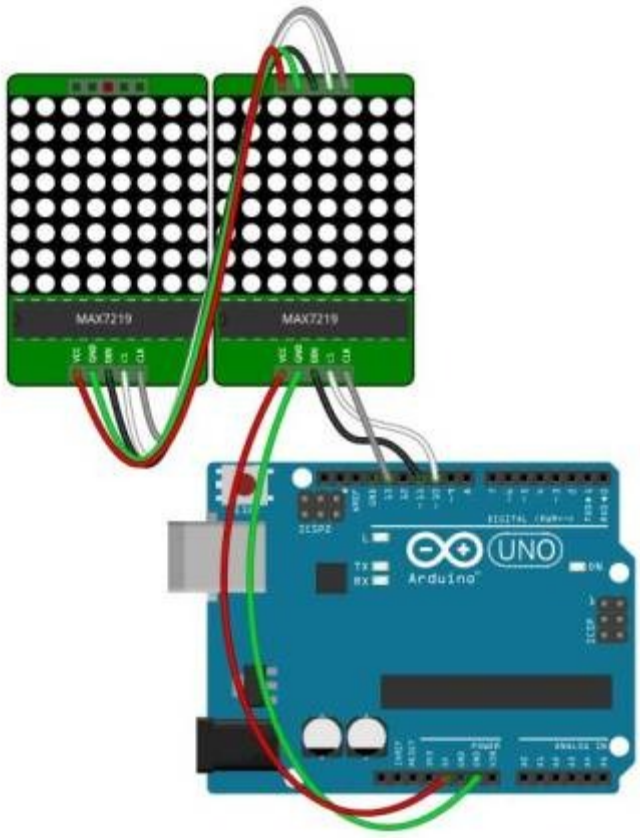
MAX7219: How to Use the MAX7219 to drive an 8x8 LED display Matrix on the Arduino.

The MAX7219 LED driver saves you processor pins and processing time! Below, you can find out exactly why this is true and how you can use these devices on an Arduino. You'll also see how to use multiple devices without using any more processor pins!

Using a 7219 you can drive 64 LEDs while you only need 4 wires to interface it to a microcontroller. In addition you can daisy chain multiple 7219 chips for bigger displays.

There are 16 output lines from the 7219 driving 64 individual LEDs. This sounds impossible but the driving method makes use of the way our eyes work. Persistence of vision is exploited to make the LEDs appear to be on all the time when in fact they are not. In fact the LEDs are arranged as an 8x8 set of rows and columns. Each colum is pulsed for a short time while the row bits for that column are driven.

Our eyes remember a flash of light for approximately 20ms, so when you continuously flash a light (or an LED) at a rate at or faster than 20ms, then it appears that the light never goes off. This is how the 7219 works. All the leds are individually turned on for a short time, at rate greater than 20ms.



MAX7219 SPI Interface

The MAX7219 has a four wire SPI interface - clock, data, chip select and ground - making it very simple to connect to a microcontroller.

1. Data - **MOSI** - Master Output Serial Input. The 7219 is a slave device.
2. Chip select - **Load (CSn)** - active low Chip select.
3. Clock - **SCK**
4. **Ground.**

MAX7219 Specifications

Parameter	**MAX7219**	
Power Supply:	4.0V ~ 5.5V	
Supply Current:	330mA	
Segment drive source current	-40mA	
Scan rate	500-1300Hz (800Hz Typ.)	
CLK	max 10MHz	
Voltage Specs.		
Vih (input high)	min 3.5V	
Vil (input low)	max 0.8V	
Voh (output high)	min V^{+}-1	
Vol (output low)	max 0.4V	

MAX7219 and MAX7221 Datasheet:

[MAX4192 Datasheet : http://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf]

Note: The MAX6950/MAX6951 is an equivalent device for a 3 or 3.3V system but there are differences so read the datasheet carefully e.g. no cascading - importantly you can not use them for an internally wired 7segment display (you have to have access to each LED segment (probably using a charlieplexing method - although this is not stated directly). The MAX6950/MAX6951 is available only in a surface mount package.

Display Brightness

There are two ways to control the display brightness.

- Via current setting resistor (Rset).
- By using an internal control register.

Resistor Brightness Control

You can not use one without the other though, as you have to use the resistor to set the maximum intensity (maximum current - the current into pin Iset - actual current is 100 times the Iset current value). The minimum value of the resistor should be 9.53k for a 40mA Brightness - in reality you just use a 10k resistor (this depends on the LEDs you use - probably should be using a 12k resistor - for maximum 7219 lifetime using RED LEDs - see below).

The current setting also depends on the forward voltage drop of the LEDs you use; Each different LED colour has a different forward voltage drop - see table 11:

Table 11 :RSET vs. Segment Current and LED Forward Voltage

Iseg (mA)	Vled (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

From datasheet:[MAX4192 **http://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf**

Here are forward voltage drops (at ~15mA constant current) for some of the LEDs on my bench:

LED	Vf (V) @~15mA
Red	1.91
Yellow	2.03
Green	2.14
Blue	3.07

Register Brightness control

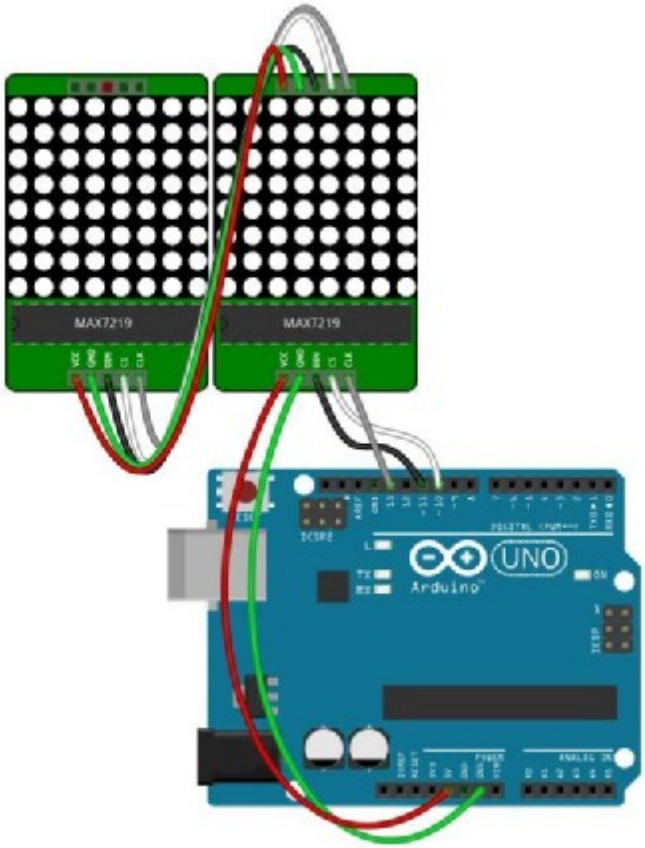
The second way to control brightness is more useful as it is under program control. The register 0xXA sets the level using the 4 lower data bits i.e. there are fifteen brightness level values with 15 being the brightest.

