// PROJECT  :MBV3Shiftout

// PURPOSE  :Demonstration of the Shiftout function on the MBV3

// DEVICE   :MBV3 on a Nano

// AUTHOR   :C. D'Arcy

// DATE     :2023 01 19

// uC       :328

// COURSE   :ICS3U

// STATUS   :Working

// REFERENCE:http://darcy.rsgc.on.ca/ACES/TEI3M/Fritzing/MBV3Prototype.png

// REFERENCE:https://www.asciitable.com/

// SEE      :MBV3SPI.ino

// NOTES    :The example uses Shiftout to echo the ASCII value of characters

//          :into the Serial Input box onto the Serial Monitor and Morland Bargraph

#define CLOCK   SCK       // 328P pin 13  (exploit #predefines for added clarity)

#define DATA    MOSI      // 328P pin 11

#define LATCH   SS        // 328P pin 10

#define BAUD    9600      // Serial communication rate

#define PAUSE   1000      // pacing between display of multiple characters

#define NEWLINE 10        // ASCII Control Character

void setup() {

  pinMode(CLOCK, OUTPUT);

  pinMode(DATA, OUTPUT);

  pinMode(LATCH, OUTPUT);

  **Serial**.begin(BAUD);

  while (!**Serial**);

  **Serial**.println("Enter one or more characters into the Serial Input box...");

}

void loop() {

  // Are there characters in the Serial input buffer to be read???

  if (**Serial**.available()) {         // if so...

    char ch = **Serial**.read();        // ...read the first one (oldest)

    digitalWrite(LATCH, LOW);       // shift out to the MBV3...note the shift order

    shiftOut(DATA, CLOCK, MSBFIRST, ch);

    digitalWrite(LATCH, HIGH);

    **Serial**.write(ch);               // echo to the Serial Monitor (write)

    **Serial**.write(NEWLINE);          // go down to the next line

    delay(PAUSE);                   // admire...

  }

}

// PROJECT  :MBV3SPI

// PURPOSE  :Demonstration of SPI to the MBV3

// DEVICE   :MBV3 on a Nano

// AUTHOR   :C. D'Arcy

// DATE     :2023 01 19

// uC       :328

// COURSE   :ICS3U

// STATUS   :Working

// REFERENCE:http://darcy.rsgc.on.ca/ACES/TEI3M/Fritzing/MBV3Prototype.png

// REFERENCE:https://www.asciitable.com/

// SEE      :MBV3Shiftout.ino

// NOTES    :The example uses SPI to echo the ASCII value of the characters entered

//          :into the Serial Input box onto the Serial Monitor and Morland Bargraph

#include <**SPI**.h>          // library support

#define BAUD    9600      // Serial communication rate

#define PAUSE   1000      // pacing between display of multiple characters

#define NEWLINE 10        // ASCII Control Character

void setup() {

  pinMode(SCK, OUTPUT);   //

  pinMode(MOSI, OUTPUT);  //

  pinMode(SS, OUTPUT);    //

  **Serial**.begin(BAUD);

  while (!**Serial**);

  **Serial**.println("Enter one or more characters into the Serial Input box...");

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

}

void loop() {

  // Are there characters in the Serial input buffer to be read???

  if (**Serial**.available()) {         // if so...

    char ch = **Serial**.read();        // ...read the first one (oldest)

    digitalWrite(SS, LOW);          // identify the target device

    **SPI**.transfer(ch);               // transmit a byte

    digitalWrite(SS, HIGH);         // release the target device

    **Serial**.write(ch);               // echo to the Serial Monitor (write)

    **Serial**.write(NEWLINE);          // go down to the next line

    delay(PAUSE);                   // admire...

  }

}

// PROJECT  :SPIvsShiftOutV2

// PURPOSE  :Demonstrates similarity between ShiftOut (software) vs SPI (hardware)

// COURSE   :ICS3U

// AUTHOR   :C. D'Arcy

// DATE     :2022 04 14

// MCU      :328P

// STATUS   :Working

// REFERENCE:Hardware prototype to be as depicted on the right side of this image:

//          :[http://darcy.rsgc.on.ca/ACES/TEI3M/images/MorlandV3Versatility.png](http://darcy.rsgc.on.ca/ACES/TEI3M/images/MorlandV3Versatility.png%20)

// NOTES    :Introduces conditional compiler directives(#if, #else, #endif, etc.)

//          : to customize (minimize) code footprint

#define HARDWARE\_SHIFT false

#if HARDWARE\_SHIFT

#include <**SPI**.h>

#endif

#define DURATION  500

uint8\_t i = 1;

void setup() {

  pinMode(SS, OUTPUT);

#if HARDWARE\_SHIFT

  **SPI**.begin();

#else

  pinMode(SCK, OUTPUT);

  pinMode(MISO, OUTPUT);

  pinMode(MOSI, OUTPUT);

#endif

}

void loop() {

  digitalWrite(SS, LOW);              //select the target device

#if HARDWARE\_SHIFT                    //apply the desired transfer protocol...

  **SPI**.transfer(i);                    // hardware? (faster)

 #else

  shiftOut(MOSI, SCK, MSBFIRST, i);   // software? (slower, but more flexible?)

#endif

  digitalWrite(SS, HIGH);             //complete the transfer: release the device

  delay(DURATION);                    //pause and admire....

  i = i ? i << 1 : 1;                 //continue

}

// PROJECT  :MBv3SPIBus4Peris

// PURPOSE  :Sample Demonstration L-to-R 1D Scrolling of an SPI Bus

//          :on a bank of 4 MBv3s ('595s)...

//          :Updated version of 2020 MBv3SPIBus.ino of 3 MBv3 Peripherals

// COURSE   :ICS3U

// DATE     :2022 12 27

// STATUS   :Working!

// PHOTO    :http://darcy.rsgc.on.ca/ACES/TEI3M/SPICommunication/images/MBV3SPIBus4.jpg

#include <**SPI**.h>                     //

#include <**TimerOne**.h>                //timed animation

#define FRAMEDELAY 50000             //

#define PER3 5                        // convenient nano access to multiple MBv3s

#define PER2 4                        // "

#define PER1 3                        // "

#define PER0 2                        // "

uint8\_t pers[] = {PER0, PER1, PER2, PER3};  // pack 'em in an array for ease of use

uint8\_t numPers = sizeof(pers);       // how many perpiherals on the Bus?

uint64\_t sprite = 0x3;                // image to be animated across multiple MBv3s

uint64\_t n = 0;                       // current state of sprite data...

volatile boolean triggered = false;   //

void setup() {

  for (uint8\_t i = 0; i < numPers; i++)

    pinMode(pers[i], OUTPUT);

  n = sprite << (numPers << 3);       // sprite placed out of view, stage left

  **SPI**.begin();                        // start the SPI session  ...

  Timer1.initialize(FRAMEDELAY);

  Timer1.attachInterrupt(ISR\_UpdateSprite);

}

void ISR\_UpdateSprite() {

  triggered = true;

}

void loop() {

  for (uint8\_t i = 0; i < numPers; i++) {     // load the frame in this loop

    digitalWrite(pers[i], LOW);               // address the Bus peripheral ...

    **SPI**.transfer((uint8\_t)(n >> (i << 3)));   // ... load with its byte segment ...

    digitalWrite(pers[i], HIGH);              // ... release the peripheral

  }

  if (triggered) {                            // does frame require updating?

    triggered = false;                        // prepare for next timer interrupt

    n = n ? n >> 1 : sprite << (numPers << 3);// update/restore frame data

  }

}