#define BA {B01110000,B10001000,B10001000,B11111000,B10001000,B10001000}

#define BB {B11110000,B10001000,B10001000,B11110000,B10001000,B11111000}

#define BC {B11111000,B10000000,B10000000,B10000000,B10000000,B11111000}

#define BD {B11110000,B10001000,B10001000,B10001000,B10001000,B11110000}

#define BE {B11111000,B10000000,B10000000,B11110000,B10000000,B11111000}

#define BF {B11111000,B10000000,B10000000,B11110000,B10000000,B10000000}

#define BG {B01110000,B10001000,B10000000,B10011000,B10001000,B01110000}

#define BH {B10001000,B10001000,B11111000,B10001000,B10001000,B10001000}

#define BI {B11111000,B00100000,B00100000,B00100000,B00100000,B11111000}

#define BJ {B00111000,B00010000,B00010000,B00010000,B10010000,B01100000}

#define BM {B10001000,B11011000,B10101000,B10101000,B10001000,B10001000}

#define BN {B10001000,B11001000,B10101000,B10101000,B10011000,B10001000}

#define BL {B10000000,B10000000,B10000000,B10000000,B10000000,B11111000}

#define BO {B01110000,B10001000,B10001000,B10001000,B10001000,B01110000}

#define BP {B11110000,B10001000,B10001000,B11110000,B10000000,B10000000}

#define BQ {B01110000,B10001000,B10101000,B10011000,B01111000,B00001000}

#define BR {B11110000,B10001000,B10001000,B11110000,B10001000,B10001000}

#define BS {B01110000,B10001000,B01100000,B00010000,B10001000,B01110000}

#define BK {B10001000,B10010000,B11100000,B11100000,B10010000,B10001000}

#define BT {B11111000,B00100000,B00100000,B00100000,B00100000,B00100000}

#define BU {B10001000,B10001000,B10001000,B10001000,B10001000,B01110000}

#define BV {B10001000,B10001000,B10001000,B10001000,B01010000,B00100000}

#define BW {B10001000,B10001000,B10101000,B10101000,B10101000,B01010000}

#define BX {B10001000,B01010000,B00100000,B00100000,B01010000,B10001000}

#define BY {B10001000,B01010000,B00100000,B00100000,B00100000,B00100000}

#define BZ {B11111000,B00001000,B00110000,B01100000,B10000000,B11111000}

#define LA{B00000000,B01110000,B00001000,B01111000,B10001000,B01111000}

#define LB{B10000000,B10000000,B10110000,B11001000,B10001000,B11110000}

#define LC{B00000000,B01110000,B10000000,B10000000,B10001000,B01110000}

#define LD{B00001000,B00001000,B01111000,B10001000,B10001000,B01111000}

#define LE{B00000000,B01110000,B10001000,B11111000,B10000000,B01110000}

#define LF{B00110000,B01001000,B01000000,B11100000,B01000000,B01000000}

#define LG{B00000000,B01111000,B10001000,B01111000,B00001000,B01110000}

#define LH{B10000000,B10000000,B10110000,B11001000,B10001000,B10001000}

#define LI{B00100000,B00000000,B01100000,B00100000,B00100000,B01111000}

#define LJ{B00010000,B00000000,B00111000,B00010000,B10010000,B01100000}

#define LK{B10000000,B10010000,B10100000,B11000000,B10100000,B10010000}

#define LL{B01100000,B00100000,B00100000,B00100000,B00100000,B01111000}

#define LM{B00000000,B00000000,B11010000,B10101000,B10101000,B10001000}

#define LN{B00000000,B00000000,B10110000,B11001000,B10001000,B10001000}

#define LO{B00000000,B01110000,B10001000,B10001000,B10001000,B01110000}

#define LP{B00000000,B11110000,B10001000,B11110000,B10000000,B10000000}

#define LQ{B00000000,B01101000,B10011000,B01111000,B00001000,B00001000}

#define LR{B00000000,B00000000,B10110000,B11001000,B10000000,B10000000}

#define LS{B00000000,B01110000,B10000000,B01110000,B00001000,B11110000}

#define LT{B01000000,B01000000,B11100000,B01000000,B01001000,B00110000}

#define LU{B00000000,B00000000,B10001000,B10001000,B10011000,B01101000}