Wiring Layout for two MAX7219s

If you only have one breakout board leave off the left hand one.

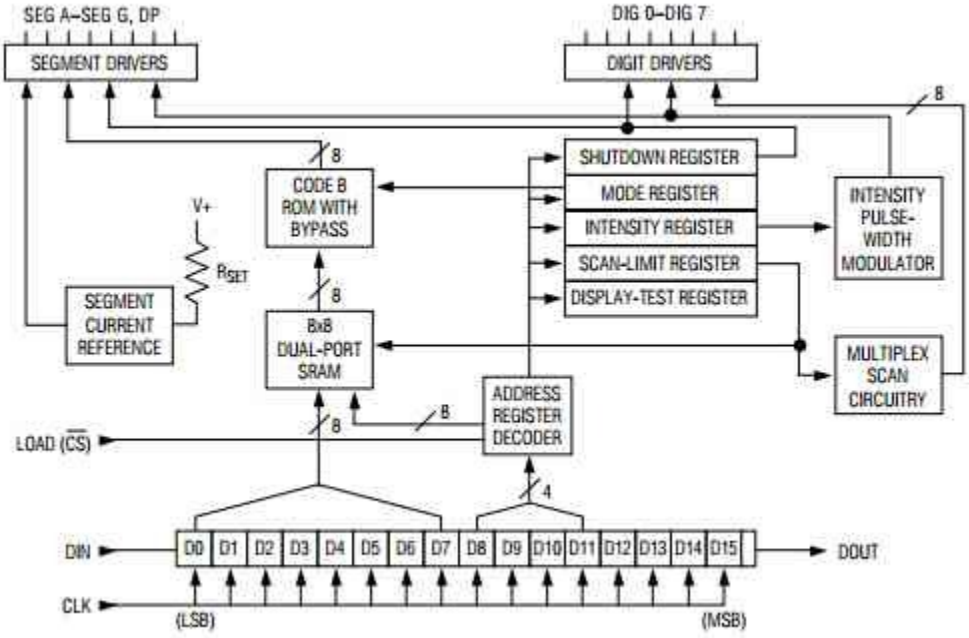
The layout below allows scrolling of text from left to right (It does not feel right to wire them up this way - but it is the way that this code allows for correct scrolling). Connections at the top of the board are identical to those at the bottom, except that the middle pin is Dout. The other 4 lines are connected straight through.

Note: To add more scrolling displays add them to the left and change the maxDisplays variable to match (2nd example code).



Understanding the MAX7219

The MAX7219 input interface is a 16 bit serial input shift register. The first 8 bits define a command while the second 8 bits define the data for that command (see diagram below):



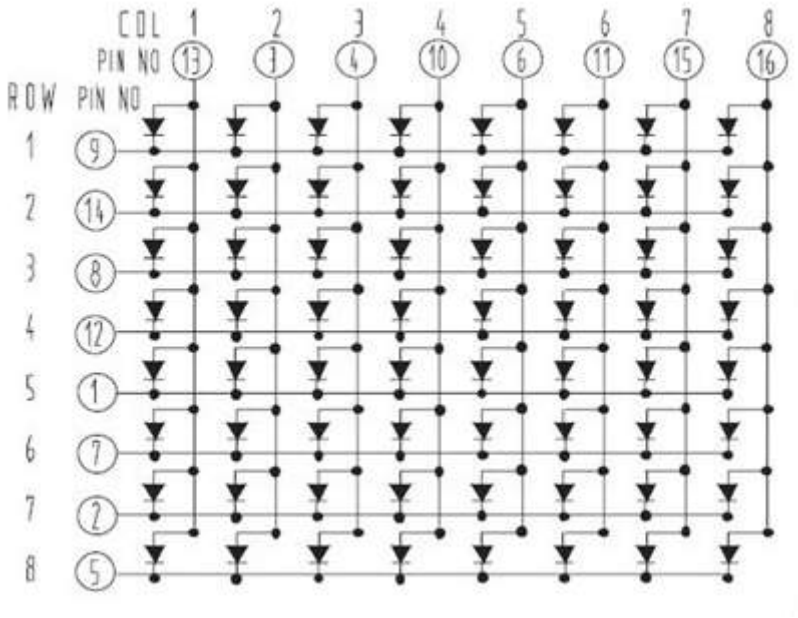
Only 4 address lines select the operation to perform - the other 4 are not used. Note that the MSB is sent first. The top 4 bits are redundant but you still have to send them into the shift register to fill that register completely.

The most important point is that multiple MAX7219 devices can be cascaded together simply by feeding in the Dout from one to the Din of the next. Load and clock (CLK) are kept the same for all devices.

The load pulse (CSn) is a trigger that sends all the data from the shift register into each control register in the device, at the same time. You can change the shift register contents as much as you need to (consider data flow when cascading many devices) and when all done, you transfer the data to the control registers using the CSn line by pulsing it low then high.

