Tutorial – Arduino and the MAX7219 LED Display Driver IC

Use the Maxim MAX7219 LED display driver with Arduino in Chapter 56 of our Arduino Tutorials. The first chapter is here, the complete series is detailed here.

Introduction

Sooner or later Arduino enthusiasts and beginners alike will come across the MAX7219 IC. And for good reason, it's a simple and somewhat inexpensive method of controlling 64 LEDs in either matrix or numeric display form. Furthermore they can be chained together to control two or more units for even more LEDs. Overall – they're a lot of fun and can also be quite useful, so let's get started.

Here's an example of a MAX7219 and another IC which is a functional equivalent, the AS1107 from Austria Microsystems. You might not see the AS1107 around much, but it can be cheaper – so don't be afraid to use that instead:

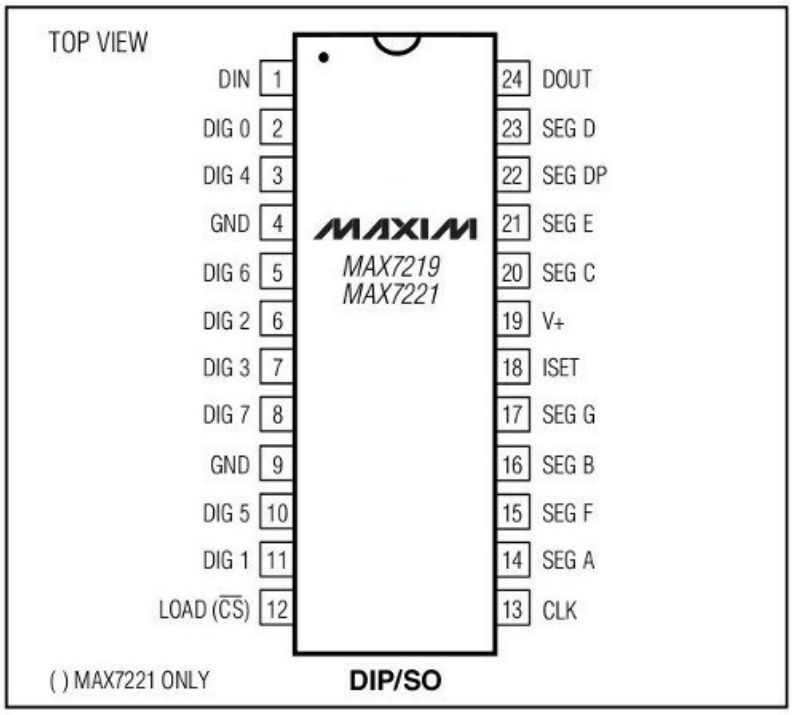
At first glance you may think that it takes a lot of real estate, but it saves some as well. As mentioned earlier, the MAX7219 can completely control 64 individual LEDs – including maintaining equal brightness, and allowing you to adjust the brightness of the LEDs either with hardware or software (or both). It can refresh the LEDs at around 800 Hz, so no more flickering, uneven LED displays.

You can even switch the display off for power saving mode, and still send it data while it is off. And another good thing – when powered up, it keeps the LEDs off, so no wacky displays for the first seconds of operation. For more technical information, here is the data sheet: MAX7219.pdf. Now to put it to work for us – we'll demonstrate using one or more 8 x 8 LED matrix displays, as well as 8 digits of 7-segment LED numbers.

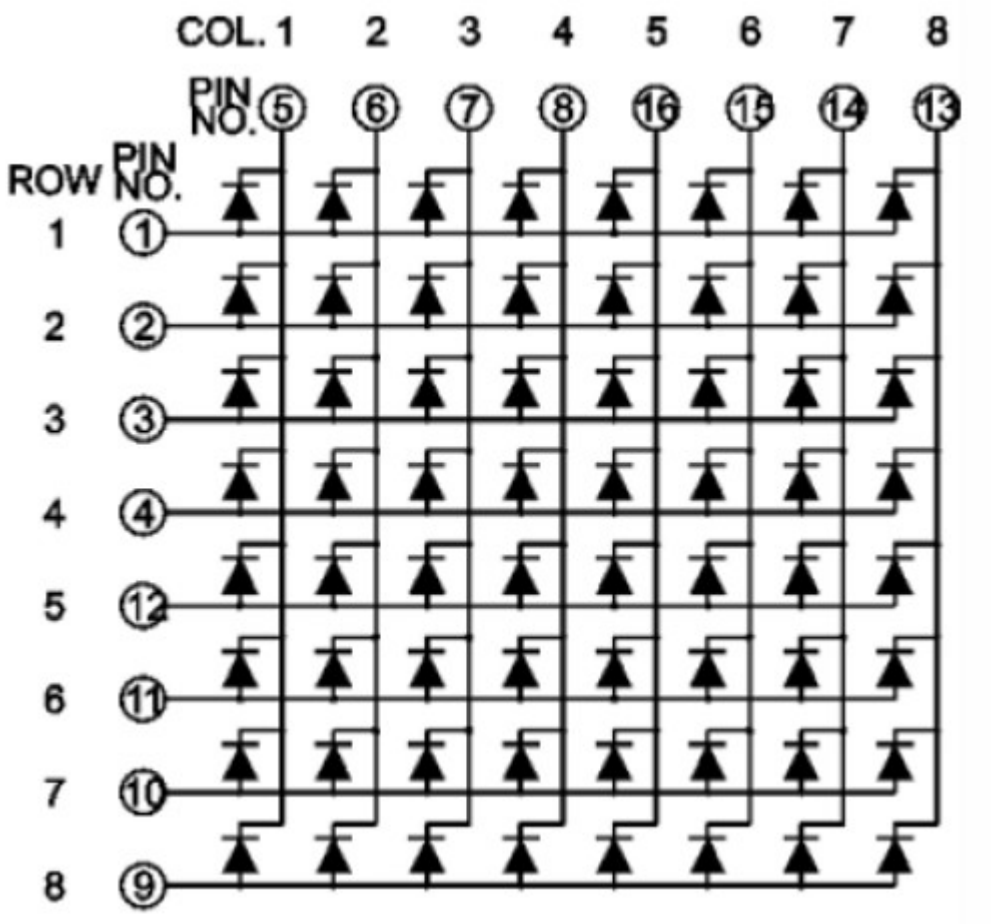
Before continuing, download and install the LedControl Arduino library as it is essential for using the MAX7219.

