DeanIsMe / SevSeg

Seven segment display controller library for Arduino

19 commits 1 branch 0 releases 2 contributors

Branch: master New pull request New file Upload files Find file HTTPS https://github.com/DeanIs Download ZIP

DeanIsMe Minor changes Latest commit ba10676 on 16 Jan

examples	Update testWholeDisplay.ino	5 months ago
LICENSE.txt	Initial upload of all files	a year ago
README.md	Minor changes	2 months ago
SevSeg.cpp	added new example & a few comments	5 months ago
SevSeg.h	Update for compatibility	a year ago
keywords.txt	Corrected structure of files	a year ago

README.md

SevSeg

This library turns your Arduino into a seven segment display controller! Use it to easily display numbers on your seven segment display without any additional controllers.

It supports common cathode and common anode displays, and the use of switching transistors. Displays with 1 to 9 digits can be used, and decimal places are supported. Characters and strings are not supported.

Download it from GitHub.

Direct any questions or suggestions to deanreading@hotmail.com. If I have the time, I'm happy to help you get things working.

Previous Versions Note

This version is not compatible with previous versions of the SevSeg library, which have been available since 2012. You can download the old version for compatibility with previously written programs.

Thanks to Mark Chambers and Nathan Seidle for code used in updates.

HARDWARE

Seven Segment Display Pins

Your display should have:
Digit Pins - One for each digit. These are the 'common pins'. They will be cathodes (negative pins) for common cathode displays, or anodes (positive pins) for common anode displays.
8 Segment Pins - One for each of the seven segments plus the decimal point.

Arduino Connections

All digit pins and segment pins can be connected to any of the Arduino's digital or analog pins; just make sure you take note of your connections!

Current-limiting Resistors

Don't forget that the display uses LEDs, so you should use current-limiting resistors in series with the *digit pins*. 330 ohms is a safe value if you're unsure. If you use current-limiting resistors on the *segment pins* instead, then open up the SevSeg.h file and set RESISTORS_ON_SEGMENTS to 1 for optimal brightness.

Hardware Configuration

You have to specify your hardware configuration to the library as the first argument in sevseg.begin. The options are detailed below.

Simple, Low Power Displays

These displays are powered directly through the Arduino output pins.

- **COMMON_CATHODE** - For common cathode displays without switches. These displays require a low voltage at the digit pin to illuminate the digit.
- **COMMON_ANODE** - For common anode displays without switches. These displays require a high voltage at the digit pin to illuminate the digit.

Displays with Switches

Some displays (mostly bigger ones) use switching transistors, but most people won't have to worry about the configurations below.

- **N_TRANSISTORS** - If you use N-type transistors to sink current (or any other active-high, low-side switches).
- **P_TRANSISTORS** - If you use P-type transistors to supply current (or any other active-low, high-side switches).
- **NP_COMMMON_CATHODE** - If your setup uses N-type AND P-type transistors with a common cathode display.
- **NP_COMMMON_ANODE** - If your setup uses N-type AND P-type transistors with a common anode display.
  Note that use of active-high, high-side switches will have no impact on the configuration chosen. There are usually called high-side switches.

Example Pinout

In the below pinout, digits are numbered 1, 2, 3, 4. Segments are numbered A through G plus Decimal Point (DP), according to this picture. Pins are ordered as looking at the front of the display.

Cheap, 4-digit, 12-pin display from Ebay (labelled HS410561k-32 on bottom edge):
4-digit common anode display, with 2 rows of 6 pins.

```
Top Row:    1 A F  2 3 B
Bottom Row: E D DP C G 4
```

SOFTWARE

To install, copy the SevSeg folder into your arduino sketchbook-libraries folder. More detailed instructions are here.

Setting Up

```
#include "SevSeg.h"
SevSeg sevseg; //Instantiate a seven segment object

void setup() {
   byte numDigits = 4;
   byte digitPins[] = {2, 3, 4, 5};
   byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13};
   sevseg.begin(COMMON_ANODE, numDigits, digitPins, segmentPins);
   ...
```

digitPins is an array that stores the arduino pin numbers that the digits are connected to. Order them from left to right.
segmentPins is an array that stores the arduino pin numbers that the segments are connected to. Order them from segment a to g , then the decimal place. If you wish to use more than 8 digits, increase MAXNUMDIGITS in SevSeg.h.

Setting the Number

```
sevseg.setNumber(3141,3); // Displays '3.141'
```

The first argument is the number to display. The second argument indicates where the decimal place should be, counted from the least significant digit. E.g. to display an integer, the second argument is 0.
Floats are supported. In this case, the second argument indicated how many decimal places of precision you want to display. E.g:

```
sevseg.setNumber(3.141f,3); //Displays '3.141'
```

Out of range numbers show up as ------.

Displaying the Number

```
sevseg.refreshDisplay();
```

Your program must run the refreshDisplay() function repeatedly to display the number.

Set the Brightness

```
sevseg.setBrightness(90);
```

The brightness can be adjusted using a value between 0 and 100.
Note that a 0 does not correspond to no brightness. If you wish for the display to be any dimmer than 0, run `sevseg.refreshDisplay();` less frequently. If your display has noticeable flickering, reducing the brightness level may correct it.