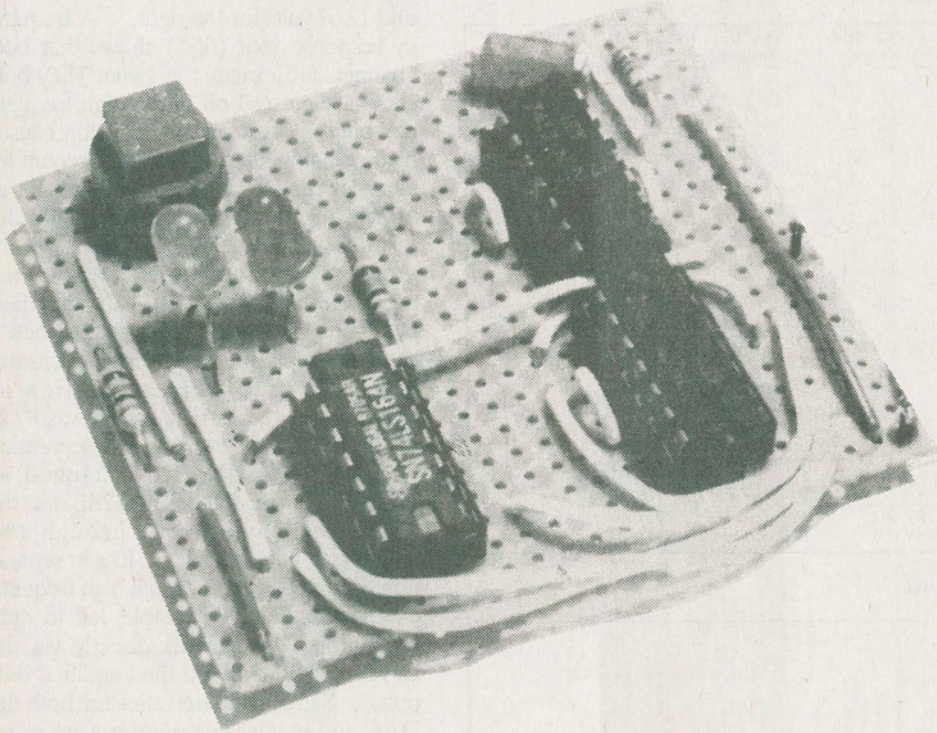
Shift Register Demonstrator

One of the basic building blocks of digital circuitry explained.

OWEN BISHOP



Shift registers, as their name implies, are used for shifting data. They have various designs, of which the one shown in Fig. 28 is a simple example. Here we have a register of four "flip-flops". Each flip-flop can be in the set or reset state, so the register as a whole can hold any of the 4-bit binary values 0000 to 1111.

The flip-flops are connected to each other in a chain and, when the clock input changes from low to high, the data is shifted one step along the chain. Let us see what happens.

A circuit for testing a shift register is shown in Fig. 2, and Fig. 3 shows how to set this up on a breadboard. The shift register used has eight flip-flops, A to H, but we use only the first four to begin with. The clock pulses that make shifting occur are provided by a 555 time IC wired as an astable.

With the resistors and capacitor indicated, the astable runs at 0.48Hz. It delivers an upward-going "clocking edge" to the register every two seconds approximately. This register (IC2) has two inputs, useful for other applications, but here we input *A* '*high*' and use input *B*. Fig. 3 shows the input wire unconnected. Before connecting the power to the circuit, push the free end of the input *B* wire into a +V socket in the top row of the breadboard. This gives a high input to *B*.

The four LEDs (D1-D4) show the state of each of the flip-flops *A* to *D*, reading from left to right. When you first switch on, these LEDs all come on, though perhaps not all at once. After a few seconds they are all on, since a high input is being fed into the chain of flip-flops.

Now put the flying lead into one of the sockets in the bottom row of the board (0V). Keep it there until diode D1 goes out, then return it to the top row. Watch the LEDs. Can you see the "low" input being shifted along the chain? Repeat this a few times, until you understand just what is happening.