Controlling LED matrix displays with the MAX7219

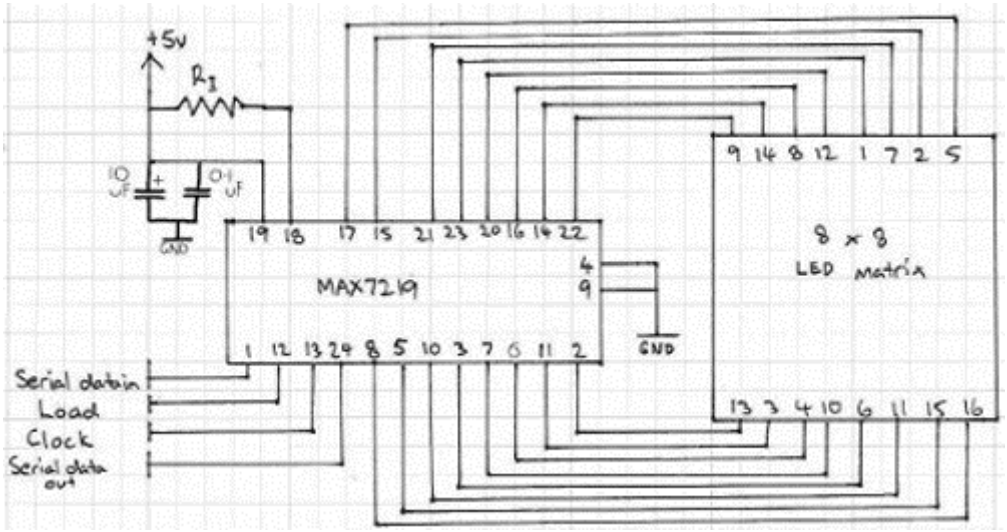
First of all, let's examine the hardware side of things. Here is the pinout diagram for the MAX7219:



The MAX7219 drives eight LEDs at a time, and by rapidly switching banks of eight your eyes don't see the changes. Wiring up a matrix is very simple – if you have a common matrix with the following schematic:



connect the MAX7219 pins labelled DP, A~F to the row pins respectively, and the MAX7219 pins labelled DIG0~7 to the column pins respectively. A total example circuit with the above matrix is as follows:



