# Arduino

## ez Serial/Parallel IC Library (SIPO8) ~ Tutorial 5

A Tutorial to Consolidate Understanding
& Use of the ez Serial/Parallel IC Library
(SIPO8)

Tutorial 5
~ Bank Interleaving ~

Author: R D Bentley, Stafford, UK.
Date: May 2021
Version: 1.00

## Warranties & Exceptions



## Change Record

| Version | Date | Change |
|---|---|---|
| 1.00 | May 2021 | Initial version published |
| | | |
| | | |

## Audience

Whilst it is not necessary to and be familiar with 8bit serial/parallel ICs, such as the 74HC595 IC, some understanding with wiring these ICs and driving them with suitable Arduino code, for example the standard Arduino `shiftOut` function, would provide an excellent primer on which to build and develop sophisticated and innovative solutions through the use of the `<ez_SIPO8_lib>` library.

## Contents

# Introduction to the Tutorial

In this tutorial we will look at bank interleaving. Bank interleaving allows us to define a number of identicaly sized SIPO banks but all sharing the same 3-wire digital pin microcontroller interface. This provides us with the means to maintain SIPO banks with different SIPO output pin configurations and xfer them to the same physical SIPO ICs, singly or s a multiple cascade, as our project dictates.

(If you have not already done so, download the User Guide from github.)

# Objectives

In this tutorial we shall concern ourselves with creating a virtual software SIPO environment in which we can explore/learn:

1. (if the basic test harness is not already to hand from previous tutorials,) how we can wire up and connect a single 74HC595 IC (SIPO) to a microcontroller, to test out some of the features covered by the tutorial
2. how we can create multiple SIPO banks, each with a different output pin status configuration/profile
3. how we can use the technique of bank interleaving to good effect, in this tutorial we shall revisit the User Guide sketch example of strobing LEDs
4. witness the outputs of the sketch configuration – strobing and sketch data structures/values

# Steps

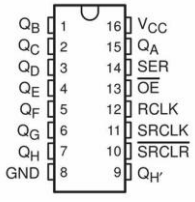
## Kit List

Gather together the following components:

| Components | Number | Comment |
|---|---|---|
| Arduino UNO | 1 | The design uses an Arduino UNO, but any suitable microcontroller (Arduino or clone, e.g. Elegoo) will do, providing it is able to support the pin out requirements for drving the SIPO interface and power requirements |
| Breadboard | 1 | Small or large, whatever you have to hand |
| 74HC595 IC | 1 | 8bit SIPO IC, or other clone providing it is genuinely 'plug-compatible' |
| LEDs | 8 | Whatever you have around |
| Restistors, 220 ohm | 8 | One per LED. Use 220 ohm resisters and ignore the suggested 180 ohm values in the wiring diagrams |
| Connecting wires | Lots | Short/long or breadboard wire connectors, whatever suits |

## 74HC595 Orientation and Pin Outs

Which end is which? Well, notice that the 74HC595 has a notch at one end, here at the top of the diagram. Pin numbering starts at 1 at the top left and continues down the left hand side and the around the bottom of the IC rising to the top right hand side to pin 16:




Pin Outs 1 - 74HC595

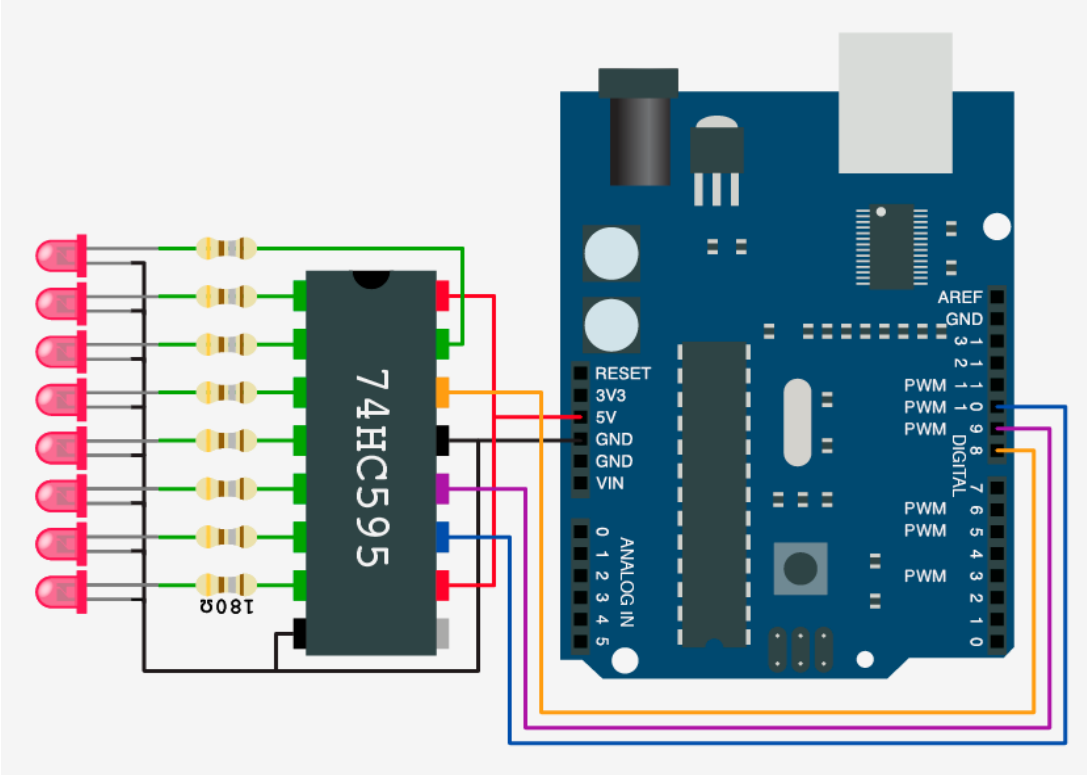
## Microcontroller / SIPO Interface Pin Configuration