#define LV{B00000000,B00000000,B10001000,B10001000,B01010000,B00100000}

#define LW{B00000000,B00000000,B10001000,B10101000,B10101000,B01010000}

#define LX{B00000000,B10001000,B01010000,B00100000,B01010000,B10001000}

#define LY{B00000000,B10001000,B10001000,B01111000,B00001000,B01110000}

#define LZ{B00000000,B11111000,B00010000,B00100000,B01000000,B11111000}

#define SPACE{B00000000,B00000000,B00000000,B00000000,B00000000,B00000000}

#define NUM0{B01110000,B10011000,B10101000,B10101000,B11001000,B01110000}

#define NUM1{B00100000,B01100000,B10100000,B00100000,B00100000,B01110000}

#define NUM2{B01110000,B10001000,B00001000,B01110000,B10000000,B11111000}

#define NUM3{B11110000,B00001000,B00001000,B01111000,B00001000,B11110000}

#define NUM4{B10001000,B10001000,B10001000,B11111000,B00001000,B00001000}

#define NUM5{B11111000,B10000000,B11110000,B00001000,B10001000,B01110000}

#define NUM6{B11111000,B10000000,B11111000,B10001000,B10001000,B11111000}

#define NUM7{B11111000,B00001000,B00001000,B01111000,B00001000,B00001000}

#define NUM8{B11111000,B10001000,B11111000,B10001000,B10001000,B11111000}

#define NUM9{B11111000,B10001000,B11111000,B00001000,B00001000,B11111000}

#define DEVIDE{B00001000,B00010000,B00100000,B00100000,B01000000,B10000000}

#define TWODOTS{B01100000,B01100000,B00000000,B00000000,B01100000,B01100000}

#define DOT{B00000000,B00000000,B00000000,B00000000,B01100000,B01100000}

#define COMA{B00000000,B00000000,B00000000,B00110000,B00110000,B01100000}

#define LINE{B00000000,B00000000,B11111000,B11111000,B00000000,B00000000}

#define QUASTION{B01110000,B10001000,B00010000,B00100000,B00000000,B00100000}

#define MARK{B00100000,B01110000,B01110000,B00100000,B00000000,B00100000}

int latchPin = 10;

int clockPin = 13;

int dataPin = 11;

int clock = 9;

int Reset = 8;

int latchPinPORTB = latchPin - 8;

int clockPinPORTB = clockPin - 8;

int dataPinPORTB = dataPin - 8;

int i = 0;

int incomingByte[44];

long scrolling\_word[6];

int array\_turn=0;

byte patterns[100][6];

byte dummy\_array[70][6] ={BA,BB,BC,BD,BE,BF,BG,BH,BI,BJ,BK,BL,BM,BN,BO,BP,BQ,BR,BS,BT,BU,BV,BW,BX,BY,BZ,SPACE,NUM0,NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,NUM7,NUM8,NUM9,DEVIDE,TWODOTS,DOT,COMA,LINE,QUASTION,MARK,LA,LB,LC,LD,LE,LF,LG,LH,LI,LJ,LK,LL,LM,LN,LO,LP,LQ,LR,LS,LT,LU,LV,LW,LX,LY,LZ};

void setup(){

Serial.begin(9600);

pinMode(dataPin,OUTPUT);

pinMode(clockPin,OUTPUT);

pinMode(latchPin,OUTPUT);

pinMode(clock,OUTPUT);

pinMode(Reset,OUTPUT);

digitalWrite(Reset,HIGH);

digitalWrite(Reset,LOW);

setupSPI();

}

void display\_word(int loops,byte word\_print[][6],int num\_patterns,int delay\_langth){// this function displays your symbols

i = 0;// resets the counter fot the 4017

for(int g=0;g<6;g++)//resets the the long int where your word goes

scrolling\_word[g] = 0;

for(int x=0;x<num\_patterns;x++){//main loop, goes over your symbols

// you will need to find a better way to make the symbols scroll my way is limited for 24 columns

for(int r=0;r<6;r++)//puts the buildes the first symbol

scrolling\_word[r] |= word\_print[x][r];

for (int z=0;z<6;z++){//the sctolling action

for(int p=0;p<6;p++)

scrolling\_word[p] = scrolling\_word[p] << 1;

// end of the scrolling funcion

for(int t=0;t<delay\_langth;t++){// delay function, it just loops over the same display

for(int y=0;y<6;y++){// scaning the display

if(i == 6){// counting up to 6 with the 4017

digitalWrite(Reset,HIGH);

digitalWrite(Reset,LOW);

i = 0;

