// PROJECT  :SPIShiftOut

// PURPOSE  :To highlight the overlap between hardware SPI & software shiftOut

//          :by shifting out a single byte to a Morland Bargraph

// DEVICE   :Arduino UNO + Morland Bargraph

// AUTHOR   :C. D'Arcy

// DATE     :2019 01 31

// uC       :328p

// COURSE   :ICS3U/ICS4U

// STATUS   :Working

// REFERENCE:<http://darcy.rsgc.on.ca/ACES/TEI3M/images/SPIConcept.png>

//<http://darcy.rsgc.on.ca/ACES/TEI3M/SPICommunication/images/SPIvsShiftOut.png>

// NOTES    :This code highlights and contrasts the difference between the use

//          :of the software shiftOut function and the hardware SPI peripheral

//          :on the UNO. They are strikingly similar but each has their own

//          :advantages and disadvantages depending on your application.

#include <**SPI**.h>

#define VALUE 0x03 //sample data to for confirmation and comparison

void setup() {

  **Serial**.begin(9600); //Useful for time benchmarks to compare speeds

  //comment out one or the other...

  softwareShiftOut();

  //hardwareShiftOut();

}

void softwareShiftOut() {

  pinMode(SCK, OUTPUT);     //System Clock (pin 13) (exploit predefines:)

  pinMode(MOSI, OUTPUT);    //MOSI(11). MISO(12) not needed in this example

  pinMode(SS, OUTPUT);      //Slave Select (10)

  digitalWrite(SS, LOW);

  shiftOut(MOSI, SCK, LSBFIRST, VALUE); //No control over transfer parameters

  digitalWrite(SS, HIGH);

}

void hardwareShiftOut() {

  //Initializes the SPI bus setting SCK, MOSI, and SS to outputs,

  **SPI**.begin();              //pull SCK and MOSI low, SS high. Default: MSBFIRST

  //SPI.beginTransaction(SPISettings(14000000, LSBFIRST, SPI\_MODE0));//optional

  digitalWrite(SS, LOW); //

  **SPI**.transfer(VALUE); //

  digitalWrite(SS, HIGH); //

  **SPI**.end(); //disables SPI Bus (leaving pin modes unchanged)

  //SPI.endTransaction(); //optional (use with SPI.beginTransaction above)

}

 void loop() {}

// PROJECT  :SPIvsShiftOut

// PURPOSE  :A slight code improvement from previous SPIShiftOut sketch

// DEVICE   :Arduino + Morland Bargraph

// AUTHOR   :C. D'Arcy

// DATE     :2019 02 02

// uC       :328/84/85

// COURSE   :ICS3U/ICS4U

// STATUS   :Working

// REFERENCE:

// <https://shift-register.org/2017/spi-vs-shiftout-on-arduino-and-attiny>

// NOTES :Test program to shift out some bits to measure the difference // // :between different Atmel chips and SPI vs. shiftOut()

#include <**SPI**.h>

#define RCLK\_PIN SS // latching pin for shift registers to show values

#define DATA\_PIN MOSI

#define CLK\_PIN  SCK

// pick one of these

//#define SHIFTMODE

#define SPIMODE

void setup() {

  pinMode(RCLK\_PIN, OUTPUT); //

#ifdef SHIFTMODE

  pinMode(DATA\_PIN, OUTPUT); //

  pinMode(CLK\_PIN, OUTPUT); //

  shiftOut(DATA\_PIN, CLK\_PIN, LSBFIRST, 0);

#endif

#ifdef SPIMODE

  **SPI**.begin(); //

  **SPI**.beginTransaction(SPISettings(16000000, LSBFIRST, SPI\_MODE0));

#endif

}

uint8\_t num = 0x00;

void loop() {

  digitalWrite(RCLK\_PIN, LOW); //

#ifdef SHIFTMODE //

  shiftOut(DATA\_PIN, CLK\_PIN, LSBFIRST, num);

#endif

#ifdef SPIMODE

  **SPI**.transfer(num); //

#endif

  digitalWrite(RCLK\_PIN, HIGH); //

  num++;

  delay(20);

}

// PROJECT  :MCP4231

// PURPOSE  :Demonstrates SPI Library manipulation of MCP4231 Digital Pot

// DEVICE   :Arduino + MCP4231 Digital Pot + DMM set to Voltage

// AUTHOR   :C. D'Arcy

// DATE     :2019 02 06

// uC       :328p

// COURSE   :ICS3U/ICS4U

// STATUS   :Working

// REFERENCE:<https://mail.rsgc.on.ca/~cdarcy/Datasheets/MCP4231.pdf>

// REFERENCE:<https://www.arduino.cc/en/Reference/SPISettings>

// NOTES    :MCP4231 Datasheet: up to 10MHz SCK

//          :Connect a DMM to the P0W Pin to watch potential (DCV) change

#include <**SPI**.h>            // include the SPI library of course

SPISettings mySettings(10000000, MSBFIRST, SPI\_MODE0);

//------------MCP4231 Parameters------------------------------

#define WRITE B00000000     //Set Command (MOSI)

#define INCR  B00000100     //Increment Command

#define DECR  B00001000     //Decrement Command

#define READ  B00001100     //Get Command (MISO)

#define REG0  B00000000     //Register 0 Write command

#define REG1  B00010000     //Register 1 Write command

#define LIMIT 127           //Maximum value for (7-bit) MCP4231

//------------Code Variables----------------------------------

uint8\_t data;               //value written to the MCP4231

uint8\_t res;                //value read back from the MCP4231

int8\_t  delta = -1;         //amount to change the value written (up/down)

void setup() {

  **Serial**.begin(9600);       //Serial monitoring useful

  **SPI**.begin();              //Initialize SPI (sets SCK, MISO & MOSI for output)

  **SPI**.beginTransaction(mySettings);     //(optional) tailoring of SPI session settings

  pinMode(SS, OUTPUT);      //set the Slave Select pin (to /CS pin of MCP4231) as an output

}

void loop() {

  SPIWrite(SS, REG0 | WRITE, data);       //sets and gets a new value

  **Serial**.print("SET:" + String(data));    //display the value written...

  res = SPIRead(SS, REG0 | READ);         //let's read it back to confirm...

  **Serial**.println("\tGET:" + String(res)); //display it for confirmation

  delay(64);                              //hmmmm...not to good...

  delta = (data == 0 || data == LIMIT) ? -delta : delta;  // change direction?

  data += delta;                          //update value to be written

}

void SPIWrite(uint8\_t slave, uint8\_t command, uint8\_t data) {

  digitalWrite(slave, LOW);   //Set the provided SS pin low

  **SPI**.transfer(command);      //Choose the register to write to

  **SPI**.transfer(data);         //Set the LED level [0,127]

  digitalWrite(slave, HIGH);  //Set the provided SS pin high again

}

uint8\_t SPIRead(uint8\_t slave, uint8\_t command){

  digitalWrite(slave, LOW);       //Set the provided SS pin low

  **SPI**.transfer(command);          //Choose the register to write to

  uint8\_t res = **SPI**.transfer(0);  //Send a dummy value to force a return

  digitalWrite(slave, HIGH);      //Set the provided SS pin high again

  return res;                     //Send the vlaue back to the caller

}