Dot Matrix Display 1088AS

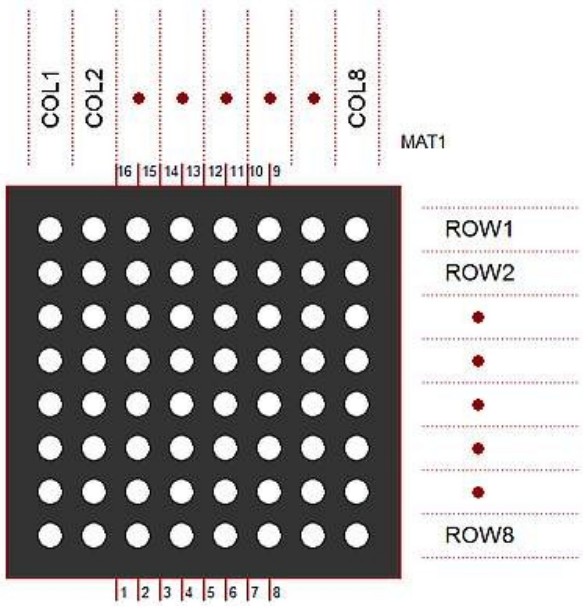
The datasheet shows the diode layout in a fairly unhelpful way (it is biased towards the row/columns and not towards the pin layout.:



Source [http://www.datasheetbank.com/datasheet/ETC/1088AS.html]

1088AS Row and Column orientation

In fact the pins are arranged physically as follows:



Note: The lower side with pins 1-8 can be identified by the small half moon tag on the case.

If you want to prove this is the case then use a multimeter on its diode check setting. Put Positive probe (red) to pin 16 and negative probe to (pin 1). You will then see diode in column 8 and row 5 light up.

TIP: The above method can be used to identify LED locations on any unknown LED MATRIX.

MAX7219 Libraries

There are quite a few libraries around (one calls itself yet-another MAX7219 library!).

Note: To get going quickly with scrolled text
just use the second library MaxMatrix below.

Some work better than others but of the few I chose at random - I did have some problems. What I want to do sounds simple enough - use two MAX7219 serial displays (64 LEDs) daisy chained and scroll some text across them.

One thing you should note is that these libraries probably are expecting more than two devices - that however should not stop them operating correctly! They may work for you!

Warning: These libraries are bit-banged and do not use the SPI interface (although they are attached to the correct pins if you want to use the internal SPI module - and find/write the code yourself!). Therefore they are fairly slow - especially if you want to do other tasks with the microcontroller. However they work well enough for the task here (scrolled text).

Library LedControl

The first library I tried provided basic operation of character output and allowing individual pixel control. But on my displays the text was back to front. This is a basic library that will use less memory than others - so it may be worth correcting if you need smaller code.

This library was installed using the Arduino library manager!

LedControl Examples

Edit the following (examples in the install directory) for the pins used (save as your own):

- LCDemoCascadedDevices.ino (2286 Bytes) Individual pixel control - does work.
- LCDemo7Segment.ino (2766 Bytes) - no idea what this is doing!
- LCDemoMatrix.ino (3584 Bytes) - Sort of works with backwards text.

Note: This library does not scroll text. Examples are for one 64 led unit only - you can edit them to use the other.

This library can be made to work by adding your own code on top but you have to write all the display shifting yourself - has been done!. One problem I had with that code is that it uses the deprecated type (where prog_uchar is no longer used in the new Arduino compiler):

prog_uchar

To get round this add the following to the top of the code:

```
#define prog_uchar const char
```

The other problem is that the scroll direction is left/right. The assumption is that you are using the LEDs correctly i.e. rows are horizontal and columns are vertical (this is correct for the displays that plug directly together i.e. when laid end to end rows and columns are in the correct orientation). The ones shown in the wiring diagram below have rows and columns swapped - the top of the LED display (pin diagram above) is the left side.

I am using ones in which it easier to have rows as vertical and columns as horizontal (see wiring diagram below). Would be nice to have a code switch to change this around as I have done on the non MAX7219 project here: (This LED matrix project directly controls a 8x8 64 LED matrix using 8 transistors (COLUMNS) and 8 data drives (ROWS)).

Alternative library Maxtrix_Sprite.

Another one you can try (I did not - just had a quick look) is Maxtrix_Sprite.This has an interesting way of defining a sprite that is not fixed to a specific size.

Use of all the functions in this library is more detailed here (although it does not show you what happens when you use the functions - see Example 3 for that).

Library MaxMatrix

This is a manual install to the <location of your sketches>/libraries - unzip to this directory.

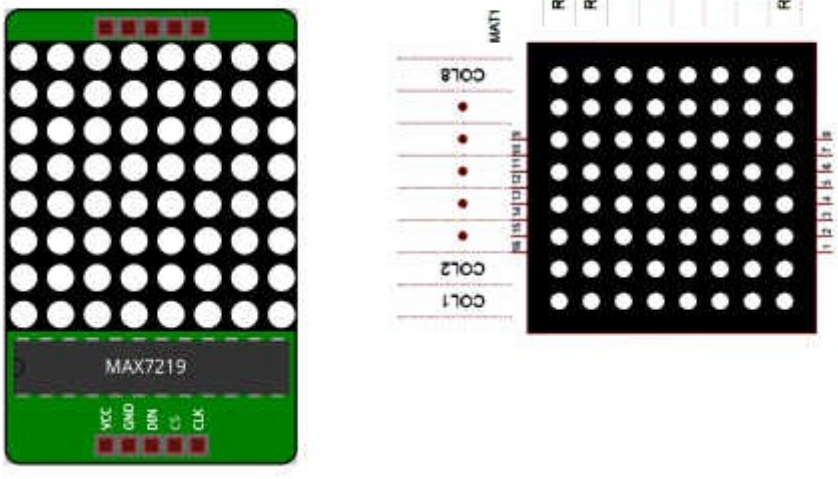
This second library I used (MaxMatrix). This provides a simple interface and multiple ways of scrolling the text:

- shiftLeft - probably the only function you need to use (see example 3)
- shiftUp
- shiftRight - see code notes (below) as this one does not work that well.
- shiftDown

For most users wanting to left scroll scrolling text this library works well but it only works with the modules oriented as in the breakout board shown in the wiring diagram below i.e. only with **COLS and ROWS swapped**. It will not work correctly with plug together modules that have correctly oriented rows and columns (ones with a surface mount MAX7912 mounted on the back).

Warning: This library will not work (text will not scroll to left) unless the 1088AS LED is oriented with rows vertical and columns horizontal (1088AS pinout) and each display is aligned beside the other (see wiring diagram).

This diagram shows what is going on:



MaxMatrix Example

Code size used (3482 Bytes) (with example below).

Arduino Software Setup for MAX7219:

IDE Version Used : 1.6.4
Board used : Arduino Uno R3
Device Used : MAX7219
LED Display: 64 element LED Matrix (8x8) - Part No. 1088AS.

