# Arduino

## ez Serial/Parallel IC Library (SIPO8) ~ Tutorial 1

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

Tutorial 1
~ Setup and Absolute Addressing ~

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

## Warranties & Exceptions



## Change Record

| Version | Date | Change |
|---|---|---|
| 1.00 | April 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 first tutorial we will look at configuring and driving a single 8bit SIPO IC, the 74HC595 IC, which provides us with eight outputs to which we will connect LEDs, one to each SIPO output pin.

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

# Objectives

We shall concern ourselves with creating a virtual software SIPO environment in which we can apply some basic principles, starting small. In particular we shall learn:

1. how we can wire up and connect a single 74HC595 IC (SIPO IC) to a microcontroller
2. the structure of an Arduino sketch suitably configured to drive the physically connected SIPO IC and its LEDs, in particular:
   a. how to create a SIPO8 library class instance
   b. how to create a SIPO bank from an array pool of SIPO output pins, thereby making the SIPO output pins individually addressable by software
   c. how to use absolute SIPO output pin addressing at array level
3. witness the outputs of the sketch – shifting LED patterns and serial monitor SIPO data

# Steps

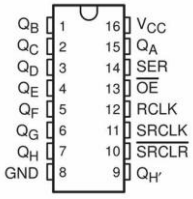
## Kit List

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

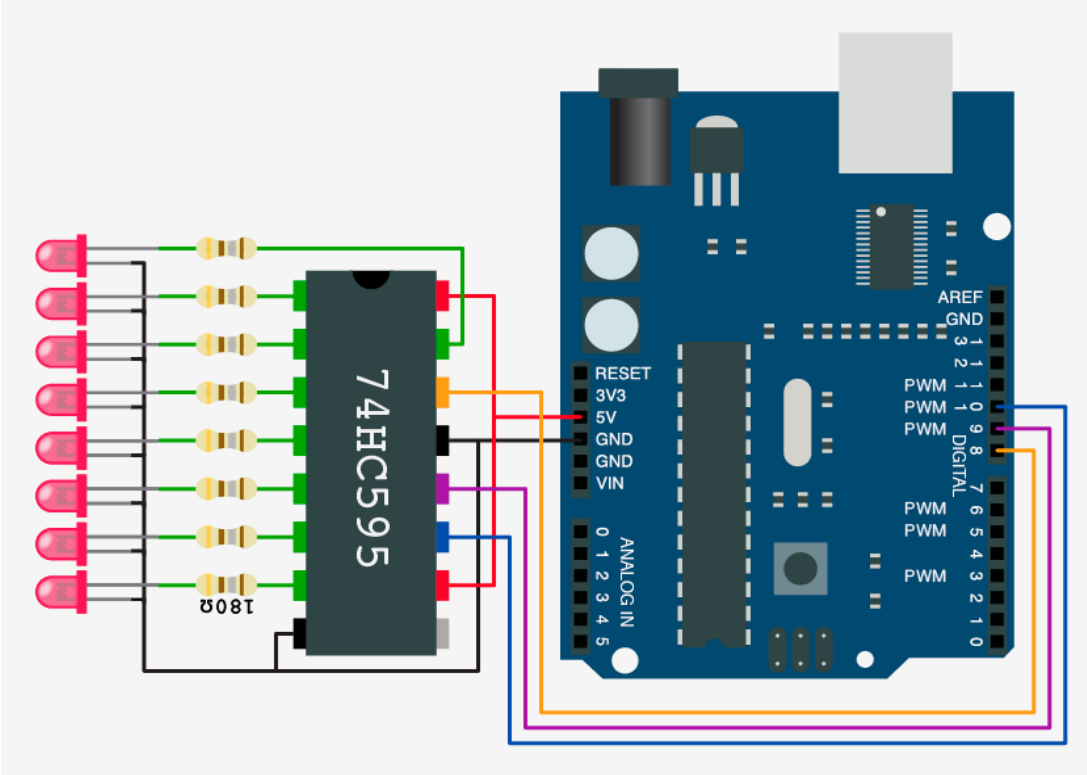
## Microcontroller / SIPO Interface Pin Configuration

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

Now that is done, let's look at a simple sketch to play around with the 75HC595. This sketch will create a single SIPO IC bank of output pins, these having absolute virtual addresses from 0 to 7, once the bank is created using the `create_bank` function.

