    = 2,
    SC_BTN_DOWN  = 4,
    SC_BTN_LEFT  = 8,
    SC_BTN_RIGHT = 16,
    SC_BTN_START = 32,
    SC_BTN_A     = 64,
    SC_BTN_B     = 128,
    SC_BTN_C     = 256,
    SC_BTN_X     = 512,
    SC_BTN_Y     = 1024,
    SC_BTN_Z     = 2048,
    SC_BTN_MODE  = 4096,
    SC_BTN_1     = 128, // Master System compatibility
    SC_BTN_2     = 256  // Master System compatibility
};

const byte SC_INPUT_PINS = 6;

const byte SC_CYCLES = 8;

const unsigned long SC_READ_DELAY_MS = 5; // Must be >= 3 to give 6-button controller time to reset

class SegaController {
    public:
        SegaController(byte db9_pin_7, byte db9_pin_1, byte db9_pin_2, byte db9_pin_3, byte db9_pin_4, byte db9_pin_6,
byte db9_pin_9);

        word getState();

    private:
        void readCycle(byte cycle);

        word _currentState;

        unsigned long _lastReadTime;

        boolean _sixButtonMode;

        byte _selectPin; // output select pin
        byte _inputPins[SC_INPUT_PINS];
};

#endif
```

```
//
// SegaController.cpp
//
// Author:
//       Jon Thysell <thysell@gmail.com>
//
// Copyright (c) 2017 Jon Thysell <http://jonthysell.com>
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in
// all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
// THE SOFTWARE.

#include "Arduino.h"
#include "SegaController.h"

SegaController::SegaController(byte db9_pin_7, byte db9_pin_1, byte db9_pin_2, byte db9_pin_3, byte db9_pin_4, byte
db9_pin_6, byte db9_pin_9)
{
    // Set pins
    _selectPin = db9_pin_7;

    _inputPins[0] = db9_pin_1;
    _inputPins[1] = db9_pin_2;
    _inputPins[2] = db9_pin_3;
    _inputPins[3] = db9_pin_4;
    _inputPins[4] = db9_pin_6;
    _inputPins[5] = db9_pin_9;

    // Setup output pin
    pinMode(_selectPin, OUTPUT);
    digitalWrite(_selectPin, HIGH);

    // Setup input pins
    for (byte i = 0; i < SC_INPUT_PINS; i++)
    {
        pinMode(_inputPins[i], INPUT_PULLUP);
    }

    _currentState = 0;
    _sixButtonMode = false;
    _lastReadTime = millis();
}

word SegaController::getState()
{
    if (max(millis() - _lastReadTime, 0) < SC_READ_DELAY_MS)
    {
        // Not enough time has elapsed, return previously read state
        return _currentState;
    }

    noInterrupts();

    // Clear current state
    _currentState = 0;

    for (byte cycle = 0; cycle < SC_CYCLES; cycle++)
    {
        readCycle(cycle);
    }

    // When a controller disconnects, revert to three-button polling
    if (!(_currentState & SC_CTL_ON))
    {
        _sixButtonMode = false;
    }

    interrupts();

    _lastReadTime = millis();

    return _currentState;
}

void SegaController::readCycle(byte cycle)
{
    // Set the select pin low/high
    digitalWrite(_selectPin, cycle % 2);

    // Read flags
    switch (cycle)
    {
        case 2:
            // Check that a controller is connected
            _currentState |= (digitalRead(_inputPins[2]) == LOW && digitalRead(_inputPins[3]) == LOW) * SC_CTL_ON;
```

```
            // Check controller is connected before reading A/Start to prevent bad reads when inserting/removing cable
            if (_currentState & SC_CTL_ON)
            {
                // Read input pins for A, Start
                if (digitalRead(_inputPins[4]) == LOW) { _currentState |= SC_BTN_A; }
                if (digitalRead(_inputPins[5]) == LOW) { _currentState |= SC_BTN_START; }
            }
            break;
        case 3:
            // Read input pins for Up, Down, Left, Right, B, C
            if (digitalRead(_inputPins[0]) == LOW) { _currentState |= SC_BTN_UP; }
            if (digitalRead(_inputPins[1]) == LOW) { _currentState |= SC_BTN_DOWN; }
            if (digitalRead(_inputPins[2]) == LOW) { _currentState |= SC_BTN_LEFT; }
            if (digitalRead(_inputPins[3]) == LOW) { _currentState |= SC_BTN_RIGHT; }
            if (digitalRead(_inputPins[4]) == LOW) { _currentState |= SC_BTN_B; }
            if (digitalRead(_inputPins[5]) == LOW) { _currentState |= SC_BTN_C; }
            break;
        case 4:
            _sixButtonMode = (digitalRead(_inputPins[0]) == LOW && digitalRead(_inputPins[1]) == LOW);
            break;
        case 5:
            if (_sixButtonMode)
            {
                // Read input pins for X, Y, Z, Mode
                if (digitalRead(_inputPins[0]) == LOW) { _currentState |= SC_BTN_Z; }
                if (digitalRead(_inputPins[1]) == LOW) { _currentState |= SC_BTN_Y; }
                if (digitalRead(_inputPins[2]) == LOW) { _currentState |= SC_BTN_X; }
                if (digitalRead(_inputPins[3]) == LOW) { _currentState |= SC_BTN_MODE; }
            }
            break;
    }
}
```

```
//
// SegaControllerSerialReader.ino
//
// Author:
//       Jon Thysell <thysell@gmail.com>
//
// Copyright (c) 2017 Jon Thysell <http://jonthysell.com>
//
// Permission is hereby granted, free of charge, to any person obtaining a copy
// of this software and associated documentation files (the "Software"), to deal
// in the Software without restriction, including without limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
// copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in
// all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
// FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
// THE SOFTWARE.

#include <SegaController.h>

// Controller DB9 pins (looking face-on to the end of the plug):
//
// 5 4 3 2 1
//  9 8 7 6
//
// Connect pin 5 to +5V and pin 8 to GND
// Connect the remaining pins to digital I/O pins (see below)

// Specify the Arduino pins that are connected to
// DB9 Pin 7, DB9 Pin 1, DB9 Pin 2, DB9 Pin 3, DB9 Pin 4, DB9 Pin 6, DB9 Pin 9
SegaController controller(8, 2, 3, 4, 5, 6, 7);

// Controller states
word currentState = 0;
word lastState = 0;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    currentState = controller.getState();
    sendState();
}

void sendState()
{
    // Only report controller state if it's changed
    if (currentState != lastState)
    {
        Serial.print((currentState & SC_CTL_ON)    ? "+" : "-");
        Serial.print((currentState & SC_BTN_UP)    ? "U" : "0");
        Serial.print((currentState & SC_BTN_DOWN)  ? "D" : "0");
        Serial.print((currentState & SC_BTN_LEFT)  ? "L" : "0");
        Serial.print((currentState & SC_BTN_RIGHT) ? "R" : "0");
        Serial.print((currentState & SC_BTN_START) ? "S" : "0");
        Serial.print((currentState & SC_BTN_A)     ? "A" : "0");
        Serial.print((currentState & SC_BTN_B)     ? "B" : "0");
        Serial.print((currentState & SC_BTN_C)     ? "C" : "0");
        Serial.print((currentState & SC_BTN_X)     ? "X" : "0");
        Serial.print((currentState & SC_BTN_Y)     ? "Y" : "0");
        Serial.print((currentState & SC_BTN_Z)     ? "Z" : "0");
        Serial.print((currentState & SC_BTN_MODE)  ? "M" : "0");

        Serial.print("\n");
        lastState = currentState;
    }
}
```