Parts for MAX7219 project:

1. 2off - MAX7219 (on breakout board).
2. 2off - 64 element LED Matrix (8x8) - Part No. 1088AS (on breakout board).
3. 2off - 10k resistor (on breakout board).
4. Arduino Uno R3.
5. 2off - 10uF Electrolytic (on breakout board).

Arduino Sketch Example 1

This code uses 2 units and uses the internal SPI module. The display intensity is slowly scaled up and down while random pixels are lit/unlit on the display - quite good in a darkened room! It is based on code found here :

https://gist.github.com/nrdobie/8193350

...but modified to show operation with 2 units.

I have deliberately left the code as a basic operational code without algorithms e.g. not setting the number of display units, as this allows you to see fundamental operation of the system - and it keeps everything in one file.

Warning: You might hypnotize yourself with this random output!

To use one unit - **disconnect the last** one! - and hit reset.

Note: The serial debug output slows the code - remove it (and the delay(10)) to see VERY fast operation.

Note how code is added twice to fill up the two serially connected MAX7219 (internal) shift registers - see functions:

`maxTransferCMD`
`maxTransferDATA`

Example Code:

```
// max7219_two
// Based on https://gist.github.com/nrdobie/8193350

#include <SPI.h>
#define SS_PIN 10

// MAX7219 SPI LED Driver
#define MAX7219_TEST 0x0f // in real code put into a .h file
#define MAX7219_BRIGHTNESS 0x0a // in real code put into a .h file
#define MAX7219_SCAN_LIMIT 0x0b // in real code put into a .h file
#define MAX7219_DECODE_MODE 0x09 // in real code put into a .h file
#define MAX7219_SHUTDOWN 0x0C // in real code put into a .h file

void maxTransferCMD(uint8_t address, uint8_t value) {
uint8_t i;

  digitalWrite(SS_PIN, LOW);
  SPI.transfer(address);      // Send address.
  SPI.transfer(value);        // Send the value.
  SPI.transfer(address);      // Send address.
  SPI.transfer(value);        // Send the value.
  digitalWrite(SS_PIN, HIGH); // Finish transfer.
}

void maxTransferDATA(uint8_t address, uint8_t value, uint8_t v2) {
uint8_t i;

  digitalWrite(SS_PIN, LOW);

  SPI.transfer(address);      // Send address.
  SPI.transfer(value);        // Send the value.
  SPI.transfer(address);      // Send address.
  SPI.transfer(v2);           // Send the value.

  digitalWrite(SS_PIN, HIGH); // Finish transfer.
}

void setup() {

  Serial.begin(9600);

  Serial.println("Debug MAX7219");

  pinMode(SS_PIN, OUTPUT);

  SPI.setBitOrder(MSBFIRST);   // Reverse the SPI Data o/p.
  SPI.begin();                 // Start SPI
```

```
  // Run test - All LED segments lit.
  maxTransferCMD(MAX7219_TEST, 0x01);  delay(1000);
  maxTransferCMD(MAX7219_TEST, 0x00);        // Finish test mode.
  maxTransferCMD(MAX7219_DECODE_MODE, 0x00); // Disable BCD mode.
  maxTransferCMD(MAX7219_BRIGHTNESS, 0x00);  // Use lowest intensity.
  maxTransferCMD(MAX7219_SCAN_LIMIT, 0x0f);  // Scan all digits.
  maxTransferCMD(MAX7219_SHUTDOWN, 0x01);    // Turn on chip.

}

void loop() {
uint8_t row=0;
int i=0,ud=1; // Need signed numbers.

   for(;;) {

      i += ud;

      if (i>255) {ud=-1;i=255;}

      if (i<0)   {ud=1 ;i=0;}

      if (row++>8) row=1;

      maxTransferDATA(row, random(0,255), random(0,255));
      maxTransferCMD(MAX7219_BRIGHTNESS,i>>4);

      delay(10);

      Serial.print(" i ");
      Serial.print(i);
      Serial.print(" ud ");
      Serial.println(ud);
   }
}
```

Arduino
Sketch Example 2

This sketch scrolls text left using the MaxMatrix library using bit banged operation (the function used is shiftout - in the library code - this is an automatically included Arduino function) so devices can be used on any pins (at a slower rate).

Buffer Buf7219 is defined to allow the maximum data you use so if you use more than 5 data bytes increase it - others just set it to 10 - but that is confusing as it is not easy to see what it is used for unless you read the code.