}

latchOff();

spi\_transfer(make\_word(0x01000000,y));// sending the data

spi\_transfer(make\_word(0x00010000,y));

spi\_transfer(make\_word(0x00000100,y));

latchOn();

delayMicroseconds(800);//waiting a bit

latchOff();

spi\_transfer(0);// clearing the data

spi\_transfer(0);

spi\_transfer(0);

latchOn();

digitalWrite(clock,HIGH);//counting up with the 4017

digitalWrite(clock,LOW);

i++;

}

}

}

}

finish\_scroll(delay\_langth);

}

void finish\_scroll(int delay\_scroll){// this function is the same as the funcion above, it just finishing scrolling

for (int n=0;n<24;n++){

for(int h=0;h<6;h++)

scrolling\_word[h] = scrolling\_word[h] << 1;

for(int w=0;w<delay\_scroll;w++){

for(int k=0;k<6;k++){

if(i == 6){

digitalWrite(Reset,HIGH);

digitalWrite(Reset,LOW);

i = 0;

}

latchOff();

spi\_transfer(make\_word(0x01000000,k));

spi\_transfer(make\_word(0x00010000,k));

spi\_transfer(make\_word(0x00000100,k));

latchOn();

delayMicroseconds(800);

latchOff();

spi\_transfer(0);

spi\_transfer(0);

spi\_transfer(0);

latchOn();

digitalWrite(clock,HIGH);

digitalWrite(clock,LOW);

i++;

}

}

}

}

byte make\_word (long posistion,byte turn){

byte dummy\_word = 0;

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

if(scrolling\_word[turn] & (posistion<<q))

dummy\_word |= 0x01<<q;

}

return dummy\_word;

}

void loop() {

// send data only when you receive data:

if(Serial.available() > 0){

delay(100);

incomingByte[array\_turn] = Serial.read();

array\_turn++;

}

else{

if(array\_turn != 0){

for(int az=0;az<array\_turn;az++){

if((incomingByte[az] > 64 && incomingByte[az] < 91) || (incomingByte[az] > 96 && incomingByte[az] < 123)){

if(incomingByte[az] > 64 && incomingByte[az] < 91){

for(int lal=0;lal<6;lal++)

patterns[az][lal] = dummy\_array[incomingByte[az] - 65][lal];

}

else{

for(int lal=0;lal<6;lal++)

patterns[az][lal] = dummy\_array[incomingByte[az] - 53][lal];

}}

else{

switch(incomingByte[az]){

case 32://space

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[26][lol];

break;

case 33://mark

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[43][lol];

break;

case 45://line

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[41][lol];

break;

case 44://coma

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[40][lol];

break;

case 46://dot

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[39][lol];

break;

case 47://dvide

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[37][lol];

break;

case 48://0

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[27][lol];

break;

case 49://1

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[28][lol];

break;

case 50://2

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[29][lol];

break;

case 51://3

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[30][lol];

break;

case 52://4

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[31][lol];

break;

case 53://5

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[32][lol];

break;

case 54://6

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[33][lol];

break;

case 55://7

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[34][lol];

break;

case 56://8

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[35][lol];

break;

case 57://9

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[36][lol];

break;

case 58://tow dots

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[38][lol];

break;

case 63://quastion

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[42][lol];

break;

default:

for(int lol=0;lol<6;lol++)

patterns[az][lol] = dummy\_array[26][lol];

break;

}

}

}

}

display\_word(1,patterns,array\_turn,15);

array\_turn =0;

}

}

//display\_word(1,patterns,43,15);// calls for the display\_pattern function and says that int loop = 15(if you do more loop the pattern whould scrrol slower).

void latchOn(){

bitSet(PORTB,latchPinPORTB);

}

void latchOff(){

bitClear(PORTB,latchPinPORTB);

}

void setupSPI(){

byte clr;

SPCR |= ( (1<<SPE) | (1<<MSTR) ); // enable SPI as master

//SPCR |= ( (1<<SPR1) | (1<<SPR0) ); // set prescaler bits

SPCR &= ~( (1<<SPR1) | (1<<SPR0) ); // clear prescaler bits

clr=SPSR; // clear SPI status reg

clr=SPDR; // clear SPI data reg

SPSR |= (1<<SPI2X); // set prescaler bits

//SPSR &= ~(1<<SPI2X); // clear prescaler bits

delay(10);

}

byte spi\_transfer(byte data)

{

SPDR = data; // Start the transmission

while (!(SPSR & (1<<SPIF))) // Wait the end of the transmission

{

};

return SPDR; // return the received byte, we don't need that

}