ELECTRONICS BY EXPERIMENT

Part 4 by Graham Dixey C. Eng., M.I.E.R.E.

Introduction

A register is also a sequential logic circuit but, whereas a counter is concerned with the process of 'counting' the input data, the register is principally concerned with storing it. However, this bold statement can be a little misleading, since it implies an event that is going on for a substantial period of time, which may not be the case at all. As an example of this consider the familiar electronic calculator. When two numbers are added together, each is deposited into a register and on the command 'add' the two numbers are replaced by their sum. This sum is held in a register that only a moment before held one of the numbers. Thus, the storage of the original number was of very short duration. Another example is when data is being sent from a microcomputer to a printer. It passes out via a register, each byte in turn being held by this register until it is replaced by the next one in the queue. Thus, a register is really a string of flip-flops that can hold a data word for a period of time, no matter how long or short that is. There are also certain other ways in which particular types of register can be useful and they will also be discussed.

To start with the most elementary idea of a register, consider Figure 1, in which the four possible modes of operation are shown. These modes arise because there are two ways in which data can be transmitted from point to point. These ways are known as 'parallel' and 'serial' transmission respectively.

In parallel transmission, there is one line for each bit of the data word and all bits are transmitted at once. Thus an eight-bit word (a byte) requires eight separate lines – an example of parallel data transmission is the well-known Centronics printer interface. Because all bits are sent at once, parallel transmission is fast but requires lots of conductors.

In serial transmission, there is one line only, which all bits have to share. They are therefore sent down it one at a time. Thus, no matter how many bits there are in the data word only one transmission line is used but the process gets slower and slower as the bits queue up for their turn on the line! The advantage is the simplicity of it. An example of serial transmission is the RS232 interface for printers and modems.

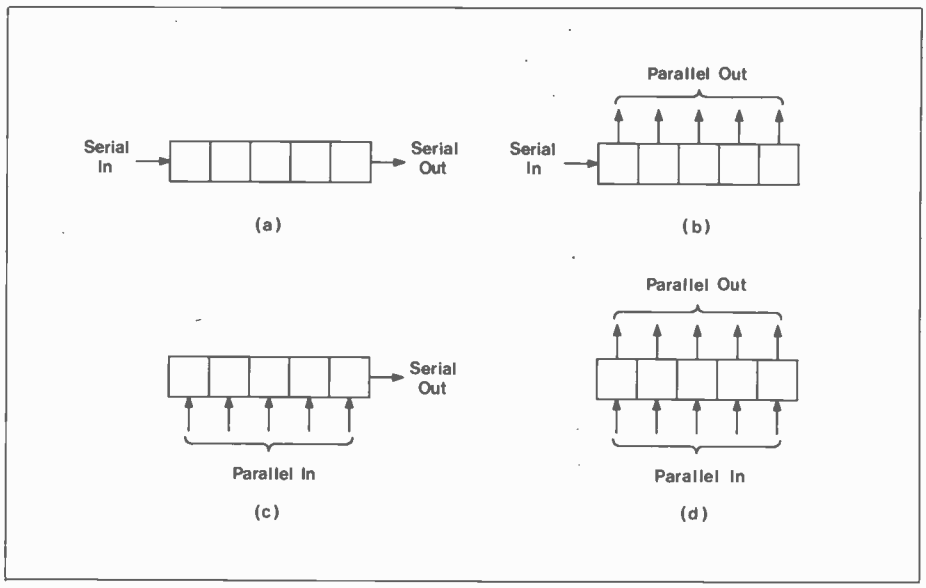


Figure 1. The four main modes of register operation. (a) Serial In, Serial Out (SISO); (b) Serial In, Parallel Out (SIPO); (c) Parallel In, Serial Out (PISO); (d) Parallel In, Parallel Out (PIPO).

How does this relate to registers? Quite simply. Each flip-flop of a register holds just one bit of the data word. Serial or parallel operation refers to the ways in which the data is loaded into the register or sent out from it.

For example, one mode is Serial In Serial Out (SISO). There is only one input line and only one output line. Data bits are 'clocked in' one at a time and then 'clocked out' in a similar fashion. For example, if the register is 'eight bits wide' (meaning it has eight flip-flops), it will take eight clock pulses to store a data word and a further eight clock pulses to 'shift' it out again. Note the use of the word 'shift'. Registers of this type are known as shift registers. In continuous operation a stream of bits flows in at one end and out at the other.

However, it is possible to shift the data in serially until all flip-flops hold one bit of the data word and then access all bits of the data word at once, so that the output is 'parallel'. Obviously this mode of operation is known as Serial In Parallel Out (SIPO).

Reversing the latter mode is also possible. With one clock pulse the whole register can be loaded with the data word, which can then be shifted out serially. This is known as Parallel In Serial Out (PISO) operation.