Note: Observe the PROGMEM qualifier that puts data into internal programming memory and the access function that you must use to retrieve it memcpy_P().

```
// maxmatrix-disp-scroll-text-7219
// based on
// https://code.google.com/p/arudino-maxmatrix-
library/wiki/Example_Display_Scrolling_Tex
#include <MaxMatrix.h>
#include <avr/pgmspace.h>
```

```
#define maxDisplays 2 // Number of MAX7219's in use.

byte Buf7219[7]; // "width,height,data[5]" single character buffer.
const int data  = 11;     // DIN or MOSI
const int load  = 10;     // CS
const int clock = 13;     // SCK

MaxMatrix m(data, load, clock, maxDisplays);
// Data array is stored in program memory (see memcpy_P for access).
// Parameters are width, height, character data...
// There is a speed improvement for characters with height 8 bits see lib.

PROGMEM const unsigned char CH[] = {
3, 8, B0000000, B0000000, B0000000, B0000000, B0000000, // space
1, 8, B1011111, B0000000, B0000000, B0000000, B0000000, // !
3, 8, B0000011, B0000000, B0000011, B0000000, B0000000, // "
5, 8, B0010100, B0111110, B0010100, B0111110, B0010100, // #
4, 8, B0100100, B1101010, B0101011, B0010010, B0000000, // $
5, 8, B1100011, B0010011, B0001000, B1100100, B1100011, // %
5, 8, B0110110, B1001001, B1010110, B0100000, B1010000, // &
1, 8, B0000011, B0000000, B0000000, B0000000, B0000000, // '
3, 8, B0011100, B0100010, B1000001, B0000000, B0000000, // (
3, 8, B1000001, B0100010, B0011100, B0000000, B0000000, // )
5, 8, B0101000, B0011000, B0001110, B0011000, B0101000, // *
5, 8, B0001000, B0001000, B0111110, B0001000, B0001000, // +
2, 8, B10110000, B1110000, B0000000, B0000000, B0000000, // ,
4, 8, B0001000, B0001000, B0001000, B0001000, B0000000, // -
2, 8, B1100000, B1100000, B0000000, B0000000, B0000000, // .
4, 8, B1100000, B0011000, B0000110, B0000001, B0000000, // /
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // 0
3, 8, B1000010, B1111111, B1000000, B0000000, B0000000, // 1
4, 8, B1100010, B1010001, B1001001, B1000110, B0000000, // 2
4, 8, B0100010, B1000001, B1001001, B0110110, B0000000, // 3
4, 8, B0011000, B0010100, B0010010, B1111111, B0000000, // 4
4, 8, B0100111, B1000101, B1000101, B0111001, B0000000, // 5
4, 8, B0111110, B1001001, B1001001, B0110000, B0000000, // 6
4, 8, B1100001, B0010001, B0001001, B0000111, B0000000, // 7
4, 8, B0110110, B1001001, B1001001, B0110110, B0000000, // 8
4, 8, B0000110, B1001001, B1001001, B0111110, B0000000, // 9
2, 8, B01010000, B0000000, B0000000, B0000000, B0000000, // :
2, 8, B10000000, B01010000, B0000000, B0000000, B0000000, // ;
3, 8, B0010000, B0101000, B1000100, B0000000, B0000000, // <
3, 8, B0010100, B0010100, B0010100, B0000000, B0000000, // =
3, 8, B1000100, B0101000, B0010000, B0000000, B0000000, // >
4, 8, B0000010, B1011001, B0001001, B0000110, B0000000, // ?
5, 8, B0111110, B1001001, B1010101, B1011101, B0001110, // @
4, 8, B1111110, B0010001, B0010001, B1111110, B0000000, // A
4, 8, B1111111, B1001001, B1001001, B0110110, B0000000, // B
4, 8, B0111110, B1000001, B1000001, B0100010, B0000000, // C
4, 8, B1111111, B1000001, B1000001, B0111110, B0000000, // D
4, 8, B1111111, B1001001, B1001001, B1000001, B0000000, // E
4, 8, B1111111, B0001001, B0001001, B0000001, B0000000, // F
4, 8, B0111110, B1000001, B1001001, B1111010, B0000000, // G
4, 8, B1111111, B0001000, B0001000, B1111111, B0000000, // H
3, 8, B1000001, B1111111, B1000001, B0000000, B0000000, // I
4, 8, B0110000, B1000000, B1000001, B0111111, B0000000, // J
4, 8, B1111111, B0001000, B0010100, B1100011, B0000000, // K
4, 8, B1111111, B1000000, B1000000, B1000000, B0000000, // L
5, 8, B1111111, B0000010, B0001100, B0000010, B1111111, // M
5, 8, B1111111, B0000100, B0001000, B0010000, B1111111, // N
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // O
4, 8, B1111111, B0001001, B0001001, B0000110, B0000000, // P
4, 8, B0111110, B1000001, B1000001, B10111110, B0000000, // Q
4, 8, B1111111, B0001001, B0001001, B1110110, B0000000, // R
4, 8, B1000110, B1001001, B1001001, B0110010, B0000000, // S
5, 8, B0000001, B0000001, B1111111, B0000001, B0000001, // T
```

```
4, 8, B0111111, B1000000, B1000000, B0111111, B0000000, // U
5, 8, B0001111, B0110000, B1000000, B0110000, B0001111, // V
5, 8, B0111111, B1000000, B0111000, B1000000, B0111111, // W
5, 8, B1100011, B0010100, B0001000, B0010100, B1100011, // X
5, 8, B0000111, B0001000, B1110000, B0001000, B0000111, // Y
4, 8, B1100001, B1010001, B1001001, B1000111, B0000000, // Z
2, 8, B1111111, B1000001, B0000000, B0000000, B0000000, // [
4, 8, B0000001, B0000110, B0011000, B1100000, B0000000, // backslash
2, 8, B1000001, B1111111, B0000000, B0000000, B0000000, // ]
3, 8, B0000010, B0000001, B0000010, B0000000, B0000000, // hat
4, 8, B1000000, B1000000, B1000000, B1000000, B0000000, // _
2, 8, B0000001, B0000010, B0000000, B0000000, B0000000, // `
4, 8, B0100000, B1010100, B1010100, B1111000, B0000000, // a
4, 8, B1111111, B1000100, B1000100, B0111000, B0000000, // b
4, 8, B0111000, B1000100, B1000100, B0000000, B0000000, // c // JFM MOD.
4, 8, B0111000, B1000100, B1000100, B1111111, B0000000, // d
4, 8, B0111000, B1010100, B1010100, B0011000, B0000000, // e
3, 8, B0000100, B1111110, B0000101, B0000000, B0000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B0000000, // g
4, 8, B1111111, B0000100, B0000100, B1111000, B0000000, // h
3, 8, B1000100, B1111101, B1000000, B0000000, B0000000, // i
4, 8, B1000000, B10000000, B10000100, B1111101, B0000000, // j
4, 8, B1111111, B0010000, B0101000, B1000100, B0000000, // k
3, 8, B1000001, B1111111, B1000000, B0000000, B0000000, // l
5, 8, B1111100, B0000100, B1111100, B0000100, B1111000, // m
4, 8, B1111100, B0000100, B0000100, B1111000, B0000000, // n
4, 8, B0111000, B1000100, B1000100, B0111000, B0000000, // o
4, 8, B11111100, B0100100, B0100100, B0011000, B0000000, // p
4, 8, B0011000, B0100100, B0100100, B11111100, B0000000, // q
4, 8, B1111100, B0001000, B0000100, B0000100, B0000000, // r
4, 8, B1001000, B1010100, B1010100, B0100100, B0000000, // s
3, 8, B0000100, B0111111, B1000100, B0000000, B0000000, // t
4, 8, B0111100, B1000000, B1000000, B1111100, B0000000, // u
5, 8, B0011100, B0100000, B1000000, B0100000, B0011100, // v
5, 8, B0111100, B1000000, B0111100, B1000000, B0111100, // w
5, 8, B1000100, B0101000, B0010000, B0101000, B1000100, // x
4, 8, B10011100, B10100000, B10100000, B1111100, B0000000, // y
3, 8, B1100100, B1010100, B1001100, B0000000, B0000000, // z
3, 8, B0001000, B0110110, B1000001, B0000000, B0000000, // {
1, 8, B1111111, B0000000, B0000000, B0000000, B0000000, // |
3, 8, B1000001, B0110110, B0001000, B0000000, B0000000, // }
4, 8, B0001000, B0000100, B0001000, B0000100, B0000000, // ~
};

void setup()
{
  m.init();
  m.setIntensity(3);
}

// Scrolling Text
char string[] = "The quick brown fox juumped over the lazy DOG.;:@'+=-/&";

void loop()
{
  delay(100);
  m.shiftLeft(false, true);
  printStringWithShift(string,100);
}

void printCharWithShift(char c, int shift_speed)
{
  if (c < 32) return;
  c -= 32;
  memcpy_P(Buf7219, CH + 7*c, 7);
  m.writeSprite(maxDisplays*8, 0, Buf7219);
```

```
  m.setColumn(maxDisplays*8 + Buf7219[0], 0);


  for (int i=0; i<=Buf7219[0]; i++)
  {
    delay(shift_speed);
    m.shiftLeft(false, false);
  }
}

void printStringWithShift(char* s, int shift_speed)
{
  while (*s != 0)
  {
    printCharWithShift(*s, shift_speed);
    s++;
  }
}
```

