```
/*
  14 Segment Display Revolving Message Sketch

  By Charlie Cook a.k.a. C-Squared
  Last Edited July 31st, 2019

  This sketch allows for the storage and presentation of alphanumeric messages
  across four digits of 14 segments each. The hardware needed besides the
  displays, which themselves are LPT-3784 common-cathode modules from Lite On,
  is just one (1) CD4051 or CD4052 analog multiplexer, and two (2) CD4094
  SIPO shift registers.

  The shift registers should be hooked up head-to-tail (i.e. Q_s of the first chip
  should connect to DATA of the second chip; also tie OUT.EN. high), and the first
  14 bits should be connected to the 14 seg. displays on pins A thru P
  (the letters I and O are not used to avoid ambiguity). The last 2 bits
  of the shift registers should be fed to the A & B addressing pins of the
  multiplexer. The lower four connection pins of the mux should connect to the
  cathodes of the individual digits of the display.

  Details on which cathodes are which and how the pins map from their letter names
  to numbers can be found in the datasheet for the LPT-3784. It should be noted
  that this sheet neglects to inform that the numbering of the pins is like that
  of DIP logic chips, where pin 1 is in the lower left corner, then the number
  increases counter-clockwise to pin 18 in the upper left corner.

  Once all connections are made, The DATA, CLOCK, and STROBE inputs of the 4094s
  should connect to digital pins 2, 3, and 4 respectively. After that, simply use
  the Arduino IDE's serial monitor or something like moserial to set the message.
  Don't worry if when you boot it up the first time and see garbage, it won't
  impede writes from a serial link.

  One last note, my fontface, contained in the unsigned int arrays, only handles
  alphanumeric characters at this time. All non-A.N. characters will be rendered
  blank on the display. Also, some of the lowercase letters can look weird, so I
  recommend sticking to capitals.
*/
```

```
#include <EEPROM.h>
#define DATAPIN 2
#define CLOCKPIN 3
#define STROBEPIN 4
#define NEXTDIGIT 0x40
#define DIGITSHIFT 6
#define HIGHSHIFT 8
#define LOWMASK 0x00FF
#define DIGITDIFF 0x30
#define LOWERCASEDIFF 0x61
#define UPPERCASEDIFF 0x41

const unsigned int segTransNum[10] = {
  0x08BF,
  0x0086,
  0x1099,
  0x010F,
  0x1126,
  0x210D,
  0x1239,
  0x0881,
  0x113F,
  0x210F
};
```

```
const unsigned int segTransUpp[26] = {
  0x0586,
  0x054F,
  0x0039,
  0x044F,
  0x1039,
  0x1131,
  0x013D,
  0x1136,
  0x0449,
  0x0851,
  0x12B0,
  0x0038,
  0x20B6,
  0x2236,
  0x003F,
  0x10B1,
  0x023F,
  0x12B1,
  0x112D,
  0x0441,
  0x003E,
  0x2206,
  0x0A36,
  0x2A80,
  0x2480,
  0x0889
};
```

```
const unsigned int segTransLow[26] = {
  0x1A08,
  0x1238,
  0x1118,
  0x090E,
  0x0382,
  0x1580,
  0x210F,
  0x1230,
  0x0104,
  0x0850,
  0x06C0,
  0x0440,
  0x1514,
  0x1210,
  0x111C,
  0x3030,
  0x2107,
  0x1110,
  0x0308,
  0x1540,
  0x001C,
  0x0204,
  0x0A14,
  0x2A80,
  0x2880,
  0x1808
};
```

```
byte getDigit(byte d, unsigned int n) {
  unsigned int powTen = 1;
  byte k;
  for (k = 1; k < d; k++) {
    powTen *= 10;
  }
  return (byte) ((n / powTen) % 10);
}
```

```
unsigned int translate(char c) {
  if (isDigit(c)) {
    return segTransNum[c - DIGITDIFF];
  } else if (isLowerCase(c)) {
    return segTransLow[c - LOWERCASEDIFF];
  } else if (isUpperCase(c)) {
    return segTransUpp[c - UPPERCASEDIFF];
  } else {
    return 0;
  }
}
```

```
byte i, j, high, low, loops, temp;
char c;
unsigned int t;
String message;
byte mesgLen;
void setup() {
  pinMode(DATAPIN, OUTPUT);
  pinMode(CLOCKPIN, OUTPUT);
  pinMode(STROBEPIN, OUTPUT);

  digitalWrite(DATAPIN, LOW);
  digitalWrite(CLOCKPIN, LOW);
  digitalWrite(STROBEPIN, HIGH);

  loops = 0;
  j = 0;
  message = "";

  temp = EEPROM.read(0);
  if (temp > 0 && temp <= 128) {
    i = 1;
    do {
      c = EEPROM.read(i);
      message.concat(c);
      i++;
    } while (i <= temp && c != 0 && c != 0xFF);
  } else {
    message = "PLEASE SET THIS MESSAGE VIA A SERIAL LINK   ";
  }
  mesgLen = message.length();

  Serial.begin(9600);
  Serial.println("14 Segment Display with Programmable Message");
  Serial.println("Stored message length: " + String(mesgLen));
  Serial.println("Send a new message if you like (limit of 128 chars)");
  Serial.print("$ ");
}
```

```
void loop() {
  for (i = 0; i < 4; i++) {
    t = translate(message[(j + 3 - i) % mesgLen]);
    high = (byte) (t >> HIGHSHIFT);
    low = (byte) (t & LOWMASK);
    high += (i << DIGITSHIFT);
    digitalWrite(STROBEPIN, LOW);
    shiftOut(DATAPIN, CLOCKPIN, MSBFIRST, high);
    shiftOut(DATAPIN, CLOCKPIN, MSBFIRST, low);
    digitalWrite(STROBEPIN, HIGH);
    delay(5);
  }

  loops++;
  if (loops == 10) {
    loops = 0;
    j++;
    j %= mesgLen;
  }

  if (Serial.available() > 0) {
    message = "";
    while (Serial.available() > 0) {
      c = Serial.read();
      message.concat(c);
      delay(1);
    }
    if (message.length() > 0) {
      mesgLen = message.length();
      Serial.println("Message recieved! Length: " + String(mesgLen));
      Serial.println("Message: " + message);
      EEPROM.write(0, mesgLen);
      for (i = 0; i < mesgLen; i++) {
        EEPROM.write(i + 1, message.charAt(i));
      }
      EEPROM.write(i + 1, 0);
      Serial.println("Length Confirmation (Read from EEPROM): " + String(EEPROM.read(0)));
      Serial.print("$ ");
      j = 0;
      loops = 0;
    }
  }
}
```