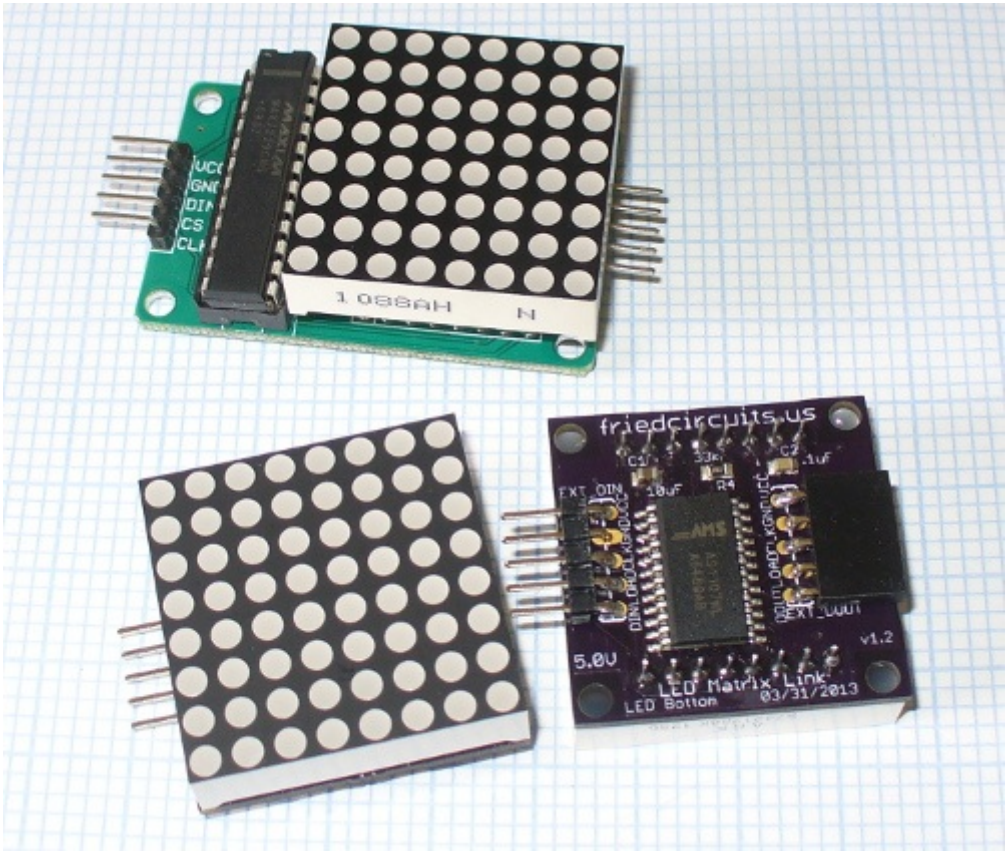
The circuit is quite straight forward, except we have a resistor between 5V and MAX7219 pin 18. The MAX7219 is a constant-current LED driver, and the value of the resistor is used to set the current flow to the LEDs. Have a look at table eleven on page eleven of the data sheet:

Table 11. R_{SET} vs. Segment Current and LED Forward Voltage

I_{SEG} (mA)	V_{LED} (V)				
	1.5	2.0	2.5	3.0	3.5
40	12.2	11.8	11.0	10.6	9.69
30	17.8	17.1	15.8	15.0	14.0
20	29.8	28.0	25.9	24.5	22.6
10	66.7	63.7	59.3	55.4	51.2

You'll need to know the voltage and forward current for your LED matrix or numeric display, then match the value on the table. E.g. if you have a 2V 20 mA LED, your resistor value will be 28kΩ (the values are in kΩ). Finally, the MAX7219 serial in, load and clock pins will go to Arduino digital pins which are specified in the sketch. We'll get to that in the moment, but before that let's return to the matrix modules.

In the last few months there has been a proliferation of inexpensive kits that contain a MAX7219 or equivalent, and an LED matrix. These are great for experimenting with and can save you a lot of work – some examples of which are shown below:



At the top is an example from tronixlabs.com, and the pair on the bottom are the units from a recent kit review. We'll use these for our demonstrations as well.

Now for the sketch. You need the following two lines at the beginning of the sketch:

```
#include "LedControl.h"
LedControl lc=LedControl(12,11,10,1);
```

The first pulls in the library, and the second line sets up an instance to control. The four parameters are as follows:

1. the digital pin connected to pin 1 of the MAX7219 ("data in")
2. the digital pin connected to pin 13 of the MAX7219 ("CLK or clock")
3. the digital pin connected to pin 12 of the MAX7219 ("LOAD")
4. The number of MAX7219s connected.

If you have more than one MAX7219, connect the DOUT ("data out") pin of the first MAX7219 to pin 1 of the second, and so on. However the CLK and LOAD pins are all connected in parallel and then back to the Arduino.

Next, two more vital functions that you'd normally put in void setup():

```
lc.shutdown(0,false);
lc.setIntensity(0,8);
```

The first line above turns the LEDs connected to the MAX7219 on. If you set TRUE, you can send data to the MAX7219 but the LEDs will stay off. The second line adjusts the brightness of the LEDs in sixteen stages. For both of those functions (and all others from the LedControl) the first parameter is the number

of the MAX7219 connected. If you have one, the parameter is zero… for two MAX7219s, it's 1 and so on.

Finally, to turn an individual LED in the matrix on or off, use:

```
lc.setLed(0,col,row,true);
```

which turns on an LED positioned at col, row connected to MAX7219 #1. Change TRUE to FALSE to turn it off. These functions are demonstrated in the following sketch:

```
#include "LedControl.h" //  need the library
LedControl lc=LedControl(12,11,10,1); //

// pin 12 is connected to the MAX7219 pin 1
// pin 11 is connected to the CLK pin 13
// pin 10 is connected to LOAD pin 12
// 1 as we are only using 1 MAX7219

void setup()
{
  // the zero refers to the MAX7219 number, it is zero for 1 chip
  lc.shutdown(0,false);// turn off power saving, enables display
  lc.setIntensity(0,8);// sets brightness (0~15 possible values)
  lc.clearDisplay(0);// clear screen
}
void loop()
{
  for (int row=0; row<8; row++)
  {
    for (int col=0; col<8; col++)
    {
      lc.setLed(0,col,row,true); // turns on LED at col, row
      delay(25);
    }
  }

  for (int row=0; row<8; row++)
  {
    for (int col=0; col<8; col++)
    {
      lc.setLed(0,col,row,false); // turns off LED at col, row
      delay(25);
    }
  }
}
```

And a quick video of the results:

How about controlling two MAX7219s? Or more? The hardware modifications are easy – connect the serial data out pin from your first MAX7219 to the data in pin on the second (and so on), and the LOAD and CLOCK pins from the first MAX7219 connect to the second (and so on). You will of course still need the 5V, GND, resistor, capacitors etc. for the second and subsequent MAX7219.