Arduino
Sketch Example 3

You can explore how the MaxMatrix library works/"does not work" with this code.This code explores the four shift directions provided in the MaxMatrix library.

- shiftLeft - probably the only function you need to use (see example 3)
- shiftUp
- shiftRight - see code notes (below) as this one does not work that well.
- shiftDown

Note how the example for right shift first left shifts the data by two and loses two columns because the library does not perform wrap around.

```
// t1-maxmatrix-disp-scroll-text-7219
// sort of based on (but not entirely!)
// https://code.google.com/p/arudino-maxmatrix-
library/wiki/Example_Display_Scrolling_Text
#include <MaxMatrix.h>
#include <avr/pgmspace.h>

#define maxDisplays 2 // Number of MAX7219's in use.

byte Buf7219[10]; // width,height,data single character buffer.
const int data  = 11;     // DIN or MOSI
const int load  = 10;     // CS
const int clock = 13;     // SCK

MaxMatrix m(data, load, clock, maxDisplays);

PROGMEM const unsigned char CH[] = {
3, 8, B0000000, B0000000, B0000000, B0000000, B0000000, // space
1, 8, B1011111, B0000000, B0000000, B0000000, B0000000, // !
3, 8, B0000011, B0000000, B0000011, B0000000, B0000000, // "
5, 8, B0010100, B0111110, B0010100, B0111110, B0010100, // #
4, 8, B0100100, B1101010, B0101011, B0010010, B0000000, // $
5, 8, B1100011, B0010011, B0001000, B1100100, B1100011, // %
5, 8, B0110110, B1001001, B1010110, B0100000, B1010000, // &
1, 8, B0000011, B0000000, B0000000, B0000000, B0000000, // '
3, 8, B0011100, B0100010, B1000001, B0000000, B0000000, // (
```

```
3, 8, B1000001, B0100010, B0011100, B0000000, B0000000, // )
5, 8, B0101000, B0011000, B0001110, B0011000, B0101000, // *
5, 8, B0001000, B0001000, B0111110, B0001000, B0001000, // +
2, 8, B10110000, B1110000, B0000000, B0000000, B0000000, // ,
4, 8, B0001000, B0001000, B0001000, B0001000, B0000000, // -
2, 8, B1100000, B1100000, B0000000, B0000000, B0000000, // .
4, 8, B1100000, B0011000, B0000110, B0000001, B0000000, // /
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // 0
3, 8, B1000010, B1111111, B1000000, B0000000, B0000000, // 1
4, 8, B1100010, B1010001, B1001001, B1000110, B0000000, // 2
4, 8, B0100010, B1000001, B1001001, B0110110, B0000000, // 3
4, 8, B0011000, B0010100, B0010010, B1111111, B0000000, // 4
4, 8, B0100111, B1000101, B1000101, B0111001, B0000000, // 5
4, 8, B0111110, B1001001, B1001001, B0110000, B0000000, // 6
4, 8, B1100001, B0010001, B0001001, B0000111, B0000000, // 7
4, 8, B0110110, B1001001, B1001001, B0110110, B0000000, // 8
4, 8, B0000110, B1001001, B1001001, B0111110, B0000000, // 9
2, 8, B01010000, B0000000, B0000000, B0000000, B0000000, // :
2, 8, B10000000, B01010000, B0000000, B0000000, B0000000, // ;
3, 8, B0010000, B0101000, B1000100, B0000000, B0000000, // <
3, 8, B0010100, B0010100, B0010100, B0000000, B0000000, // =
3, 8, B1000100, B0101000, B0010000, B0000000, B0000000, // >
4, 8, B0000010, B1011001, B0001001, B0000110, B0000000, // ?
5, 8, B0111110, B1001001, B1010101, B1011101, B0001110, // @
4, 8, B1111110, B0010001, B0010001, B1111110, B0000000, // A
4, 8, B1111111, B1001001, B1001001, B0110110, B0000000, // B
4, 8, B0111110, B1000001, B1000001, B0100010, B0000000, // C
4, 8, B1111111, B1000001, B1000001, B0111110, B0000000, // D
4, 8, B1111111, B1001001, B1001001, B1000001, B0000000, // E
4, 8, B1111111, B0001001, B0001001, B0000001, B0000000, // F
4, 8, B0111110, B1000001, B1001001, B1111010, B0000000, // G
4, 8, B1111111, B0001000, B0001000, B1111111, B0000000, // H
3, 8, B1000001, B1111111, B1000001, B0000000, B0000000, // I
4, 8, B0110000, B1000000, B1000001, B0111111, B0000000, // J
4, 8, B1111111, B0001000, B0010100, B1100011, B0000000, // K
4, 8, B1111111, B1000000, B1000000, B1000000, B0000000, // L
5, 8, B1111111, B0000010, B0001100, B0000010, B1111111, // M
5, 8, B1111111, B0000100, B0001000, B0010000, B1111111, // N
4, 8, B0111110, B1000001, B1000001, B0111110, B0000000, // O
4, 8, B1111111, B0001001, B0001001, B0000110, B0000000, // P
4, 8, B0111110, B1000001, B1000001, B10111110, B0000000, // Q
4, 8, B1111111, B0001001, B0001001, B1110110, B0000000, // R
4, 8, B1000110, B1001001, B1001001, B0110010, B0000000, // S
5, 8, B0000001, B0000001, B1111111, B0000001, B0000001, // T
4, 8, B0111111, B1000000, B1000000, B0111111, B0000000, // U
5, 8, B0001111, B0110000, B1000000, B0110000, B0001111, // V