A little thought will reveal that the SIPO type is a 'serial to parallel' converter, while the PISO type is a 'parallel to serial' converter. The significance of this can be seen where a computer is connected either to another computer or to a peripheral through a serial connection, i.e. a single line. Since the internal organisation of a computer is on parallel lines (referring to the data bus), there must be a parallel to serial conversion at the sending end and a serial to parallel conversion at the receiving end. It is an ideal function for registers to perform.

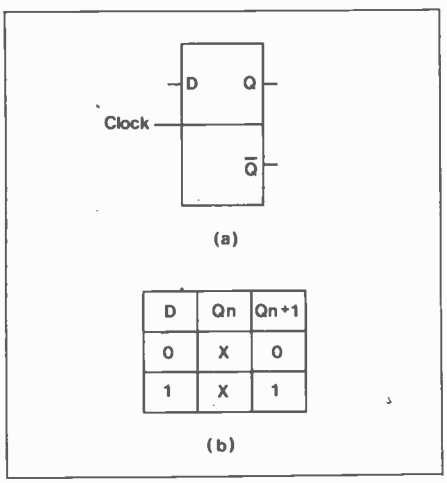
Finally, it is possible to load a register by parallel input and access it also by parallel mode. This is Parallel In Parallel Out (PIPO).

The D-type Flip-flop

As a general (but not invariable) rule, counters use JK flip-flops and registers use a type known as the D flip-flop (D stands for Data). The D flip-flop is rather simpler in operation then its JK equivalent. Both its symbol and truth table are shown in Figure 2. It has a single input line, marked D, a CLOCK input and a pair of complementary outputs, of which the Q output is the more useful. The truth table reveals its true simplicity.

The logic level, logic 0 or logic 1, to be stored is first applied to the D input. When a clock pulse is applied this level at D shifts to the Q output, where it will remain until a new data bit is applied to D and a further clock pulse arrives. Thus, the Q output merely copies whatever is at the D input after the arrival of a clock pulse.

As an experiment, try connecting a low-frequency clock oscillator to the clock input of a D flip-flop, say 1Hz (a suitable design appeared in Part Three). At the same time connect a switched logic level to the D input. Connect a logic level indicator to the Q output. Randomly vary the logic level input and note how, on the arrival of the next clock pulse, the logic level applied to D transfers to Q.



D	Qn	Qn+1
0	X	0
1	X	1

Figure 2. The D type flip-flop. (a) Circuit symbol; (b) Truth Table showing that, after clocking, the output at Q equals the input at D. Note: Qn = state of Q before clock pulse; Qn+1 = state of Q after clock pulse; X = 'Don't care', i.e. can be 0 or 1.

The 7474 D-type Flip-flop

This package contains two D flip-flops and its pin-out diagram appears in Figure 3. This is an example of a 'positive edge triggered flip-flop'. This was mentioned in Part Three but will be repeated here. The clock pulse can be considered as two transitions occurring one after the other. The first is from Logic 0 to Logic 1 (the positive edge) and the second is from Logic 1 back to Logic 0 (the negative edge). In a positive edge triggered flip-flop only the first transition matters. The data transfer from D to Q occurs at this instant only. The duration of the clock pulse is of no importance, nor is the subsequent trailing edge.

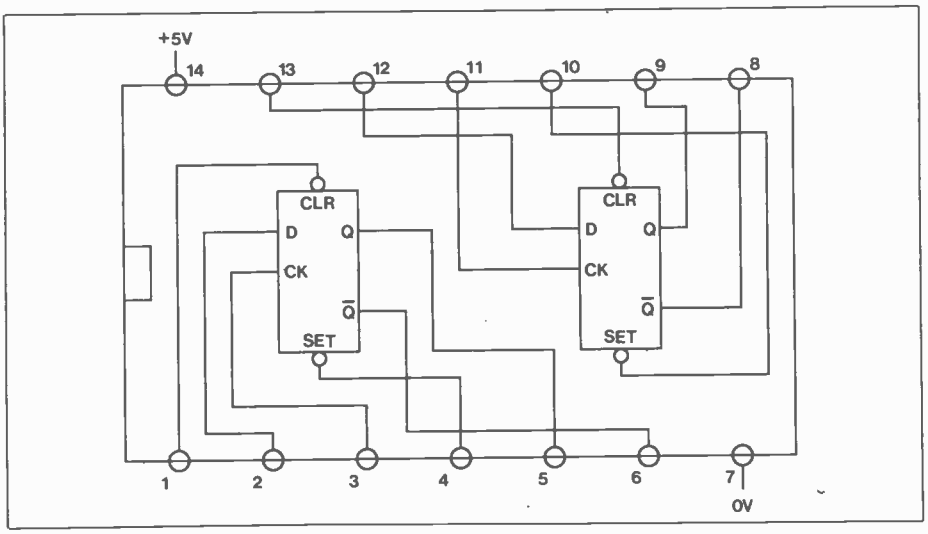


Figure 3. Pinout diagram of the 7474 Dual D type, positive edge triggered, flip-flop.