You will also need to make a few changes in your sketch. The first is to tell it how many MAX7219s you're using in the following line:

```
LedControl lc=LedControl(12,11,10,X);
```

by replacing X with the quantity. Then whenever you're using a MAX7219 function, replace the (previously used) zero with the number of the MAX7219 you wish to address. They are numbered from zero upwards, with the MAX7219 directly connected to the Arduino as unit zero, then one etc. To demonstrate this, we replicate the previous example but with two MAX7219s:

```
#include "LedControl.h" //  need the library
LedControl lc=LedControl(12,11,10,2); //

// pin 12 is connected to the MAX7219 pin 1
// pin 11 is connected to the CLK pin 13
// pin 10 is connected to LOAD pin 12
// 1 as we are only using 1 MAX7219

void setup()
{
  lc.shutdown(0,false);// turn off power saving, enables display
  lc.setIntensity(0,8);// sets brightness (0~15 possible values)
  lc.clearDisplay(0);// clear screen

  lc.shutdown(1,false);// turn off power saving, enables display
  lc.setIntensity(1,8);// sets brightness (0~15 possible values)
  lc.clearDisplay(1);// clear screen
}

void loop()
{
  for (int row=0; row<8; row++)
  {
    for (int col=0; col<8; col++)
    {
      lc.setLed(0,col,row,true); // turns on LED at col, row
      lc.setLed(1,col,row,false); // turns on LED at col, row
      delay(25);
    }
  }

  for (int row=0; row<8; row++)
  {
    for (int col=0; col<8; col++)
    {
      lc.setLed(0,col,row,false); // turns off LED at col, row
      lc.setLed(1,col,row,true); // turns on LED at col, row
      delay(25);
    }
  }
}
```

And again, a quick demonstration:

Another fun use of the MAX7219 and LED matrices is to display scrolling text. For the case of simplicity we'll use the LedControl library and the two LED matrix modules from the previous examples.

