ELECTRONICS BY EXPERIMENT

Part 3 by Graham Dixey C. Eng., M.I.E.R.E.

Introduction

Logic circuits fall into two classes, known as 'combinational' logic and 'sequential' logic. The principles and some applications of combinational logic were the subjects of Parts One and Two. Now it is the turn of sequential logic. This, as the name implies, is to do with events taking place in some particular order. The broad divisions of circuits that come under the heading of sequential logic are 'counters' and 'registers'. There is more 'action' in a sequential circuit than in a combinational one. For this reason they tend to generate more interest. Also there is a great variety of possibilities for both counter and register circuits, so many in fact that, in this issue a selection of counter circuits only will be described, registers being covered in the next issue.

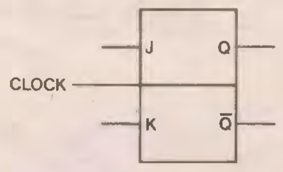
The basis of the sequential circuit is the 'flip-flop', also known as a 'bistable' (because it can take up one of two stable states) and sometimes, in particular applications, as a 'latch'. It is best, for practical purposes, to regard the flip-flop as just a 'black box' that performs some prescribed function. Textbooks for students of electronics invariably show the flip-flop function as being performed by combinations of gates, which is quite true. However, once these gates are packaged up to perform as a particular flip-flop type, there is little to be gained by considering it other than as a functional package. With this in mind look at Figure 1.

The JK Flip-flop

This figure shows the most commonly used flip-flop type, known as the 'JK flip-flop'. The symbol is shown in (a) and the truth-table in (b). The symbol is generalised to show three inputs, known as the 'steering inputs', J and K and a 'clock' input; also two complementary outputs, Q and $\overline{Q}$ (not-Q or Q-bar).

The truth-table describes the performance of the flip-flop when all possible combinations of logic levels are applied to J and K. Note the columns called Q_{NOW} and Q_{NEXT}. These are the logic levels at Q immediately before (NOW) and after (NEXT) clocking. So what is clocking?

In order to cause a counter circuit to go through its designated sequence it must be clocked. This is done by applying pulses, known generally as clock pulses to the clock input. A clock pulse is shown in Figure 2, which defines the 'edges' and



(a)

J	K	Q NOW	Q NEXT	COMMENTS
0	0	0	0	No Change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

(b)

Figure 1. The JK Flip-flop, symbol (a) and truth table (b).

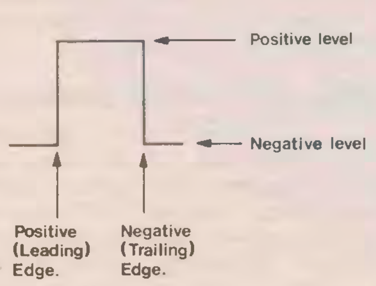


Figure 2. Clock pulse definitions.

'levels' of the pulse. The JK flip-flops that we shall be using are known as 'master-slave' types. All that this means is that the flip-flop will change state at the negative or trailing edge of the pulse.

Imagine the pulse arriving at the clock input. From logic 0 it rises abruptly to logic 1 (this is the positive or leading edge); the flip-flop does not respond at this point. Instead it waits until the logic level falls back to logic 0 when the negative or trailing edge arrives. If the J and K inputs are set up so as to demand a change of state then, at this instant of the clock pulse, the flip-flop will change state. That is, Q will either go from logic 0 to logic 1 or from logic 1 to logic 0. It is this sort of information that the truth table tells us. So back to Figure 1.

The first possible combination of J and K is that they are both logic 0. It is obviously possible for Q_{NOW} to be initially either logic 0 or logic 1. Thus the truth table has two lines for J = K = 0. Notice that for both lines Q does not change when the flip-flop is clocked. Not very exciting but, nonetheless, important. This condition is known as 'no change'.

The second possible combination is J = 0 and K = 1. Note what happens for these two lines. If Q = 0 initially, it remains at 0 after clocking; if Q = 1 initially, it becomes 0 after clocking. The flip-flop is said to be RESET when Q = 0. Thus these two lines describe the 'reset' mode.

Now reverse the logic levels of J and K so that J = 1 and K = 0. If Q = 0 initially, it becomes 1 and if Q = 1 initially, it stays at 1. The flip-flop is said to be SET when Q = 1, so that these two lines describe the 'set' mode.

Finally, there is the case when J and K both equal 1. Notice that whatever the value of Q initially, after clocking it changes to the other logic level. It is said to 'toggle' backwards and forwards between the set and reset states every time it is clocked. This is, therefore, known as the 'toggle' mode.

In order to be able to understand how any counter or register works, it is essential that the effects of the logic levels at J and K on what the flip-flop does when clocked are thoroughly grasped. This understanding, together with the realisation that the flip-flop (if it is going to change when clocked) will only do so at the negative edge of the pulse, holds the key to a proper appreciation of counter and register operation.