Each flip-flop has two connections not so far mentioned. These are marked on the 'chips' as SET and CLR and are known usually as 'preset' and 'preclear'. The 'bubble' on each of these indicates that they are 'active low' connections, that is taking the terminal to Logic 0 causes the required action to take place. It is usual to connect all the CLR pins together for all the flip-flops in a register and call this the RESET line. Taking this low, momentarily, clears all the flip-flops, that is all Q outputs go immediately to Logic 0. The SET pins can be used when required to preset a given value into a register, that is to give it a particular state to start off with rather than starting with a clear register. Examples of this use appear later.

Shift Registers

The simplest types of register are the SISO and SIPO types. Their circuit diagrams appear in Figure 4. Basically they are the same; it is only in the way they are used that they differ. The data input to the register is the D input of the first flip-flop; the data input of any subsequent flip-flop is always the Q output of the preceding one. All flip-flops are clocked at the same time.

Purely for simplicity, only four-bit registers are shown. The same principles apply however many stages there are. To study the mode of operation, connect up this register using two 7474 ICs (Figure 5). Connect all CLR pins together and wire this common line through a switch to the 0V line; leave the switch open for the moment. Connect a de-bounced switch (see Part Three for details) to the clock input and logic level indicators to each of the Q outputs. Finally connect a logic level switch to the data input. Switch on; the Q outputs will be quite arbitrary. Momentarily connect the RESET line to 0V using the switch provided for this purpose. The Q outputs should now all be zero. Set the logic level at the input to Logic 1. No changes should be observed in any of the flip-flops. Leave this level as set and pulse the de-bounced switch four times. The Q outputs should take up the successive

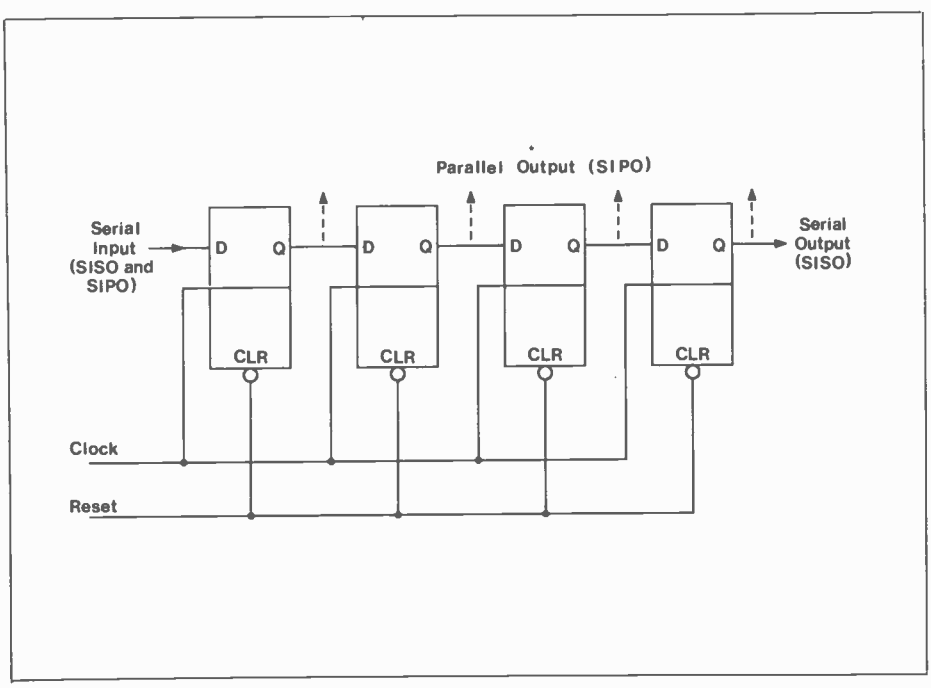


Figure 4. A 4 bit Serial In, Serial Out shift register (SISO), with the simple modification needed to get a parallel output (SIPO).