First our example sketch – it is quite long however most of this is due to defining the characters for each letter of the alphabet and so on. We'll explain it at the other end!

```
// based on an orginal sketch by Arduino forum member "danigom"
// http://forum.arduino.cc/index.php?action=profile;u=188950
```

```
#include <avr/pgmspace.h>
#include <LedControl.h>

const int numDevices = 2;      // number of MAX7219s used
const long scrollDelay = 75;   // adjust scrolling speed

unsigned long bufferLong [14] = {0};

LedControl lc=LedControl(12,11,10,numDevices);

prog_uchar scrollText[] PROGMEM ={
    "  THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG 1234567890 the quick brown fox jumped over
the lazy dog   \0"};

void setup(){
    for (int x=0; x<numDevices; x++){
        lc.shutdown(x,false);       //The MAX72XX is in power-saving mode on startup
        lc.setIntensity(x,8);       // Set the brightness to default value
        lc.clearDisplay(x);         // and clear the display
    }
}

void loop(){
    scrollMessage(scrollText);
    scrollFont();
}

////////////////////////////////////////////////////////////////////////////////////////
//////////////////////

prog_uchar font5x7 [] PROGMEM = {      //Numeric Font Matrix (Arranged as 7x font data + 1x
kerning data)
    B00000000, //Space (Char 0x20)
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    6,

    B10000000, //!
    B10000000,
    B10000000,
    B10000000,
    B00000000,
    B00000000,
    B10000000,
    2,

    B10100000, //"
    B10100000,
    B10100000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    4,

    B01010000, //#
    B01010000,
    B11111000,
    B01010000,
    B11111000,
    B01010000,
    B01010000,
    6,

    B00100000, //$
    B01111000,
    B10100000,
    B01110000,
```

```
    B00101000,
    B11110000,
    B00100000,
    6,

    B11000000, //%
    B11001000,
    B00010000,
    B00100000,
    B01000000,
    B10011000,
    B00011000,
    6,

    B01100000, //&
    B10010000,
    B10100000,
    B01000000,
    B10101000,
    B10010000,
    B01101000,
    6,

    B11000000, //'
    B01000000,
    B10000000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    3,

    B00100000, //(
    B01000000,
    B10000000,
    B10000000,
    B10000000,
    B01000000,
    B00100000,
    4,

    B10000000, //)
    B01000000,
    B00100000,
    B00100000,
    B00100000,
    B01000000,
    B10000000,
    4,

    B00000000, //*
    B00100000,
    B10101000,
    B01110000,
    B10101000,
    B00100000,
    B00000000,
    6,

    B00000000, //+
    B00100000,
    B00100000,
    B11111000,
    B00100000,
    B00100000,
    B00000000,
    6,

    B00000000, //,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    B11000000,
```

```
    B01000000,
    B10000000,
    3,

    B00000000, //-
    B00000000,
    B11111000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    6,

    B00000000, //.
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    B11000000,
    B11000000,
    3,

    B00000000, ///
    B00001000,
    B00010000,
    B00100000,
    B01000000,
    B10000000,
    B00000000,
    6,

    B01110000, //0
    B10001000,
    B10011000,
    B10101000,
    B11001000,
    B10001000,
    B01110000,
    6,

    B01000000, //1
    B11000000,
    B01000000,
    B01000000,
    B01000000,
    B01000000,
    B11100000,
    4,

    B01110000, //2
    B10001000,
    B00001000,
    B00010000,
    B00100000,
    B01000000,
    B11111000,
    6,

    B11111000, //3
    B00010000,
    B00100000,
    B00010000,
    B00001000,
    B10001000,
    B01110000,
    6,

    B00010000, //4
    B00110000,
    B01010000,
    B10010000,
    B11111000,
    B00010000,
    B00010000,
```

```
    6,

    B11111000, //5
    B10000000,
    B11110000,
    B00001000,
    B00001000,
    B10001000,
    B01110000,
    6,

    B00110000, //6
    B01000000,
    B10000000,
    B11110000,
    B10001000,
    B10001000,
    B01110000,
    6,

    B11111000, //7
    B10001000,
    B00001000,
    B00010000,
    B00100000,
    B00100000,
    B00100000,
    6,

    B01110000, //8
    B10001000,
    B10001000,
    B01110000,
    B10001000,
    B10001000,
    B01110000,
    6,

    B01110000, //9
    B10001000,
    B10001000,
    B01111000,
    B00001000,
    B00010000,
    B01100000,
    6,

    B00000000, //:
    B11000000,
    B11000000,
    B00000000,
    B11000000,
    B11000000,
    B00000000,
    3,

    B00000000, //;
    B11000000,
    B11000000,
    B00000000,
    B11000000,
    B01000000,
    B10000000,
    3,

    B00010000, //<
    B00100000,
    B01000000,
    B10000000,
    B01000000,
    B00100000,
    B00010000,
    5,
```

```
    B00000000, //=
    B00000000,
    B11111000,
    B00000000,
    B11111000,
    B00000000,
    B00000000,
    6,

    B10000000, //>
    B01000000,
    B00100000,
    B00010000,
    B00100000,
    B01000000,
    B10000000,
    5,

    B01110000, //?
    B10001000,
    B00001000,
    B00010000,
    B00100000,
    B00000000,
    B00100000,
    6,

    B01110000, //@
    B10001000,
    B00001000,
    B01101000,
    B10101000,
    B10101000,
    B01110000,
    6,

    B01110000, //A
    B10001000,