Every bank of SIPO ICs you connect to the microcontroller requires a 3-wire digital interface. The table below suggests pin mapping between the microcontroller and the SIPO IC for this tutorial, but you may choose what microcontroller digital pins you wish. If you do choose different pins then be sure to alter the sketch `create_bank` call.

| UNO Pin | SIPO Pin(s) | Comment |
|---|---|---|
| 8 | 14 | SIPO Data Pin |
| 9 | 12 | SIPO Latch Pin |
| 10 | 11 | SIPO Clock Pin |
| +5v | 10, 16 | Power to the SIPO |
| GND | 8, 13 | Return ground (0v) |

## Connecting It All Together

Using the following diagram, wire up all of the components, taking care to get the output/input connections correct:




Wiring Scheme 1 - Single SIPO IC, 8 outputs

Notice that the only 74HC595 IC pin not to have anything connected is pin 9, $Q_H'$. This is used as the serial output pin to connect to the serial input pin, 14 SER, of the next SIPO 74HC595 in a cascade.

## The Code/Sketch

Okay, now let's look at a different version of the sketch we saw in the User Guide which strobed a single SIPO IC back and forth. In this version we shall use bank interleaving such that:

1. we will create eight single SIPO banks
2. each bank will be configured with a single pin, set `HIGH`, in a position representing the bank id. For example, bank id 0 will have pin 0 set `High`, all other pins in the bank being set `LOW` (relative addressing), bank id 1 will have pin 1 set `High`, all other pins set `LOW`, etc., thus:

```
bank id 0       0b00000001
bank id 1       0b00000010
bank id 2       0b00000100
bank id 3       0b00001000
bank id 4       0b00010000
bank id 5       0b00100000
bank id 6       0b01000000
bank id 7       0b10000000
```

3. we will simply xfer each bank in sequence to produce our strobing effect, starting at bank id 0 and at the end of each complete cycle of eight xfers, switch the direction of the shift out, thereby achieving a forward and reverse strobe
4. witness the outputs of the sketch – strobing LED patterns and serial monitor SIPO data

Using the Arduino IDE, start with a new sketch and enter the following (download from github):

```
//   Tutorial 5 - use of ez_SPI8 library,
//   Demonstration of SIPO bank interleaving - 8 x physical SIPOs
//   each mapped to an individual bank but with the same 3-wire digital
//   pin microcontroller interface.
//
//   The sketch sets up each single SIPO bank with a different binary 8bit
//   (8 pin) pattern which is then xferred to the physical single SIPO IC.
//   The sketch mimics the strobe sketch by using bank interleaving.
//   To note is the small amount of SIPO8 library code that is used.
//
//   Ron D Bentley, Stafford, UK
//   April 2021
//
//   This example and code is in the public domain and
//   may be used without restriction and without warranty.
//

#include <ez_SIPO8_lib.h>

int bank_id;

#define Num_SIPOs   8
#define Num_timers  0

SIPO8 my_SIPOs(Num_SIPOs, Num_timers); // initiate the class for the tutorial

uint8_t bank_ids[Num_SIPOs]; // one bank_id per SIPO bank
uint8_t bank;

void setup() {
  Serial.begin(9600);
```

```
  // create banks of 1 x SIPO, all of same 3-wire interface and initialise
  // with each strobe pattern - 0b00000001, 0b00000010, 0b00000100, etc.
  // create_bank params are: data pin, clock pin, latch pin, number of SIPOs this
bank
  for (bank = 0; bank < Num_SIPOs; bank++) {
    bank_ids[bank] = my_SIPOs.create_bank(8, 10, 9, 1);
    if (bank_ids[bank] == create_bank_failure) {
      Serial.println(F("failed to create bank"));
      Serial.flush();
      exit(0);
    }
    // now set up the strobe patterns in the bank's single SIPO, relative SIPO
address is 0
    // sliding pattern of 1's starting at 0b00000001
    my_SIPOs.set_bank_SIPO(bank_ids[bank], 0, 1 << bank); // set up this bank's
strobe pattern
  }
  // print the bank data and pin statuses for confirmation/inspection
  my_SIPOs.print_SIPO_data();
  my_SIPOs.print_pin_statuses();
}

void loop() {
  // scroll through every SIPO bank (interleave) and xfer the bank's pins
  // according to the direction for shift out.
  bool msb_or_lsb = MSBFIRST; // starting direction
  do {
    for (bank = 0; bank < Num_SIPOs; bank++) {
      my_SIPOs.xfer_bank(bank, msb_or_lsb); // xfer out this bank's SIPO pins
      delay(50);
    }
    msb_or_lsb = !msb_or_lsb; // switch direction
  } while (true);
}
```