Try putting the flying lead into the 0V socket for differing lengths of time. Watch the pattern of highs and lows being shifted along.

When you have finished, leave the circuit connected, as we shall be adding to it later.

Applications

The register used above had *serial input*; the data is fed into it one bit at a time. It has *parallel output*; each flip-flop has its own output terminal. Thus, this is a *serial-*

in/parallel-out register, or SIPO for short.

If you want *serial* output (perhaps to feed to another 8-bit register) this can be taken from the output of register *H*. So this is also a SISO (serial-in/serial-out) register. Other types of register are available with parallel input and serial output (PISO) or with parallel input and parallel output (PIPO).

Each type has its uses, especially in computers where data often needs to be shifted. For example, you may have a byte of data that has to be sent along a pair of wires to a printer. The data is put into an 8-bit PISO register, eight bits at once. Then it is shifted out a bit at a time and fed down the wire to the printer.

Shifting is used for calculations in the microprocessor itself. For example, the decimal value 116 is represented in binary by:

1110100

If this is shifted one place to the right, it becomes:

0111010

This has a decimal value 58. Shifting the binary digits one place to the right is equivalent to dividing by two. Conversely, shifting one place to the left is equivalent to multiplying by two.

Shift registers shift either right (like the one we used above) or to the left, and some can be controlled so as to shift either way. With these we can rapidly multiply or divide by two or its multiples. We would normally use a PIPO register for such calculations.

Random - Or As Good As

An extension of the demo shift register circuit to create "random" sequence is shown in Fig. 4. Fig. 5 shows how to modify the breadboard component layout. The output from the two last stages of the register are fed to a network of NAND gates and the output from this network is fed back to the input, pin 2, of the register.

The gates are connected to make up an EX-OR (exclusive-OR) gate. We could have used a ready-made gate in a 7486. The logic of EX- OR is "A or B but not *both*". Its truth table is:

INPUT		OUTPUT
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	0

A shift register with two of its outputs EX-ORed together and fed back to the input has interesting properties. Connect the battery and watch what happens.

If by chance none of the LEDs comes on, disconnect the battery and try again. Obviously, if all flip-flops hold "0" (low), both outputs will be "0", the output of the EX-OR gate will be á0", and a series of zeros will be fed back, indefinitely.

If at least one of the flip-flops holds "1" (high) to begin with, try to write down the stages as they occur. How many different combinations of "0"s and "1"s can you record?

Do they occur in a regular sequence? Does the sequence repeat? If so, how often? The answers are at the end. Incidentally, if you find things shifting too fast, slow the clock down by substituting a 22u or 47u capacitor for C1.

You could try using other pairs of outputs from the register and find out what sequences you obtain. When you have finished, keep this circuit wired up as we shall be coming back to it later.

PARTS LIST

SHIFT REGISTER

Resistors

All .25W, 5%

R1,2 ..... 100k
R3-7 180

Capacitors

C1 .....10u (see text)
C2 .....100u elec.

Semiconductors

D1-5 ..... LED
IC1 .....555 timer
IC2 74164 TTL 8-bit shift reg.
IC3 ..7400 quad 2-in NAND

Miscellaneous

Breadboard, B1 ^V battery and connector, wire, etc.

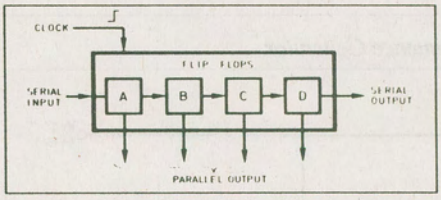


Fig. 1. A SISO/SIPO shift register.