Level-triggered and Edge-triggered Flip-flops

There is sometimes some confusion about the meanings of 'level triggering' and 'edge triggering'. After all, both terms seem capable of describing what is happening. The triggering of the flip-flop from one state to the other may be said to occur when the 'level changes' from logic 1 to logic 0 or, in other words, at the 'negative edge' of the pulse. Let us put the matter straight. Most flip-flops that will be met are level-triggered and change state when there is a change of level, usually from logic 1 to logic 0. The clock pulse responsible for the level changes at the clock input of the flip-flop is usually fairly long. This can cause certain problems, as seen by considering what may be regarded as the normal sequence of events.

First the required inputs (to perform one of the actions described previously) at J and K are set up; next a clock pulse arrives - rising from logic 0 to logic 1 (having no effect) - then 'staying at logic 1 for a certain time period' - then falling to logic 0 (causing the action to occur). The problem is caused by the part in quotes in the centre. In effect the situation that exists during this time is that the J and K logic levels to cause the desired action to occur have been established and we are merely waiting for the end of the pulse for this action to occur. Because of the delay, it is sometimes possible for either J or K (or both) to change to some other incorrect value before the negative end of the pulse arrives, with the result that when it does so the action occurring is the wrong one.

Edge triggering is the remedy in these circumstances. The clock pulse is short and fast, giving little time for the J and K levels to change since they were established. The action is achieved by capacitive coupling that differentiates the clock pulse to give a short spike. This type of flip-flop should always be used when there is any danger of J and K changing erroneously.

The Divide-by-Two Action of the JK Flip-flop

The JK flip-flop, when in the toggle mode, acts as a basic binary or 'divide-by-two' element. This means that any train of pulses applied to its clock input at a certain frequency f will result in a train of pulses at Q of half this frequency, namely f/2. A look at the oscillograms shown in photograph 1 will make it clear why this is. Because only the negative edges of input

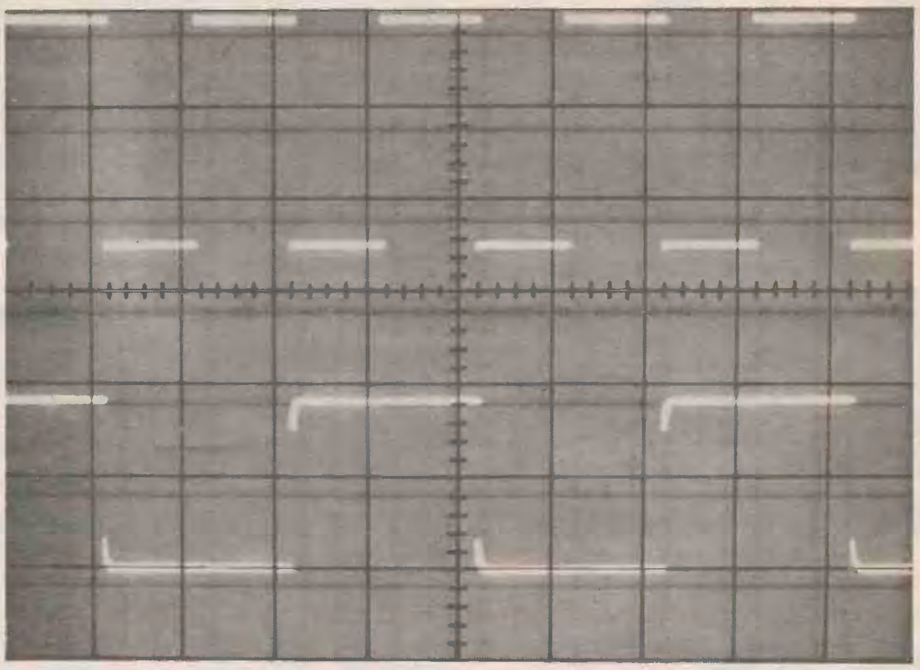


Photo 1. Oscillograms for clock input (upper waveform) and Q output (lower waveform) of a JK flip-flop wired in the 'toggle' mode. Note that the Q output changes only when the input goes 'down', hence the divide-by-two action.