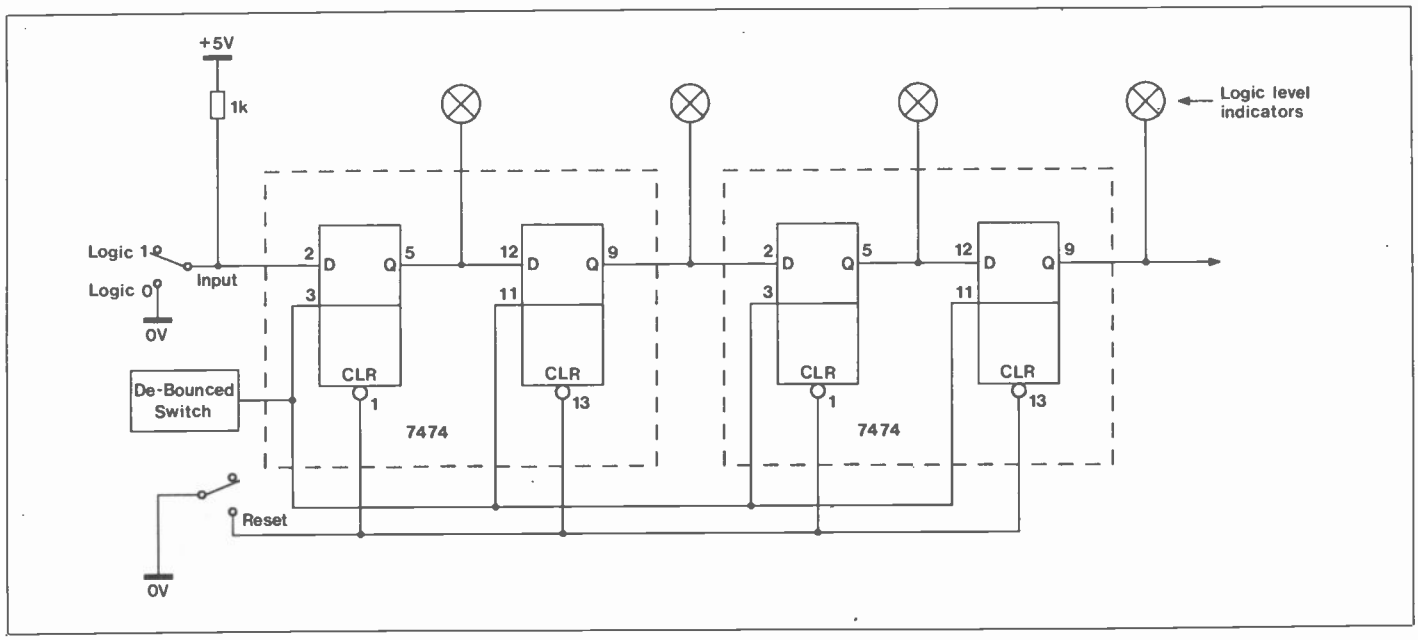


Figure 5. The circuit of Figure 4 implemented with two 7474 ICs and the necessary hardware for evaluating it.

states: 1000, 1100, 1110 and 1111, as shown by the four LEDs of the logic level indicators. If it doesn't do this, check in turn – the circuit wiring, the ICs themselves, the output from the de-bounced switch; finally, "have you left the RESET switch low?"

If all is well, consider the fact that the data word 1111 is shown by the four LED indicators. In other words, not only has it been stored but it is available at the Q outputs if four lines were to be connected to these outputs. Thus, although the data word was entered serially using four clock pulses, it is now available at the Q outputs without any further action being necessary. This is SIPO operation.

But if it is desired to remove the data word serially, four more clock pulses will be needed. This will normally load a new word so set the logic level input switch to Logic 0 and pulse the de-bounced switch four more times. The logic levels at the Q outputs should now take up the successive states: 0111, 0011, 0001 and 0000. Note that the original data word has been progressively shifted to the right until it was replaced by the new word 0000. This is an example of SISO operation. What has not been done in this case is to 'catch' the data word as it left the register. Special provision would have to be done to do this; nonetheless, the basic operation of serial shifting has been shown.

Parallel In Parallel Out and Parallel In Serial Out Registers

These registers, shown in Figure 6, illustrate that sometimes extra gating is needed to get the register to perform in particular ways. In these cases, each flip-flop requires a pair of two-input NAND gates and an inverter. These are concerned with the parallel loading. Once the parallel load has taken place, the contents of the register can be accessed in parallel mode merely by connecting wires to all of the Q outputs, or the data can be obtained serially by clocking the register four times, as already described for the