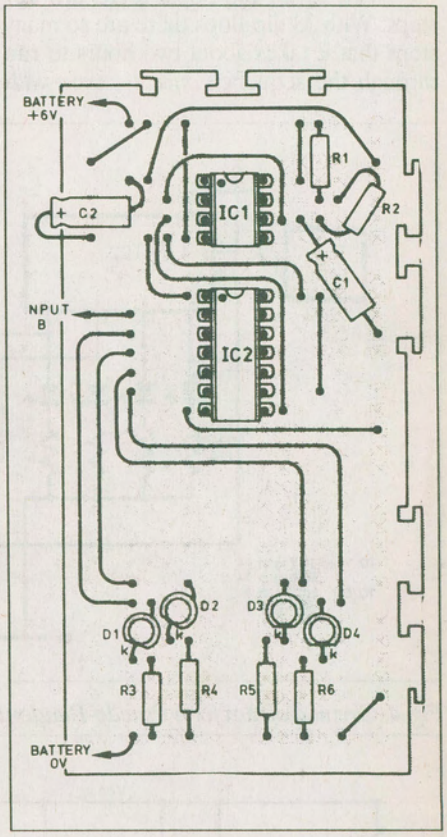


Fig. 3. Shift register demonstration board component layout.

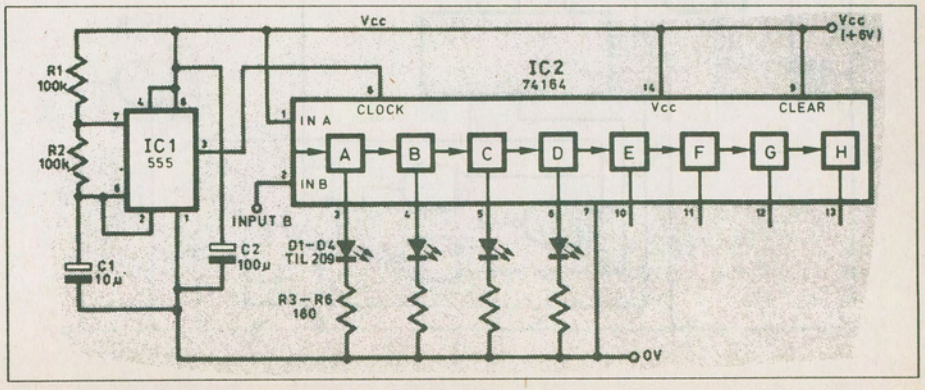


Fig. 2. Circuit diagram for investigating a shift register.

Pseudo-Random Sequences

Given any one combination of "1"s and "0"s in the register it is not difficult to work out what the next combination will be. For example, if we have "0110", the last two digits are unalike, so their EX-OR is "1" (see truth table). Shifting the existing digits gives "-011", and the result of EX-ORing is put in on the left, giving "1011".

Working in this way, you can confirm that the sequence repeats itself after 15 steps, as listed opposite. However, though it is easy to do this in the simple case of 4-bits, it becomes tedious when there are many flip-flops in the register.

With seven flip-flops, there are 127 steps. With 33 flip-flops there are so many steps that it takes about two hours to run through the sequence *once* – even with the clock running at 1MHz (one million shifts per second). With 100 flip- flops and a clock rate of 10MHz, the time taken to run through the sequence is longer than the age of the universe!

Even with a shift register of reasonable length (say, a dozen or so flip-flops) the sequence is so long that it is virtually impossible to memorize it. Although, strictly speaking, it is predictable, it is not practicable for anyone to know what the next combination of digits will be. It *appears* to be *un*predictable. In other words, the sequence is *pseudo- random*.

The situation is similar to that in which a numerical algorithm is used to generate a eries of pseudo-random numbers. Although the sequence repeats itself after many numbers have been generated, and although it is possible to calculate what the next number will be, it is just not practicable to do so and the series can be used as if it were truly random.

Pseudo-Random Noise

Replace C1 with a 100n capacitor, to speed the clock up to 480Hz. Connect the battery and watch the LEDs. They should now flicker in an *apparently* random way, like the flickering of a candle flame in a draught.

Connect a crystal earphone to the circuit, see Fig. 6. You should hear an *apparently* random series of crackles. This "random" noise is called "*white noise*". It is something that we often want to get rid of as it produces unwanted hissing and rushing sounds that spoil our hi-fidelity audio. But sometimes, we wish to generate white noise for sound effects and we normally do this by using a shift register.