    B10001000,
    B10001000,
    B11111000,
    B10001000,
    B10001000,
    6,

    B11110000, //B
    B10001000,
    B10001000,
    B11110000,
    B10001000,
    B10001000,
    B11110000,
    6,

    B01110000, //C
    B10001000,
    B10000000,
    B10000000,
    B10000000,
    B10001000,
    B01110000,
    6,

    B11100000, //D
    B10010000,
    B10001000,
    B10001000,
    B10001000,
    B10010000,
    B11100000,
    6,

    B11111000, //E
    B10000000,
```

```
    B10000000,
    B11110000,
    B10000000,
    B10000000,
    B11111000,
    6,

    B11111000, //F
    B10000000,
    B10000000,
    B11110000,
    B10000000,
    B10000000,
    B10000000,
    6,

    B01110000, //G
    B10001000,
    B10000000,
    B10111000,
    B10001000,
    B10001000,
    B01111000,
    6,

    B10001000, //H
    B10001000,
    B10001000,
    B11111000,
    B10001000,
    B10001000,
    B10001000,
    6,

    B11100000, //I
    B01000000,
    B01000000,
    B01000000,
    B01000000,
    B01000000,
    B11100000,
    4,

    B00111000, //J
    B00010000,
    B00010000,
    B00010000,
    B00010000,
    B10010000,
    B01100000,
    6,

    B10001000, //K
    B10010000,
    B10100000,
    B11000000,
    B10100000,
    B10010000,
    B10001000,
    6,

    B10000000, //L
    B10000000,
    B10000000,
    B10000000,
    B10000000,
    B10000000,
    B11111000,
    6,

    B10001000, //M
    B11011000,
    B10101000,
    B10101000,
```

```
    B10001000,
    B10001000,
    B10001000,
    6,

    B10001000, //N
    B10001000,
    B11001000,
    B10101000,
    B10011000,
    B10001000,
    B10001000,
    6,

    B01110000, //O
    B10001000,
    B10001000,
    B10001000,
    B10001000,
    B10001000,
    B01110000,
    6,

    B11110000, //P
    B10001000,
    B10001000,
    B11110000,
    B10000000,
    B10000000,
    B10000000,
    6,

    B01110000, //Q
    B10001000,
    B10001000,
    B10001000,
    B10101000,
    B10010000,
    B01101000,
    6,

    B11110000, //R
    B10001000,
    B10001000,
    B11110000,
    B10100000,
    B10010000,
    B10001000,
    6,

    B01111000, //S
    B10000000,
    B10000000,
    B01110000,
    B00001000,
    B00001000,
    B11110000,
    6,

    B11111000, //T
    B00100000,
    B00100000,
    B00100000,
    B00100000,
    B00100000,
    B00100000,
    6,

    B10001000, //U
    B10001000,
    B10001000,
    B10001000,
    B10001000,
    B10001000,
```

```
    B01110000,
    6,

    B10001000, //V
    B10001000,
    B10001000,
    B10001000,
    B10001000,
    B01010000,
    B00100000,
    6,

    B10001000, //W
    B10001000,
    B10001000,
    B10101000,
    B10101000,
    B10101000,
    B01010000,
    6,

    B10001000, //X
    B10001000,
    B01010000,
    B00100000,
    B01010000,
    B10001000,
    B10001000,
    6,

    B10001000, //Y
    B10001000,
    B10001000,
    B01010000,
    B00100000,
    B00100000,
    B00100000,
    6,

    B11111000, //Z
    B00001000,
    B00010000,
    B00100000,
    B01000000,
    B10000000,
    B11111000,
    6,

    B11100000, //[
    B10000000,
    B10000000,
    B10000000,
    B10000000,
    B10000000,
    B11100000,
    4,

    B00000000, //(Backward Slash)
    B10000000,
    B01000000,
    B00100000,
    B00010000,
    B00001000,
    B00000000,
    6,

    B11100000, //]
    B00100000,
    B00100000,
    B00100000,
    B00100000,
    B00100000,
    B11100000,
    4,
```

```
    B00100000, //^
    B01010000,
    B10001000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    6,

    B00000000, //_
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    B11111000,
    6,

    B10000000, //`
    B01000000,
    B00100000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    4,

    B00000000, //a
    B00000000,
    B01110000,
    B00001000,
    B01111000,
    B10001000,
    B01111000,
    6,

    B10000000, //b
    B10000000,
    B10110000,
    B11001000,
    B10001000,
    B10001000,
    B11110000,
    6,

    B00000000, //c
    B00000000,
    B01110000,
    B10001000,
    B10000000,
    B10001000,
    B01110000,
    6,

    B00001000, //d
    B00001000,
    B01101000,
    B10011000,
    B10001000,
    B10001000,
    B01111000,
    6,

    B00000000, //e
    B00000000,
    B01110000,
    B10001000,
    B11111000,
    B10000000,
    B01110000,
    6,

    B00110000, //f
```

```
    B01001000,
    B01000000,
    B11100000,
    B01000000,
    B01000000,
    B01000000,
    6,

    B00000000, //g
    B01111000,
    B10001000,
    B10001000,
    B01111000,
    B00001000,
    B01110000,
    6,

    B10000000, //h
    B10000000,
    B10110000,
    B11001000,
    B10001000,
    B10001000,
    B10001000,
    6,

    B01000000, //i
    B00000000,
    B11000000,
    B01000000,
    B01000000,
    B01000000,
    B11100000,
    4,

    B00010000, //j
    B00000000,
    B00110000,
    B00010000,
    B00010000,
    B10010000,
    B01100000,
    5,

    B10000000, //k
    B10000000,
    B10010000,
    B10100000,
    B11000000,
    B10100000,
    B10010000,
    5,

    B11000000, //l
    B01000000,
    B01000000,
    B01000000,
    B01000000,
    B01000000,
    B11100000,
    4,

    B00000000, //m
    B00000000,
    B11010000,
