#include <LedControl.h>

#include <LedControl.h>

//Found from Electronoobs for Max 7219 8x8 LED Matrix Scrolling Text Sketch.

//Pat McMahon 12/5/2020.

//A002

/\* Dot matrix control Scrolling

\* Tutorial: https://www.electronoobs.com/eng\_arduino\_tut56.php

Schematic: https://www.electronoobs.com/eng\_arduino\_tut56\_sch1.php

LedControl library: https://www.electronoobs.com/eng\_arduino\_ledcontrol.php

\*/

// DIN= pin 12, CS= pin 10, CLK= pin 11

#include <LedControl.h> //Led Control library: https://www.electronoobs.com/ledcontrol.php

const int numDevices=8 ; // number of MAX7219s used in this case 4

const long scrollDelay = 100; // adjust scrolling speed

//unsigned long bufferLong [14] = {0};

unsigned long bufferLong [28] = {0};

LedControl lc=LedControl(12,11,10,numDevices);//DATA | CLK | CS/LOAD | number of matrices

const unsigned char scrollText[] PROGMEM ={" Evan & Finley "}; //replace the text between quotation marks with your own message.

unsigned long delaytime=500;

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,2); // Set the brightness to default value

lc.clearDisplay(x); // and clear the display

}

}

void loop(){

/\*

//read the number cascaded devices

int devices=lc.getDeviceCount();

//we have to init all devices in a loop

for(int row=0;row<8;row++) {

for(int col=0;col<8;col++) {

for(int address=0;address<devices;address++) {

delay(delaytime);

lc.setLed(address,row,col,true);

delay(delaytime);

lc.setLed(address,row,col,false);

}

}

}

\*/

scrollMessage(scrollText); //scrollFont();

}

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

const unsigned char font5x7 [] PROGMEM = { //Numeric Font Matrix (Arranged as 7x font data + 1x kerning data)

B00000000, //Space (Char 0x20)

B00000000,

B00000000,

B00000000,

B00000000,

B00000000,

B00000000,

3,//cambias el tamaño del espacio entre letras

B01000000, //!

B01000000,

B01000000,

B01000000,

B01000000,

B00000000,

B01000000,

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,

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,

B00000000, // smiley

B01100000,

B01100110,

B00000000,

B10000001,

B01100110,

B00011000,

5

};

void scrollFont() {

for (int counter=0x20;counter<0x80;counter++){

loadBufferLong(counter);

delay(500);

}

}

// Scroll Message

void scrollMessage(const unsigned char \* 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

// byte b = bitRead(x,63); // 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(7,a,y); // Send row to relevent MAX7219 chip

x = bufferLong [a\*2]; // Get low buffer entry

y = (x>>56); // Mask off second character

lc.setRow(6,a,y); // Send row to relevent MAX7219 chip

y = (x>>48); // Mask off second character

// lc.setRow(5,a,y); // Send row to relevent MAX7219 chip

y = (x>>40); // Mask off second character

lc.setRow(4,a,y); // Send row to relevent MAX7219 chip

y = (x>>32); // Mask off second character

lc.setRow(3,a,y); // Send row to relevent MAX7219 chip

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

}

}