Sound effects chips contain a long register (about 17 flip-flops) used for this purpose. By clocking at different rates, and taking the output from different stages we are able to produce different kinds of white noise under controlled conditions.

When you listen to the white noise from a 4-bit register, you can hear that the sequence repeats fairly often. Try using seven flip-flops, taking the outputs fro *F* and *G* (pins 11 and 12) instead of *C* and *D*.

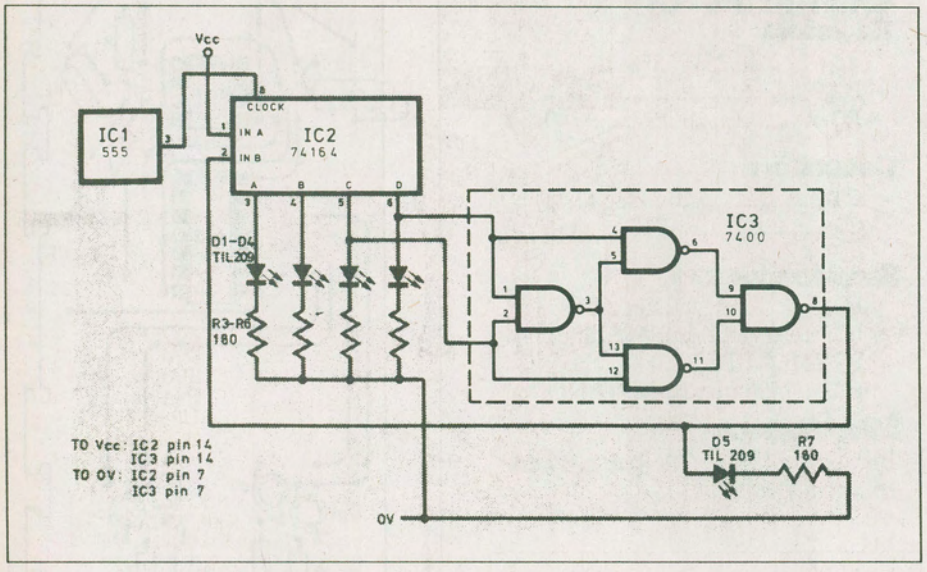


Fig. 4. Circuit diagram for a Psuedo-Random Sequence Generator.

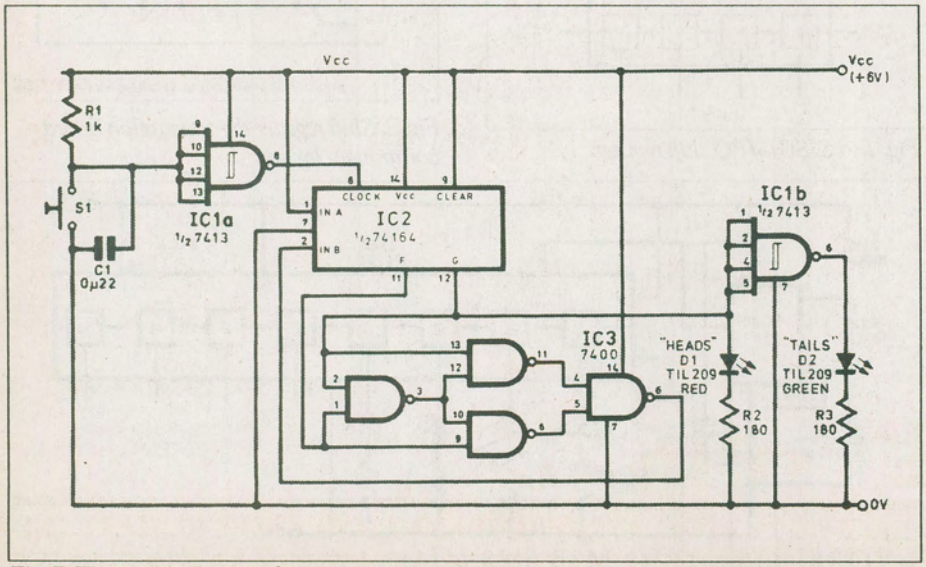


Fig. 7. The circuit diagram for creating a Pseudo-Random Heads or Tails.