Note:

1. the eight banks we created mapping to the same physical SIPO IC consumes just eight bytes of memory
2. the actual code performing the strobe effect is very succinct and uses just one SIPO8 library functions – `xfer_bank`
3. the sketch may be easily modified to vary the pattern of set pins in the eight SIPO banks to be any combination
4. which method is best – the above interleaved sketch or the version shown in the User Guide? You judge, but for me the bank interleaved approach does has a certain degree of elegance about it
5. check out the serial monitor and the data the sketch produces – note the bank data which shows that each bank does have the same 3-wire interface but has its own unique range of SIPO output pins allocated from the SIPO array pool. Notice also confirmation of the bit/pin patterns for each bank (pin status byte values – one for each bank).

Check the serial monitor, it should look like this:

Bank data –

```
SIPO global values:
pins_per_SIPO   = 8
max_SIPOs       = 8
bank_SIPO_count = 8
num_active_pins = 64
num_pin_status_bytes = 8
next_free bank =  all SIPOs used
Number timers  =  0
```

```
Bank data:
bank = 0
  num SIPOs = 1
  latch_pin = 9  clock_pin =      10  data_pin  =     8
  low_pin   = 0  high_pin  =      7
bank = 1
  num SIPOs = 1
  latch_pin = 9  clock_pin =      10  data_pin  =     8
  low_pin   = 8  high_pin  =      15
bank = 2
  num SIPOs = 1
  latch_pin = 9  clock_pin =      10  data_pin  =     8
  low_pin   = 16  high_pin  =     23
bank = 3
  num SIPOs = 1
  latch_pin = 9  clock_pin =      10  data_pin  =     8
  low_pin   = 24  high_pin  =     31
bank = 4
  num SIPOs = 1
  latch_pin = 9  clock_pin =      10  data_pin  =     8
  low_pin   = 32  high_pin  =     39
bank = 5
  num SIPOs = 1
  latch_pin = 9  clock_pin =      10  data_pin  =     8
  low_pin   = 40  high_pin  =     47
bank = 6
  num SIPOs = 1
  latch_pin = 9  clock_pin =      10  data_pin  =     8
  low_pin   = 48  high_pin  =     55
bank = 7
  num SIPOs = 1
  latch_pin = 9  clock_pin =      10  data_pin  =     8
  low_pin   = 56  high_pin  =     63
```

Pin status byte data –

```
Active pin array, pin statuses:
Bank  0: MS00000001LS
Bank  1: MS00000010LS
Bank  2: MS00000100LS
Bank  3: MS00001000LS
Bank  4: MS00010000LS
Bank  5: MS00100000LS
Bank  6: MS01000000LS
Bank  7: MS10000000LS
```

Before finishing this tutorial:

- have a go at varying the SIPO bit/pin patterns to get different effects.

- Q. what would be the result if the `my_SIPOs.xfer_bank(bank, msb_or_lsb)` call in the main loop was replaced with `my_SIPOs.xfer_banks(bank, bank, msb_or_lsb)` or `my_SIPOs.xfer_array(msb_or_lsb)`?

  A. `my_SIPOs.xfer_banks(bank, bank, msb_or_lsb)` will work just fine because the same single bank is being shifted out. However, `my_SIPOs.xfer_array(msb_or_lsb)` will not work because this function xfers all defined banks in one go – not what we want.

That is the end of Tutorial 5, which I hope you found instructional.