5, 8, B0111111, B1000000, B0111000, B1000000, B0111111, // W
5, 8, B1100011, B0010100, B0001000, B0010100, B1100011, // X
5, 8, B0000111, B0001000, B1110000, B0001000, B0000111, // Y
4, 8, B1100001, B1010001, B1001001, B1000111, B0000000, // Z
2, 8, B1111111, B1000001, B0000000, B0000000, B0000000, // [
4, 8, B0000001, B0000110, B0011000, B1100000, B0000000, // backslash
2, 8, B1000001, B1111111, B0000000, B0000000, B0000000, // ]
3, 8, B0000010, B0000001, B0000010, B0000000, B0000000, // hat
4, 8, B1000000, B1000000, B1000000, B1000000, B0000000, // _
2, 8, B0000001, B0000010, B0000000, B0000000, B0000000, // `
4, 8, B0100000, B1010100, B1010100, B1111000, B0000000, // a
4, 8, B1111111, B1000100, B1000100, B0111000, B0000000, // b
4, 8, B0111000, B1000100, B1000100, B0000000, B0000000, // c // JFM MOD.
4, 8, B0111000, B1000100, B1000100, B1111111, B0000000, // d
4, 8, B0111000, B1010100, B1010100, B0011000, B0000000, // e
3, 8, B0000100, B1111110, B0000101, B0000000, B0000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B0000000, // g
4, 8, B1111111, B0000100, B0000100, B1111000, B0000000, // h
3, 8, B1000100, B1111101, B1000000, B0000000, B0000000, // i
4, 8, B1000000, B10000000, B10000100, B1111101, B0000000, // j
```

```
4, 8, B1111111, B0010000, B0101000, B1000100, B0000000, // k
3, 8, B1000001, B1111111, B1000000, B0000000, B0000000, // l
5, 8, B1111100, B0000100, B1111100, B0000100, B1111000, // m
4, 8, B1111100, B0000100, B0000100, B1111000, B0000000, // n
4, 8, B0111000, B1000100, B1000100, B0111000, B0000000, // o
4, 8, B11111100, B0100100, B0100100, B0011000, B0000000, // p
4, 8, B0011000, B0100100, B0100100, B11111100, B0000000, // q
4, 8, B1111100, B0001000, B0000100, B0000100, B0000000, // r
4, 8, B1001000, B1010100, B1010100, B0100100, B0000000, // s
3, 8, B0000100, B0111111, B1000100, B0000000, B0000000, // t
4, 8, B0111100, B1000000, B1000000, B1111100, B0000000, // u
5, 8, B0011100, B0100000, B1000000, B0100000, B0011100, // v
5, 8, B0111100, B1000000, B0111100, B1000000, B0111100, // w
5, 8, B1000100, B0101000, B0010000, B0101000, B1000100, // x
4, 8, B10011100, B10100000, B10100000, B1111100, B0000000, // y
3, 8, B1100100, B1010100, B1001100, B0000000, B0000000, // z
3, 8, B0001000, B0110110, B1000001, B0000000, B0000000, // {
1, 8, B1111111, B0000000, B0000000, B0000000, B0000000, // |
3, 8, B1000001, B0110110, B0001000, B0000000, B0000000, // }
4, 8, B0001000, B0000100, B0001000, B0000100, B0000000, // ~
};


void setup()
{
  m.init();
  m.setIntensity(3);
}

char string[] = "Best-Microcontroller-Projects.com";  // Scrolling Text

void loop()
{
  m.shiftLeft(false, true);

  byte CH_A[] = {4, 8, B1111110, B0010001, B0010001, B1111110 };

  // Down
  m.writeSprite(0, 0, CH_A);
  for (int i=0; i<8; i++)
  {
    delay(300);
    m.shiftDown(true); // rotate = true = rotate within 8 bit block
  }

  delay(500);

  m.clear();

  // UP
  m.writeSprite(0, 0, CH_A);
  for (int i=0; i<8; i++)
  {
    delay(300);
    m.shiftUp(true);  // rotate = true = rotate within 8 bit block
  }

  delay(500);
  m.clear();

  m.writeSprite(0, 0, CH_A);

  delay(500);

  m.clear();
```

```
  // Left
  m.writeSprite(8, 0, CH_A); // write sprite OUTSIDE the LED matrix
  for (int i=0; i<8; i++)
  {
    delay(300);
    m.shiftLeft(false, false);
  }

  delay(500);
  m.clear();

  // Right - does not work in the same way as shiftleft.
  m.writeSprite(0, 0, CH_A);
  m.shiftLeft(false, false); // Code library does not work for shiftleft
  m.shiftLeft(false, false); // unless all chars are on the screen!

  for (int i=0; i<8; i++)
  {
    delay(300);
    m.shiftRight(true, false);
  }

  delay(500);
  m.clear();
}

void printCharWithShift(char c, int shift_speed)
{
  if (c < 32) return;
  c -= 32;
  memcpy_P(Buf7219, CH + 7*c, 7);
  m.writeSprite(maxDisplays*8, 0, Buf7219);
  m.setColumn(maxDisplays*8 + Buf7219[0], 0);

  for (int i=0; i<Buf7219[0]+1; i++)
  {
    delay(shift_speed);
    m.shiftLeft(false, false);
  }
}

void printStringWithShift(char* s, int shift_speed)
{
  while (*s != 0)
  {
    printCharWithShift(*s, shift_speed);
    s++;
  }
}
```

A Bit extra on the MAX7219, MIPS and 3V3 Interfacing

To Decode or Not to Decode - That is the question.

The MAX7219 is capable of automatic decode of binary coded decimal values (BCD). This means, when enabled, that 7 segment display LEDs are correctly illuminated to display a number when you enter a BCD value.

