Arduino

ez Serial/Parallel IC Library (SIPO8) ~ Tutorial 2

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

Tutorial 2
~ Relative 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 second tutorial we will build on our Tutorial 1 experiences and look at relative addressing of the virtual SIPO output array pin pool. This form of addressing is used exclusively to address the SIPO output pins associated with banks.

(If you have not already done so, download the SIPO8 User Guide from github.)

Objectives

We shall concern ourselves with creating a virtual SIPO environment in which we can apply further basic principles. In particular we shall learn:

1. how we can wire up and connect a single 74HC595 IC (SIPO) to a microcontroller (see Tutorial 1, if you have not already done so)
2. the structure of an Arduino sketch suitably configured to drive the physically connected SIPO IC and its LEDs, in particular:
    a. creating a virtual SIPO bank
    b. using relative addressing to access/update the created virtual SIPO bank
3. witness the outputs of the sketch – shifting LED patterns and serial monitor SIPO data

Steps

If you ran through Tutorial 1 and still have your components set up as for that tutorial, then skip to The Code/Sketch, otherwise continue as below.

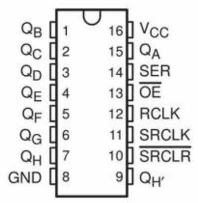
Kit List

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

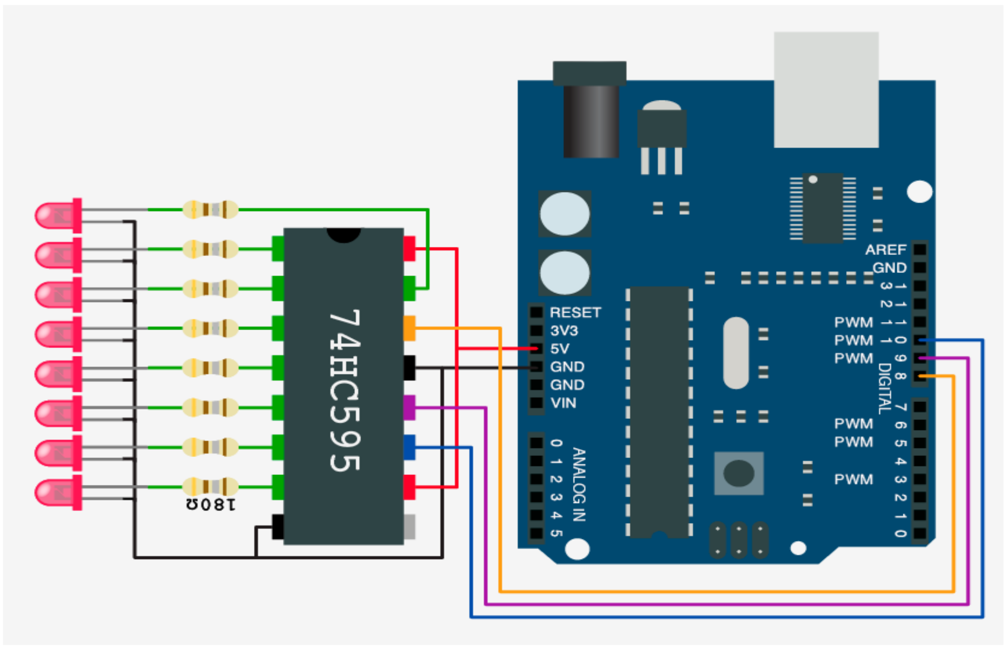
Microcontroller / SIPO Interface Pin Configuration

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's done, let's look at a sketch to demonstrate relative pin/bank addressing. This sketch will create a single SIPO bank with an output pin address range from 0 – 7 and a SIPO bank address of 0. As there will only be a single bank with one SIPO, the absolute address range coincides with the relative address bank range. This is always true but is of no consequence for what we wish to demonstrate – relative addressing.

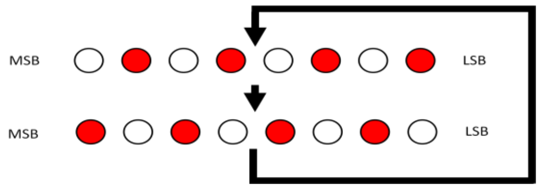
The sketch will successively setup the same bank with two different 8bit output pin patterns and xfer each out to the physical SIPO IC/LEDs, in turn, with a short delay in between. The sketch then continues this repeating cycle. Notice the two different methods used in each half of the sketch's main loop.

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

```
//
//   Tutorial 2 - use of ez_SPI8 library,
//   Relative addressing - 1 x physical SIPO IC
//   The sketch demonstrates two ways in which SIPO output pins
//   may be updated, individually (set_bank_pin) or in a group
//   of 8 pins (set_bank_SIPO), representing a single 8it SIPO
//   within a bank.
//
//   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  // two virtual SIPOs 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 bank_id;  // used to keep the SIPO bank id

void setup() {
  Serial.begin(9600);
  bank_id = my_SIPOs.create_bank(8, 10, 9, 1); // absolute pin addresses 0-7,
relative addresses 0-7
  if (bank_id == 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 the only SIPO (0) in the bank to all outputs off/LOW
  my_SIPOs.set_bank_SIPO(bank_id, 0, LOW);
  my_SIPOs.xfer_bank(bank_id, LSBFIRST);   // move virtual pin statuses to real
SIPO output pins
  do {
    // setup pattern for first cycle: 0b01010101
    // note that set_bank_pin uses relative addressing
    my_SIPOs.set_bank_pin(bank_id, 0, HIGH); // least significant bit/pin
    my_SIPOs.set_bank_pin(bank_id, 1, LOW);
    my_SIPOs.set_bank_pin(bank_id, 2, HIGH);
    my_SIPOs.set_bank_pin(bank_id, 3, LOW);
    my_SIPOs.set_bank_pin(bank_id, 4, HIGH);
    my_SIPOs.set_bank_pin(bank_id, 5, LOW);
    my_SIPOs.set_bank_pin(bank_id, 6, HIGH);
    my_SIPOs.set_bank_pin(bank_id, 7, LOW); // most significant bit/pin
    my_SIPOs.xfer_bank(bank_id, MSBFIRST);
    delay(500);
    // setup reverse pattern using 8bit write function: 0b10101010
    // note that set_bank_SIPO uses relative addressing for SIPOs in the bank
    my_SIPOs.set_bank_SIPO(bank_id, 0, 0b10101010);
    my_SIPOs.xfer_bank(bank_id, 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, as for Tutorial 1:



The result of the sketch is identical to that presented in Tutorial 1 which used absolute addressing to achieve the same ends.

Notice that the first half of the above sketch looks similar to the Tutorial 1 sketch, but uses relative pin/SIPO addressing using functions `set_bank_pin`, `set_bank_SIPO` and `xfer_bank`, in rotation, to update the virtual SIPO bank.

Which half is more efficient? Clearly the second half as this wraps up eight pin status changes up into a single function call, which itself operates on an entire status byte rather than individual bits/pins.

If we wanted to be super efficient we would replace the first part of the sketch with the same function/method as for the second part, with a different output pin bit pattern (i.e. `set_bank_SIPO(bank_id, 0, 0b10101010)`). Try it.

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
```

That is the end of Tutorial 2, which I hope you found instructional.