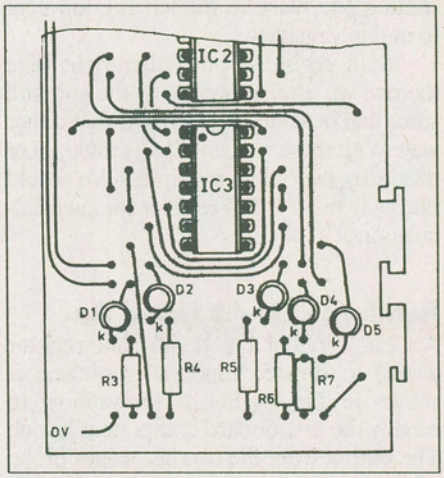


Fig. 5. Layout of components for random sequence.

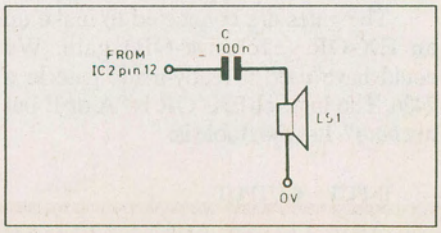


Fig. 6. Using a crystal earphone to listen to the random output.

This gives 127 stages in the sequence. Try increasing the clock rate by further reducing the value of capacitor C1.

We cannot easily use all eight flip-flops because an 8-bit register needs three outputs to be EX-ORed to get a long sequence. With only two outputs the sequence is short — as you can work out for yourself, on paper.

Heads Or Tails

We conclude with a simple project to demonstrate the shift register IC as a heads-or-tails generator. Heads or tails devices usually consist of a fast-running clock that you can stop by pressing a button.

Depending on whether the clock is "high" or "low" when it is stopped, the result is "heads" or "tails". The randomness of this result depends on *you*, not the electronics.

Moreover, unless the clock's output is high for *exactly* as long as it is low (i.e. it has a mark-space ratio of exactly 1), then the result is biased. An exact mark-space ratio is difficult to achieve and to maintain.

The Pseudo-random Heads or Tails circuit (Fig. 7), based on a 7-bit shift register, takes the output from register *G*. A high output turns on the red "heads" LED A low output turns on the green "tails" LED

It is slightly biased, since the 0000 state is not allowed, and in continuous series of 127 "throws" there will be 64 heads and 63 tails. Betting on "heads" gives you a marginal chance of profit!

Remember that the sequence is only pseudo-random, not truly random. In theory, you could memorize it and be able to "predict" the next result. But it is highly unlikely that any normal person could succeed in such a feat of memory and recognise how far along they were in the sequence. So, *in practice*, this is as random as spinning a coin.

The "throw" is made by pressing switch S1. This is debounced by the Schmitt trigger gate IC1a, to give a single clean transition from low to high when S1 is pressed. It clocks the shift register IC2 one step. The EX-OR gate IC3 is made from four NAND gates, as before.

The output from flip-flop *G* is fed directly to the red LED (D1). It also goes to the other Schmitt gate IC1b, used simply as an inverter, to turn the green LED (D2) on when *G* is low.

Construction

The stripboard component layout for the Heads or Tails circuit is shown in Fig. 8. Commence construction by making all the breaks in the copper strips as indicated in the underside view. These should be checked carefully before tackling the top-side components.

The IC holders, terminal pins and link wires should now be carefully soldered in position. It is a good idea to double-check these connections before finally soldering.

Next the resistors, capacitors, LEDs and push switch S1 should be soldered in place. This should be followed by the supply leads and the completed board given a final checkover prior to connecting the battery B1.

Answers

The sequence is as follows — 1111, 0111, 0011, 0001, 1000, 0100, 0010, 1001, 1100, 0110, 1011, 0101, 1010, 1101, 1110, and then repeats.