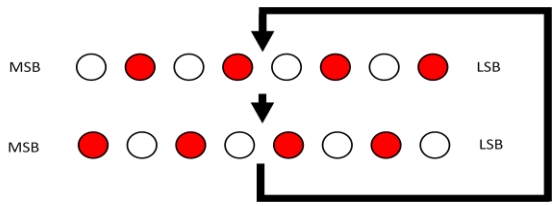
Using the Arduino IDE, start with a new sketch and enter the following (download from github):

```
//
//   Tutorial 1 - use of ez_SPI8 library
//   Setup and absolute addressing.
//
//   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>

#define max_SIPOs  1  // one 1 SIPO for this tutorial
#define max_timers 0  // no timers required

// initiate the class for max SIPOs/timers required
SIPO8 my_SIPOs(max_SIPOs, max_timers);

int my_LEDs;  // used to keep the SIPO bank id

void setup() {
  Serial.begin(9600);
  // create a bank of 1 SIPO using create_bank function:
  // data pin, clock pin, latch pin, number of SIPO this bank
  my_LEDs = my_SIPOs.create_bank(8, 10, 9, 1);
  if (my_LEDs == create_bank_failure) {
    Serial.println(F("\nfailed to create bank"));
    Serial.flush();
    exit(0);
  }
  // print the bank data for confirmation/inspection
  my_SIPOs.print_SIPO_data();
}

void loop() {
    // start by setting all SIPO outputs to low (off)
    my_SIPOs.set_all_array_pins(LOW);
    my_SIPOs.xfer_array(LSBFIRST);
  do {
    //
    // assemble pattern 0b01010101 into the array
    my_SIPOs.set_array_pin(0, HIGH);
    my_SIPOs.set_array_pin(1, LOW);
    my_SIPOs.set_array_pin(2, HIGH);
    my_SIPOs.set_array_pin(3, LOW);
    my_SIPOs.set_array_pin(4, HIGH);
    my_SIPOs.set_array_pin(5, LOW);
    my_SIPOs.set_array_pin(6, HIGH);
    my_SIPOs.set_array_pin(7, LOW);
    // now move array to physical SIPO and light up the LEDs
    my_SIPOs.xfer_array(MSBFIRST);
```

```
    delay(500);
    //
    // assemble inverted pattern 0b10101010 into the array
    my_SIPOs.set_array_pin(0, LOW);
    my_SIPOs.set_array_pin(1, HIGH);
    my_SIPOs.set_array_pin(2, LOW);
    my_SIPOs.set_array_pin(3, HIGH);
    my_SIPOs.set_array_pin(4, LOW);
    my_SIPOs.set_array_pin(5, HIGH);
    my_SIPOs.set_array_pin(6, LOW);
    my_SIPOs.set_array_pin(7, HIGH);
    // now move array to physical SIPO and light up the LEDs
    my_SIPOs.xfer_array(MSBFIRST);
    delay(500);
  } while (true);
}
```

(all SIPO8 functions/methods are shown against a darker grey background)

Compile the sketch and upload. You should now see the LEDs 'see-sawing' back and forth with a small delay between cycles:




The sketch uses the absolute addressing and functions `set_array_pin` and `xfer_array`. The pin address used by `set_array_pin` is an absolute pin address, starting at 0. Any virtual active output pin may be referenced by its absolute address in the array pool. However, virtual output pins are only active when they are assigned to a bank via the `create_bank` function – this brings them into 'life' in blocks of eight. Note that absolute addressing is different to relative addressing which is covered in tutorial 2.

Note the significance of the `MSBFIRST` parameter in the `xfer` calls and how the physical outputs reflect the virtual pin settings with this parameter. Discover what would happen if you change the `MSBFIRST` parameter to `LSBFIRST` in the two `my_SIPO.xfer_array` calls.

Finally, check the serial monitor, it should look like this:

```
SIPO global values:
pins_per_SIPO   = 8
max_SIPOs       = 1
bank_SIPO_count = 1
num_active_pins = 8
num_pin_status_bytes = 1
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
```

Note the helpful details that `print_SIPO_data()` function can provide – very useful during development and debugging.

Have a play around varying SIPO output pins, for example, use a for-loop for the pins to reduce the code a little, pick pins at random, etc; you will soon get the hang of things.

That is the end of Tutorial 1.