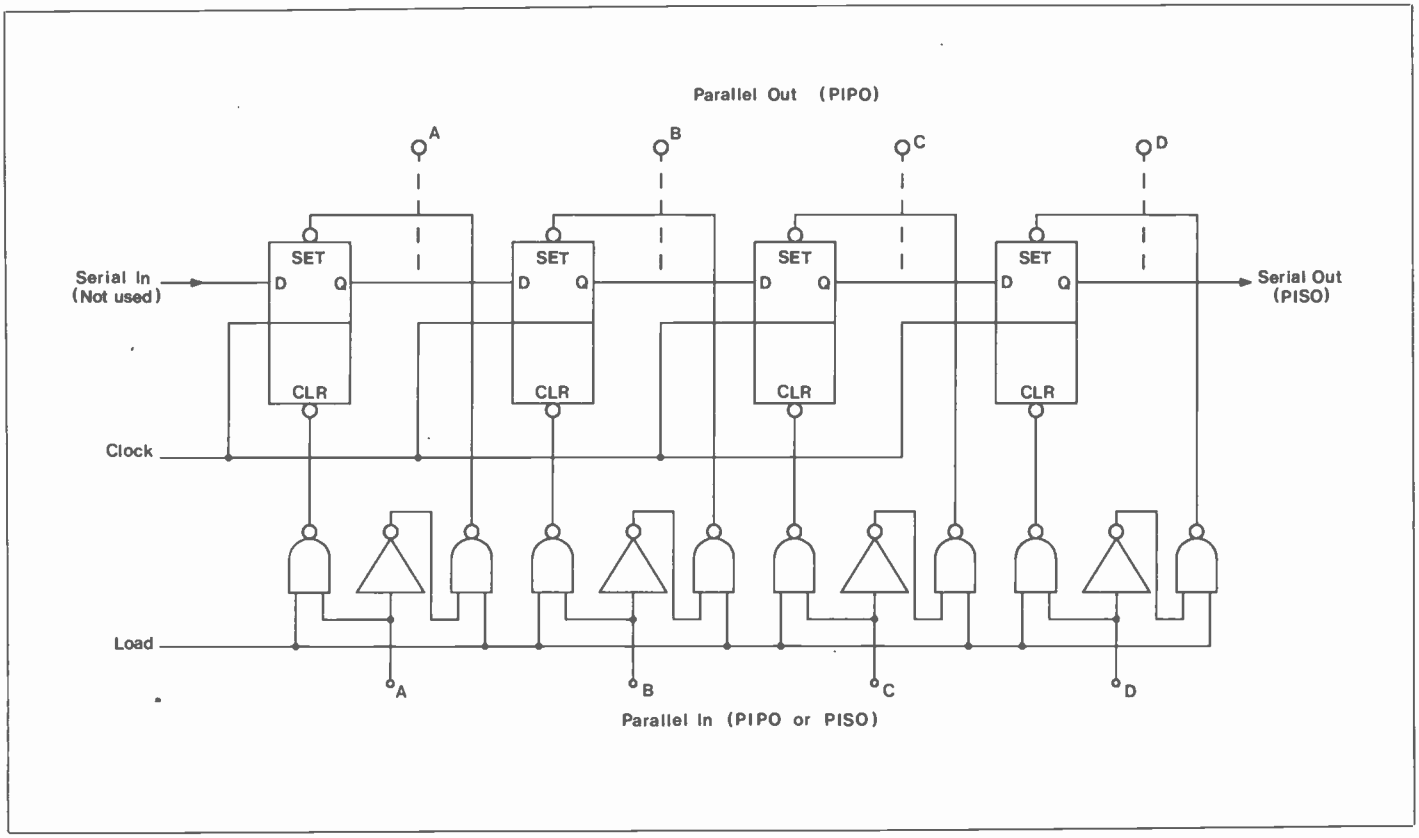


Figure 6. Circuit diagram(s) for the Parallel In, Serial Out (PISO) and the Parallel In, Parallel Out (PIPO) registers.

SISO and SIPO types.

The parallel loading works as follows. The line marked LOAD is normally held at Logic 0. Thus, both NAND gates of each pair associated with each register stage have logic 1 output levels. These drive the SET and CLR pins of the flip-flops; both pins being at logic 1s, nothing happens. Each DATA IN line will have either a Logic 0 or a Logic 1 on it, according to the data to be loaded. This is applied directly to one NAND gate and in inverted form to the other NAND gate. Consequently, one NAND gate of a pair will have Logic 1 and Logic 0 at its inputs, while the other will have both inputs at Logic 0. Now let the LOAD line go to Logic 1. One of each pair of NAND gates will now have two Logic 1 inputs, the condition that causes the output of a NAND gate to go to Logic 0. Since the SET and CLR inputs are negative-acting, each flip-flop will become SET or CLEAR according to the logic level at its DATA IN input. The best way to follow this operation is to connect up the circuit and use a logic probe to look at the logic levels on all pins of the NAND gates and inverters during the load operation.

To connect up the circuit will require an LED logic level indicator at each Q output, a set of four logic level switches for the four DATA IN lines and a switch for the logic level on the LOAD line. Once the data word has been loaded and the operation fully understood, the shifting operation can be shown by pulsing the clock line with a de-bounced switch.

