// **PROJECT  :MorlandBargraphV3**

// PURPOSE  :(Primary) Animations to exercise the Morland Bargraph V3 Appliance

//          :(Secondary) Highlights similarity between shiftout and SPI

// DATE     :2019 03 07

// STATUS   :Working

// REFERENCE:http://darcy.rsgc.on.ca/ACES/TEI3M/images/MorlandV3Versatility.png

// NOTES    :Appliance supports shifting on all three UNO ports (B,C,D)

//          :SPI functions as non-appliance due to 5V interference with MISO

//          :'breathe' only functions as 'true' appliance on PORTB (PWM pin 9)

#include <**SPI**.h>

#define PWM 9       //Output Enable for 595s to 'breathe'

#define GND 30      //dummy pin number to support code versatility

#define VCC 31      //dummy pin number to support code versatility

//Comment out all but one of the four below for a demonstration

//uint8\_t port[] = {GND, SCK, MISO, MOSI, SS, PWM}; //PORTB

uint8\_t port[] = {A0,A1,A2,A3,A4,A5};               //PORTC

//uint8\_t port[] = {PD7,PD6,PD5,PD4,PD3,PD2};       //PORTD

//uint8\_t port[] = {GND, SCK, VCC, MOSI, SS, PWM};  //SPI(GND and 5V to supply)

#define groundPin port[0]         //activate for placement other than PORTB

#define clockPin port[1]          //digital pin 13 for SPI compatibility

#define powerPin port[2]          //

#define dataPin port[3]           //digital pin 11 for SPI compatibility

#define latchPin port[4]          //digital pin 10 for SPI compatibility

#define enablePin port[5]         //digital pin 9 PWM for Output Enable

//0-flash,1-breathe,2-L2R,3-R2L,4-fastFlash,5-count,10-SPIFlash,11-SPIBreathe

#define animation 0

#define DELAYMS 128               //base for pause interval

uint8\_t n;                        //byte variable to be used for demonstration

void setup() {

  pinMode(groundPin, OUTPUT);     //All 6 pins to the MB3 declared for output

  pinMode(clockPin, OUTPUT);      //

  pinMode(powerPin, OUTPUT);      //not the best, but it'll do...

  pinMode(dataPin, OUTPUT);       //

  pinMode(latchPin, OUTPUT);      //

  pinMode(enablePin, OUTPUT);     //

  digitalWrite(groundPin, LOW);   //set required signal levels

  digitalWrite(powerPin, HIGH);   //

  digitalWrite(enablePin, LOW);   //

}

void loop() {

  switch (animation) {

    case 0: flash(); break;       //software shiftOut...

    case 1: breathe(); break;     //software shiftOut...

    case 2: L2R(); break;         //software shiftOut...

    case 3: R2L(); break;         //software shiftOut...

    case 4: fastFlash(); break;   //software shiftOut...

    case 5: count(); break;       //software shiftOut...

    case 10: SPIFlash(); break;   //hardware SPI...

    default: SPIBreathe();        //hardware SPI...

   }

}

void flash() {

  n = 255;                                      //set all bits in the data byte

  while (true) {                                //loop forever

    digitalWrite(latchPin, LOW);                //pull latch LOW

    shiftOut(dataPin, clockPin, MSBFIRST, n);   //upload byte data

    digitalWrite(latchPin, HIGH);               //present bits on output pins

    delay(DELAYMS);                             //admire...

    n = ~n;                                     //complement (invert) all bits

  }

}

void breathe() {

  digitalWrite(latchPin, LOW);                  //pull latch LOW

  shiftOut(dataPin, clockPin, MSBFIRST, 255);   //upload byte data

  digitalWrite(latchPin, HIGH);                 //present bits on output pins

  int8\_t delta = 1;

  n = 1;

  while (true) {                                //loop forever

    analogWrite(enablePin, n);                  //PWM the 595 OE pin

    delay(DELAYMS >> 4);                        //admire...

    delta = (n == 255) | (n == 0) ? -delta : delta;   //reverse direction?

    n += delta;                                 //next...

  }

}

void L2R() {

  n = 128;

  while (true) {

    digitalWrite(latchPin, LOW);                //pull latch LOW

    shiftOut(dataPin, clockPin, MSBFIRST, n);   //upload byte data

    digitalWrite(latchPin, HIGH);               //present bits on output pins

    delay(DELAYMS);                             //admire...

    n = n == 1 ? 128 : n >> 1;                  //shift right or start again

  }

}

void R2L() {

  n = 1;

  while (true) {                                //loop forever

    digitalWrite(latchPin, LOW);                //pull latch LOW

    shiftOut(dataPin, clockPin, MSBFIRST, n);   //upload byte data

    digitalWrite(latchPin, HIGH);               //present bits on output pins

    delay(DELAYMS);                             //admire...

    n = n == 128 ? 1 : n << 1;                  //shift left or start again

  }

}

void fastFlash() {

  digitalWrite(latchPin, LOW);                  //pull latch LOW

  shiftOut(dataPin, clockPin, MSBFIRST, 255);   //upload byte data

  digitalWrite(latchPin, HIGH);                 //present bits on output pins

  while (true) {                                //loop forever

    digitalWrite(enablePin, LOW);               //toggle the 595 OE pin

    delay(DELAYMS);                             //admire...

    digitalWrite(enablePin, HIGH);              //

    delay(DELAYMS);                             //admire...

  }

}

void count() {

  n = 0;

  while (true) {                                //loop forever...

    digitalWrite(latchPin, LOW);                //pull latch LOW

    shiftOut(dataPin, clockPin, MSBFIRST, n);   //upload byte data

    digitalWrite(latchPin, HIGH);               //present bits on output pins

    delay(DELAYMS >> 1);                        //admire...

    n++;                                        //next...

  }

}

void SPIFlash() {

  n = 0xAA;                               //confirm MB3 compatibility for hardware SPI

  **SPI**.begin();                            //open an SPI session

  while (true) {                          //loop forever...

    digitalWrite(SS, LOW);                //pull SS pin (aka latch) LOW

    **SPI**.transfer(n);                      //similar to shiftout

    digitalWrite(SS, HIGH);               //restore high on SS (slaveselect)

    delay(DELAYMS << 1);                  //admire

    n = ~n;                               //complement (invert) all the bits

  }

  **SPI**.end();                              //end the SPI session (unreachable)

}

void SPIBreathe() {

  n = 0xFF;                               //set all bits in the data byte

  **SPI**.begin();                            //open an SPI session

  digitalWrite(SS, LOW);                  //pull the slaveselect pin LOW

  **SPI**.transfer(n);                        //swap byte with the MB3 (slave)

  digitalWrite(SS, HIGH);                 //restore slaveselect pin to HIGH

  **SPI**.end();                              //end the SPI session

  int8\_t delta = 1;

  n = 1;

  while (true) {                          //loop forever...

    analogWrite(enablePin, n);            //PWM the 595 OE pin for ‘breathing’

    delay(DELAYMS >> 4);                  //admire...

    delta = (n == 255) | (n == 0) ? -delta : delta; //time to change direction?

    n += delta;                           //next...

  }}