    B10101000,
    B10101000,
    B10001000,
    B10001000,
    6,

    B00000000, //n
    B00000000,
    B10110000,
```

```
    B11001000,
    B10001000,
    B10001000,
    B10001000,
    6,

    B00000000, //o
    B00000000,
    B01110000,
    B10001000,
    B10001000,
    B10001000,
    B01110000,
    6,

    B00000000, //p
    B00000000,
    B11110000,
    B10001000,
    B11110000,
    B10000000,
    B10000000,
    6,

    B00000000, //q
    B00000000,
    B01101000,
    B10011000,
    B01111000,
    B00001000,
    B00001000,
    6,

    B00000000, //r
    B00000000,
    B10110000,
    B11001000,
    B10000000,
    B10000000,
    B10000000,
    6,

    B00000000, //s
    B00000000,
    B01110000,
    B10000000,
    B01110000,
    B00001000,
    B11110000,
    6,

    B01000000, //t
    B01000000,
    B11100000,
    B01000000,
    B01000000,
    B01001000,
    B00110000,
    6,

    B00000000, //u
    B00000000,
    B10001000,
    B10001000,
    B10001000,
    B10011000,
    B01101000,
    6,

    B00000000, //v
    B00000000,
    B10001000,
    B10001000,
    B10001000,
```

```
    B01010000,
    B00100000,
    6,

    B00000000, //w
    B00000000,
    B10001000,
    B10101000,
    B10101000,
    B10101000,
    B01010000,
    6,

    B00000000, //x
    B00000000,
    B10001000,
    B01010000,
    B00100000,
    B01010000,
    B10001000,
    6,

    B00000000, //y
    B00000000,
    B10001000,
    B10001000,
    B01111000,
    B00001000,
    B01110000,
    6,

    B00000000, //z
    B00000000,
    B11111000,
    B00010000,
    B00100000,
    B01000000,
    B11111000,
    6,

    B00100000, //{
    B01000000,
    B01000000,
    B10000000,
    B01000000,
    B01000000,
    B00100000,
    4,

    B10000000, //|
    B10000000,
    B10000000,
    B10000000,
    B10000000,
    B10000000,
    B10000000,
    2,

    B10000000, //}
    B01000000,
    B01000000,
    B00100000,
    B01000000,
    B01000000,
    B10000000,
    4,

    B00000000, //~
    B00000000,
    B00000000,
    B01101000,
    B10010000,
    B00000000,
    B00000000,
```

```
    6,

    B01100000, // (Char 0x7F)
    B10010000,
    B10010000,
    B01100000,
    B00000000,
    B00000000,
    B00000000,
    5
};

void scrollFont() {
    for (int counter=0x20;counter<0x80;counter++){
        loadBufferLong(counter);
        delay(500);
    }
}

// Scroll Message
void scrollMessage(prog_uchar * messageString) {
    int counter = 0;
    int myChar=0;
    do {
        // read back a char
        myChar =  pgm_read_byte_near(messageString + counter);
        if (myChar != 0){
            loadBufferLong(myChar);
        }
        counter++;
    }
    while (myChar != 0);
}
// Load character into scroll buffer
void loadBufferLong(int ascii){
    if (ascii >= 0x20 && ascii <=0x7f){
        for (int a=0;a<7;a++){                      // Loop 7 times for a 5x7 font
            unsigned long c = pgm_read_byte_near(font5x7 + ((ascii - 0x20) * 8) + a);     //
Index into character table to get row data
            unsigned long x = bufferLong [a*2];     // Load current scroll buffer
            x = x | c;                              // OR the new character onto end of
current
            bufferLong [a*2] = x;                   // Store in buffer
        }
        byte count = pgm_read_byte_near(font5x7 +((ascii - 0x20) * 8) + 7);     // Index into
character table for kerning data
        for (byte x=0; x<count;x++){
            rotateBufferLong();
            printBufferLong();
            delay(scrollDelay);
        }
    }
}
// Rotate the buffer
void rotateBufferLong(){
    for (int a=0;a<7;a++){                      // Loop 7 times for a 5x7 font
        unsigned long x = bufferLong [a*2];     // Get low buffer entry
        byte b = bitRead(x,31);                 // Copy high order bit that gets lost in
rotation
        x = x<<1;                               // Rotate left one bit
        bufferLong [a*2] = x;                   // Store new low buffer
        x = bufferLong [a*2+1];                 // Get high buffer entry
        x = x<<1;                               // Rotate left one bit
        bitWrite(x,0,b);                        // Store saved bit
        bufferLong [a*2+1] = x;                 // Store new high buffer
    }
}
// Display Buffer on LED matrix
void printBufferLong(){
  for (int a=0;a<7;a++){                    // Loop 7 times for a 5x7 font
    unsigned long x = bufferLong [a*2+1];   // Get high buffer entry
    byte y = x;                             // Mask off first character
    lc.setRow(3,a,y);                       // Send row to relevent MAX7219 chip
    x = bufferLong [a*2];                   // Get low buffer entry
```

```
    y = (x>>24);                            // Mask off second character
    lc.setRow(2,a,y);                       // Send row to relevent MAX7219 chip
    y = (x>>16);                            // Mask off third character
    lc.setRow(1,a,y);                       // Send row to relevent MAX7219 chip
    y = (x>>8);                             // Mask off forth character
    lc.setRow(0,a,y);                       // Send row to relevent MAX7219 chip
  }
}
```