The Ring Counter

This is an interesting example of a shift register apparently masquerading as a counter, if the title is to be taken literally! It certainly can be used as a particular type of counter but it has a more interesting and useful application. It is frequently used for generating a 'walking one' pattern for scanning keyboards, multiplexing seven-segment displays and generating dot-matrix characters on video displays. It works very simply as follows.

The diagram of Figure 7 shows that the Q output of the last flip-flop is connected back to the D input of the first flip-flop, forming the 'ring' of the title. Thus, there is no actual input apart from the clock. To set it up it is necessary, when it is first switched on, to clear all stages (using a RESET line) and then SET just one of the Q outputs, say the first, to Logic 1. The 'preset' pin is used for this. If it is now clocked the 'Logic 1' shifts to the next stage to the right, its place being filled by a Logic 0. Every clock pulse performs the same action so that continuous clocking causes the Logic 1 to keep circulating. Naturally, when it reaches the Q output of the last flip-flop it will pass around the ring to the D input of the first, to keep the process going.

The sequence, for a four-stage ring counter, is 1000, 0100, 0010, 0001, 1000, etc.

Try this circuit out by wiring it up with a pair of 7474 ICs. Connect LED logic level indicators to all Q outputs and a low-frequency clock to the clock input. At about 1Hz it is possible to sit back and watch the 'one' circulate with ease.

The Twisted Ring Counter

This simple derivative of the ring counter just discussed is formed merely by taking the feedback from the not-Q output of the last flip-flop rather than the Q output (Figure 8). It is initialised in the same way – cleared and then a single 'one' put in. What happens when it is successively clocked? Would you be surprised to find out that the sequence is as follows?

1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000, 1000, etc.

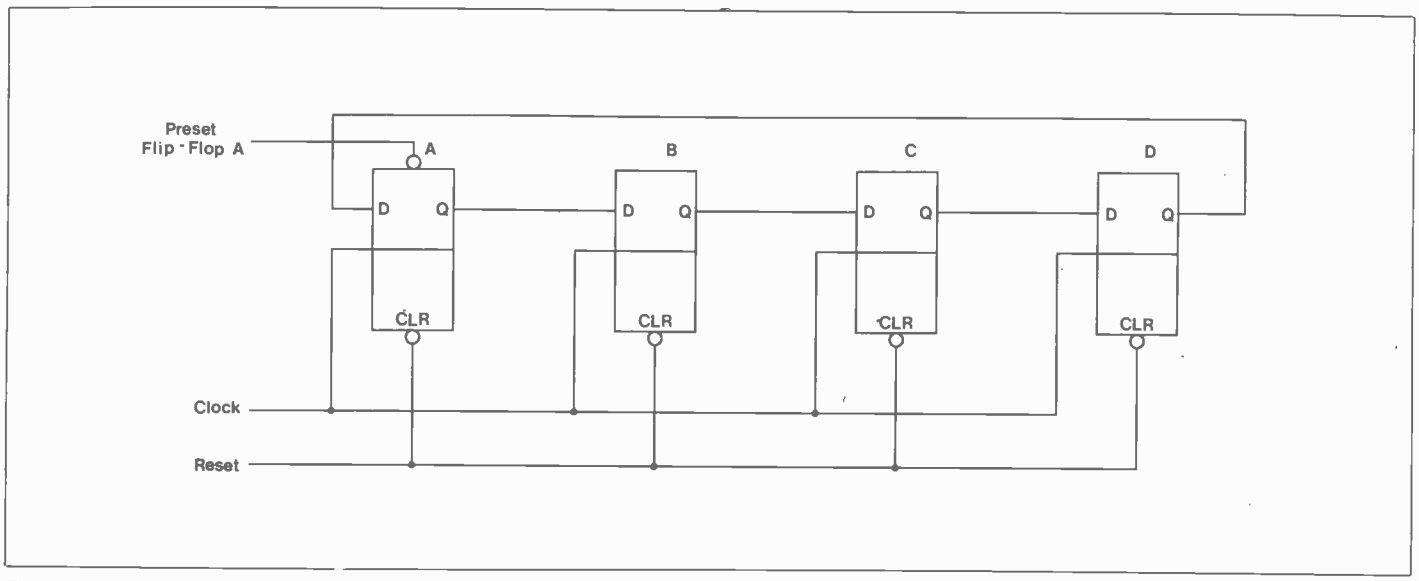


Figure 7. An example of a register with 'feedback', the Ring Counter.