There are 15 stages, representing all the 4-bit binary values, 0001 to 1111 (but not 0000, for reasons explained earlier). ■

PARTS LIST

HEADS OR TAILS

Resistors

All .25W, 5%

R1 ..... 1k
R2,3 100

Capacitors

C1..... 0.22u

Semiconductors

D1 ..... red LED
D2 .....green LED
IC1 ...7413 dual 4-in Schmitt
IC2..... 74164 8-bit shift reg.
IC3 .. 7400 quad 2-in NAND

Miscellaneous

Stripboard, 3 14-pin DIP sockets, S1 press-to-make pushbutton, B1 6V battery and connector, wire, etc.

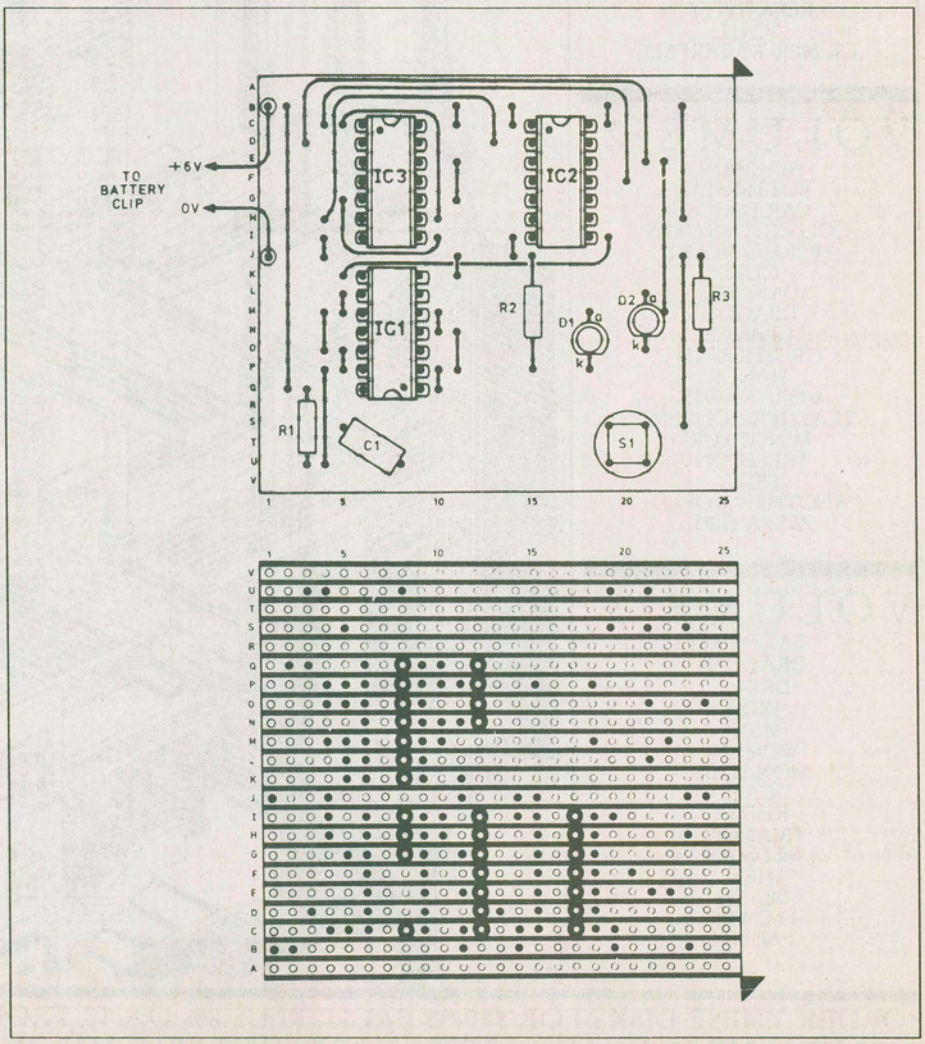


Fig. 8. Stripboard component layout for the Heads or Tails circuit.