The pertinent parts are at the top of the sketch – the following line sets the number of MAX7219s in the hardware:

```
const int numDevices = 2;
```

The following can be adjusted to change the speed of text scrolling:

```
const long scrollDelay = 75;
```

… then place the text to scroll in the following (for example):

```
prog_uchar scrollText[] PROGMEM ={
    "  THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG 1234567890 the
quick brown fox jumped over the lazy dog   \0"};
```

Finally – to scroll the text on demand, use the following:

```
 scrollMessage(scrollText);
```

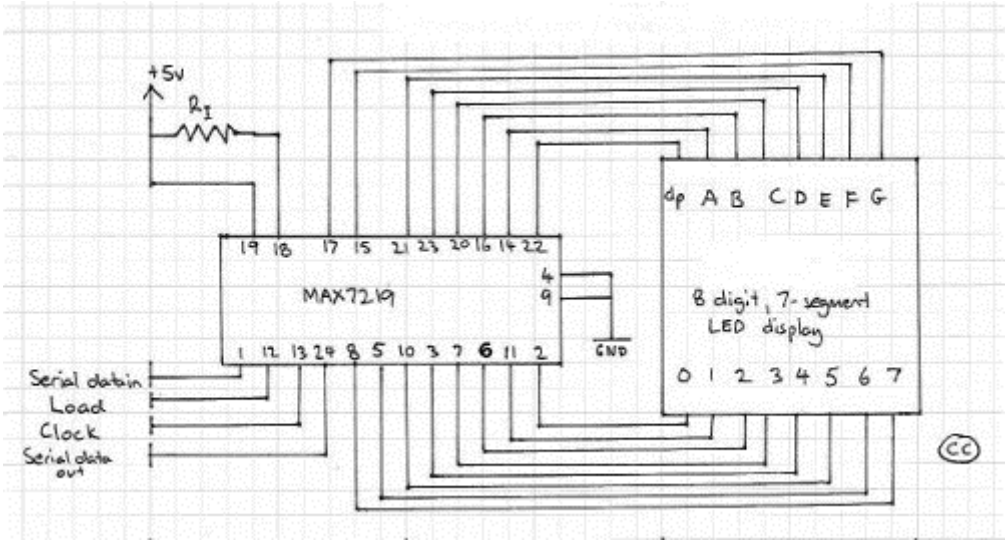
You can then incorporate the code into your own sketches. And a video of the example sketch in action:

Although we used the LedControl library, there are many others out there for scrolling text. One interesting example is *Parola* – which is incredibly customisable.

Controlling LED numeric displays with the MAX7219

Using the MAX7219 and the LedControl library you can also drive numeric LED displays – up to eight digits from the one MAX7219. This gives you the ability to make various numeric displays that are clear to read and easy to control. When shopping around for numeric LED displays, make sure you have the *common-cathode* type.

Connecting numeric displays is quite simple, consider the following schematic which should appear familiar by now:



The schematic shows the connections for modules or groups of up to eight digits. Each digit's A~F and dp (decimal point) anodes connect together to the MAX7219, and each digit's cathode connects in order as well. The MAX7219 will display each digit in turn by using one cathode at a time. Of course if you want more than eight digits, connect another MAX7219 just as we did with the LED matrices previously.

The required code in the sketch is identical to the LED matrix code, however to display individual digits we use:

```
lc.setDigit(A, B, C, D);
```

where A is the MAX7219 we're using, B is the digit to use (from a possible 0 to 7), C is the digit to display (0~9… if you use 10~15 it will display A~F respectively) and D is false/true (digit on or off). You can also send basic characters such as a dash "-" with the following:

```
lc.setChar(A, B,'-',false);
```

Now let's put together an example of eight digits:

```
#include "LedControl.h" //  need the library
LedControl lc=LedControl(12,11,10,1); // lc is our object
// pin 12 is connected to the MAX7219 pin 1
// pin 11 is connected to the CLK pin 13
// pin 10 is connected to LOAD pin 12
// 1 as we are only using 1 MAX7219
void setup()
{
  // the zero refers to the MAX7219 number, it is zero for 1 chip
  lc.shutdown(0,false);// turn off power saving, enables display
  lc.setIntensity(0,8);// sets brightness (0~15 possible values)
  lc.clearDisplay(0);// clear screen
}
void loop()
{
  for (int a=0; a<8; a++)
  {
    lc.setDigit(0,a,a,true);
    delay(100);
  }
  for (int a=0; a<8; a++)
  {
    lc.setDigit(0,a,8,1);
```

```
    delay(100);
  }
  for (int a=0; a<8; a++)
  {
    lc.setDigit(0,a,0,false);
    delay(100);
  }
  for (int a=0; a<8; a++)
  {
    lc.setChar(0,a,' ',false);
    delay(100);
  }
  for (int a=0; a<8; a++)
  {
    lc.setChar(0,a,'-',false);
    delay(100);
  }
  for (int a=0; a<8; a++)
  {
    lc.setChar(0,a,' ',false);
    delay(100);
  }
}
```

and the sketch in action:

Conclusion

We have only scratched the surface of what is possible with the MAX7219 and compatible parts. They're loads of fun and quite useful as well. And finally a plug for our own store – tronixlabs.com – which along with being Australia's #1 Adafruit distributor, also offers a growing range and Australia's best value for supported hobbyist electronics from DFRobot, Freetronics, Seeedstudio and much much more.