Note: A BCD number is just a binary number, but you are only allowed digits 0 to 9 i.e. 0000 to 1001 anything higher is invalid. i.e. it is decimal or "Binary coded Decimal".

There are routines in most C libraries for operating with BCD values so you can do normal arithmetic with BCD number strings.

The best thing about BCD is that it is easy to use within assembler or C code i.e. you can easily use a library routine to translate a binary number to its BCD equivalent - fairly fast. Since the MAX7219 can be set to BCD decode mode you don't need to figure out a character set for controlling individual LED patterns.

For instance this would be useful when displaying numbers on a frequency counter's 7 segment display.

Note: BCD decode can be set-up for individual 7 segments leaving the rest as non-decoded i.e. individual LED control. So a possible use could be having a 4 digit display plus 32 individual LED outputs, or a 5 digit display with 24 individual LED outputs etc.

Why Does a MAX7219 save Processing Time (MIPS)?

The reason is that the display refresh operation is performed entirely within the MAX7219. In the frequency counter that you can find and make on this site (see link), I used a direct drive method that performs exactly as the MAX7219 driver. The only difference is that all display refreshing is done within the microcontroller which requires interrupts .i.e. it has to update the display faster than 20ms for you to see an unflickering display.

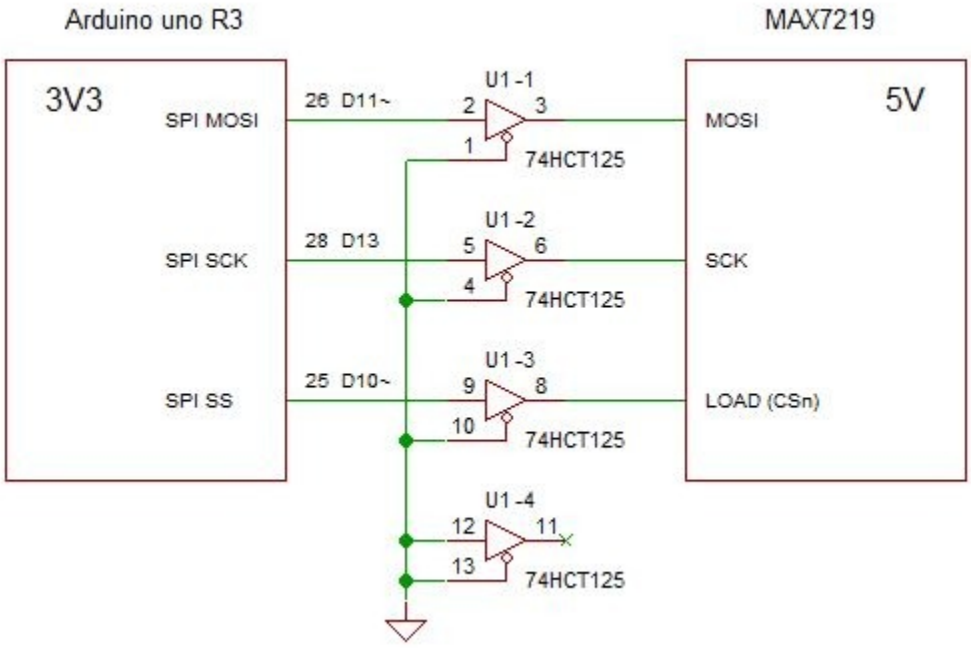
Note: Display refreshing also requires that you decide how long to output the data (how much LED light is output) and that you cycle through all 8 data outputs so that ALL 7 segments have been serviced within 20ms. With the MAX 7219, it does all this for you.

Using the MAX7219 allows you to chuck some data at it and forget it - it will always display the data until the power goes off and it does not require any other processor input or control until you want to change the data.

Using the MAX7219 with 3.3V Arduinos

As you can see from the specifications above the high input voltage level is really high i.e. way above 3.3V high output level, so a 3.3V Arduino needs a level translator to drive the inputs. The low input Vil is a standard level so that is not a problem!

The 7219 uses a fast SPI interface that can go at MHz frequencies and you may want to drive it at fast rates, depending on your project, so you need a fast level converter and some are slow. Since level conversion is only required in a single direction, one solution is to use an HCT device such as a 74HCT04 inverter running at 5V (just use two inverters for correct signal levels or just accept inverted signal levels - change them at the microcontroller) or you could use a buffer such as the 74HCT125.



Voltage analysis from 3v3 ATMeg328p (Arduino) to HCT device

Arduino outputs @3V3	**3V3 levels (V)**	**HCT inputs**	**TTL i/p Levels (V)**	**Condition for operation**
Vol	0.6 (max)	Vil	0.8 (max)	ARDVol<HCTVil = True
Voh	2.3 (min)	Vih	2.0 (min)	ARDVoh>HCTVih = True

Voltage analysis from HCT device to MAX7219

HCT outputs @5V	**5V levels (V)**	**MAX7219**	**5V levels (V)**	**Condition for operation**
Vol	0.1 (max)	Vil	0.8 (max)	HCTVol<7219Vil = True
Voh	4.4 (min)	Vih	3.5 (min)	HCTVoh>7219Vil = True

So the 3v3 system can be interfaced with the MAX7219:

Arduino Voltage levels

Arduino Voltage levels (328P datasheet)			
Parameter	**(V)**	**At Vcc of 3.3V (V)**	**At Vcc of 5V (V)**
Vil (max)	0.3 x Vcc	1.0 (max)(2.4V-5.5V)	1.5 (max)
Vih (min)	0.6 x Vcc	2.0 (min) (2.4V-5.5V)	3.0 (min)
Vol (max)	-	0.6 (max) (3V,85°C)	0.9 (max)(85°C)
Voh (min)	-	2.3 (min) (3V,85°C)	4.2 (min) (85°C)

HCT voltage levels

HCT Voltage levels	
Parameter	**At Vcc of 5V (V)**
Vil (max)	0.8 (max)
Vih (min)	2.0 (min)
Vol (max)	0.1 (max)(4.5V)
Voh (min)	4.4 (min) (4.5V)

Datasheet source for HCT levels above **[http://www.nxp.com/documents/data_sheet/74HC_HCT125.pdf]**

Other similar projects on this site:

LED matrix project
- does not require a MAX7219 - just transistors.