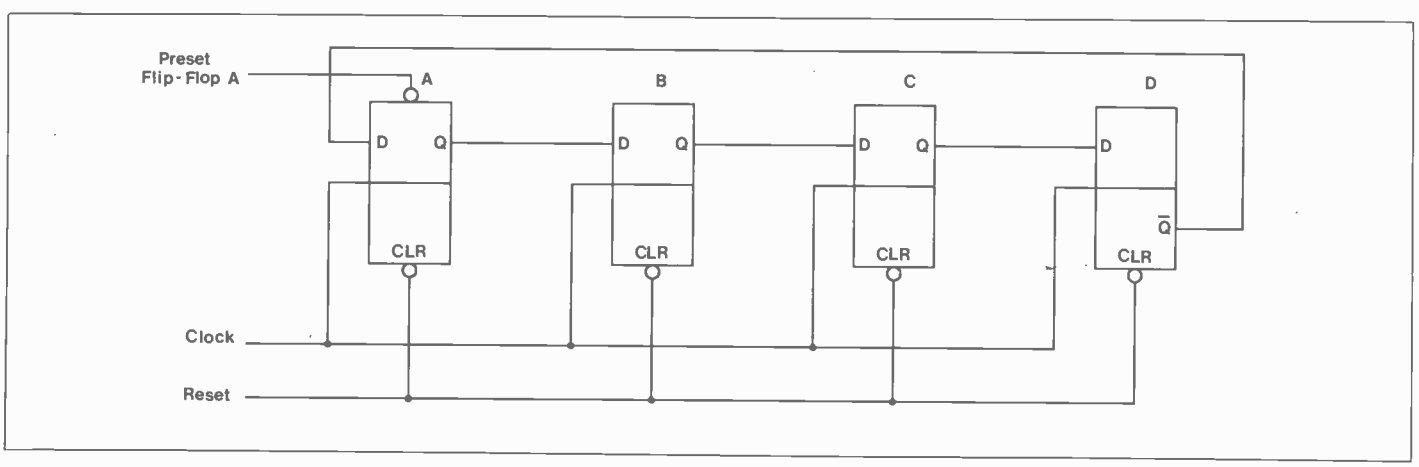


Figure 8. If the feedback is taken from the not Q output of the last flip-flop instead, the circuit of Figure 7 becomes a 'twisted ring' counter, with quite a different sequence.

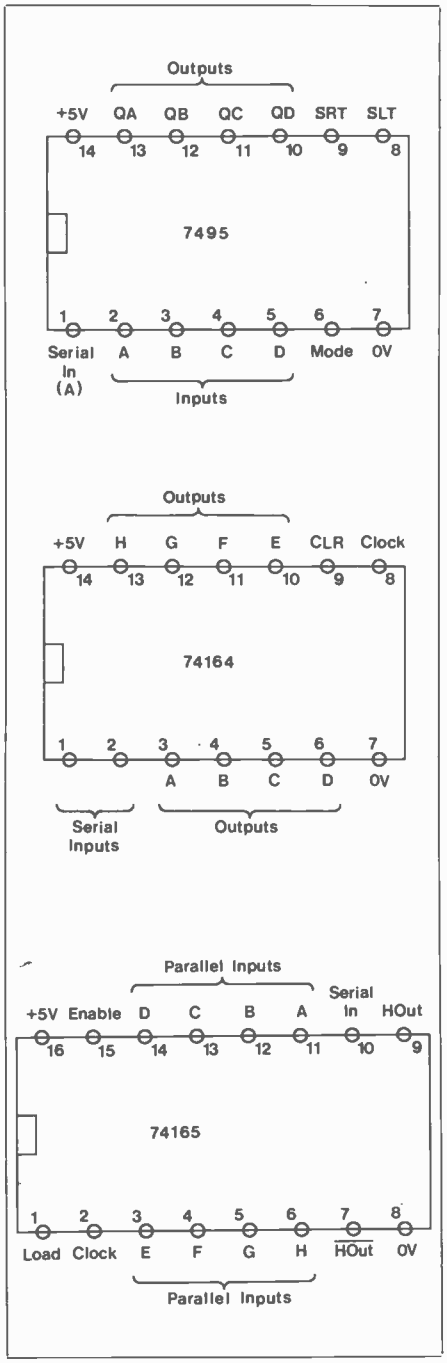


Figure 9. Pinout diagrams for some MSI shift registers.

It is interesting to notice that 'four' flip-flops give rise to 'eight' quite distinct states. In fact the number of states is always twice the number of flip-flops. Therefore, if five flip-flops are used, the number of states are ten – in other words we have a 'decade counter'. The fact that it is not the normal binary sequence for 0-9 is not really important as long as the state of the counter at any instant can be recognised by suitable decoding logic. Thus, if 0000 represents '0', then 1000 represents '1', 1100 represents '2' and so on. This code is called 'Johnson Code' and the twisted ring counter is often known as a 'Johnson Counter'. It has the characteristic of being a very fast synchronous counter, using shift register principles.

Further Work with Registers

The registers discussed so far have all been assembled from flip-flops and gates in individual IC packages. However there are, of course, fully operational registers contained within single packages. These often offer particular facilities that would be rather tedious to provide by interconnecting flip-flops and gates. A few of these will be discussed now, to round off this feature.

The 7495 Four-Bit Left-Right PIPO Register

This comes in a 14-pin DIL package and the pin-out diagram, together with those for the 74164 and 74165 registers, is shown in Figure 9. It is used as follows.

In this register all D inputs and Q outputs are individually accessible. It is possible to parallel load data into it and then either access it immediately (PIPO mode) or shift it left or right. Reversible registers normally require a fair amount of extra logic so having it 'all on one chip' is a decided advantage. It is possible to cascade these registers to make, for example, an eight-bit register.

There are two modes, known as SHIFT and LOAD. To go into SHIFT mode, the 'mode' input must be taken low. The negative edge of a clock pulse on SRT shifts data one stage 'to the right'. Similarly a negative edge on the other clock pulse, SLT, will shift the data one stage left. Thus, two clocks are needed, though in practice it could be the same clock switched between the SRT and SLT pins. Shifting works exactly as described for previous registers.

To go into LOAD mode, the 'mode' input is taken high and the data presented at the inputs LA – LD is entered into the register when a negative-going transition is made at the shift-left input (SLT). From this it can be seen that the SLT input is dual-purpose, shifting data left in SHIFT mode and loading data in LOAD mode.

The 74164 Eight-Bit SIPO Shift Register

This register, which has a right-shifting ability only, can handle serial-in data in the usual way, but can be pre-loaded with a Logic 0 or Logic 1 input as required. The data may be output in either parallel form or serial form.

For normal operation, one of the Serial Inputs is held high and the other becomes the data input. The Clear pin is also held high (taking it low momentarily clears the register). Data is shifted right one stage on every positive clock edge. To enter a Logic 1 into the register both Serial Inputs must be taken high. Taking either Serial Input low enters a Logic 0. It is essential that the clock is bounceless and noise-free.

The 74165 Eight-Bit PISO Register

This register has the versatility of being able to function as a straightforward Serial In Serial Out shift register or can be loaded with parallel data, which can then be shifted out serially.

For normal operation, the Enable pin is held low and the Load pin is held high. Data in is applied to the Serial In terminal and shifts right one stage on every positive clock edge.

To parallel load the register, the Load pin is momentarily taken low while the data input is applied to the parallel input pins A – H.

Shifting may be disabled at any time by taking the Enable pin high.

With the previous experience of connecting up and testing the other registers mentioned earlier, it shouldn't be too difficult now to devise schemes for testing the above three registers. One learns best by trying out a variety of circuits in this way.

An Oscilloscope Probe

There is sometimes a problem when using an oscilloscope to view the logic waveforms of TTL circuits. The input capacitance of the CRO may 'kill' the waveform at the test point and what is needed is some means of isolating the test circuit from these unwanted effects. This can be achieved quite easily using just two components – a resistor and a capacitor. The resistor is chosen to give attenuation together with the input impedance of the CRO, 10:1 being usual. See Figure 10. The capacitor is wired in parallel with the resistor R1 and is chosen so that the time constant of R1 and C1 equals the time constant of R_{in} and C_{in} for the CRO. This gives the widest possible bandwidth.

To take an example, the typical input impedance of a CRO consists of a parallel combination of a 1M resistance and a capacitance of the order of 10 – 50pF, say 30pF. Therefore, for 10:1 attenuation, R1 should be 9M. This gives the relation that:

$$9M \times C1 = 1M \times 30pF$$

$$\text{From which } C1 = \frac{(1 \times 30)}{9} = 3.3pF$$

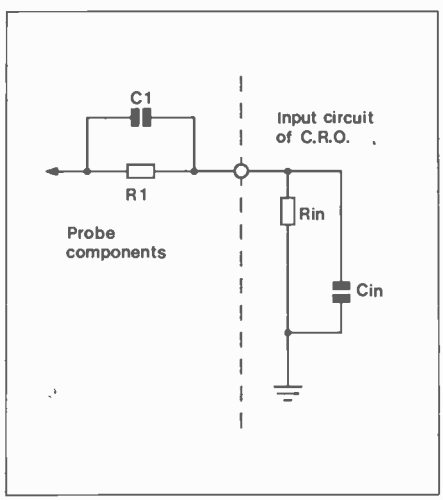


Figure 10. Probe design for a CRO.

It should be quite possible to assemble these two components into a discarded ball-pen case to make a useful probe, using a short length of co-axial cable between the probe and the Y input of the CRO.