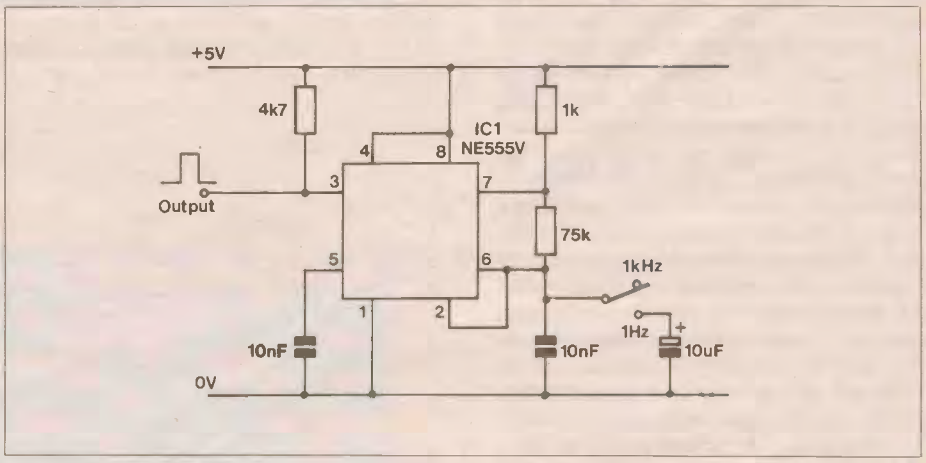


Figure 3. TTL 1Hz/1kHz oscillator.

clock pulses cause changes, all the positive edges of these input pulses are ignored. Thus, for every negative edge transition at the clock input there will be an output transition that is alternately positive and negative, hence dividing the input frequency by two. This is the basis of the counter so it is a very good idea to try this in practice. Just tie J and K together to logic 1, put in a train of pulses from the TTL oscillator (that you have now constructed!) at the clock input and look at the Q output. If you have a CRO, set the pulse frequency to 1kHz, since this will be easier to see on a 'scope. Otherwise, set the pulse frequency to 1Hz and, with LEDs on both clock and Q terminals, judge for yourself the binary dividing action.

A De-bounced Switch

Figure 4 shows another useful little circuit that you can make up. It uses a pair of cross-coupled NAND gates to form a simple latch. The feedback holds the circuit in whatever state the push-button triggers it into. Pushing the button down causes Q to go to logic 1; releasing it allows it to come back to logic 0. Thus, it generates a single pulse, very useful for

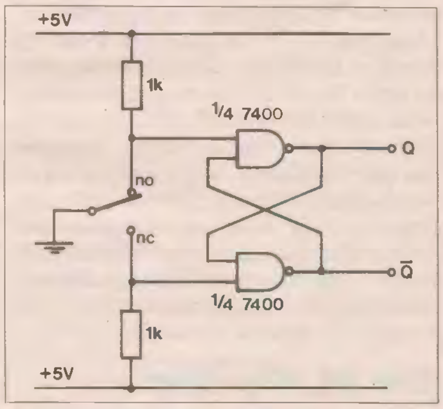


Figure 4. De-bounced switch with complementary outputs.

stepping counters (and registers) through their sequences for examination. It might be thought unnecessarily complex for such a simple task but it is not called a 'de-bounced' switch for nothing. The contacts of the average mechanical switch actually oscillate briefly between the open and closed states when the switch is operated. For many applications this effect goes unnoticed. But TTL logic devices are fast enough to follow such changes - standard TTL can respond in 10ns. Thus the circuit gets several pulses when it only needs one. The above circuit responds only to the first transition and 'masks out' those following, generating a genuine single pulse. The only awkward component is the c/o push-button switch, these mostly being push-to-make or push-to-break. If one cannot be located, a microswitch can be adapted since these usually have SPDT contacts.

Binary Asynchronous Counters

An asynchronous counter is also often known as a 'ripple through' type because the flip-flops change state one after the other, the effect appearing to 'ripple through' from first to last. Only one stage, the first, is clocked directly from the pulse train input. The others are clocked from the Q outputs of the previous stages. It is this connection that causes the ripple through action. The circuit of a three-stage counter of this type in shown in Figure 5. Since all flip-flops are connected in the toggle mode, each divides by two so the overall division ratio is 2^3. In general, for a counter using 'n' flip-flops the division ratio (also known as the scale or modulo) is equal to 2^n. Another feature of this counter, being a practical circuit, is the 'reset' line. It has nothing to do with the reset mode in line two of the JK truth-table. Instead it is a separate pin provided to reset the flip-flop; it has to be taken to logic 0 to enable this to happen. Such a connection is said to be 'active low' or, alternatively, 'negative acting'.

To investigate and learn something of the operation of this circuit, it is recommended that it is connected up, using either the TTL oscillator or de-bounced switch as input, and with LEDs on all the Q outputs. The counter should be reset to start with, so that all LEDs read 0. Pulsing the circuit through the complete sequence should take it from 000 to 111, eight states all told. From the point of view of the circuit diagram, the binary number shown by the LEDs is actually 'backwards', since the first flip-flop FFA is the Least Significant Bit (LSB) of the number. Having satisfied yourself that it does count in binary, the next step that can be tried is to connect the LEDs to the $\overline{Q}$ outputs instead. Since these are complementary to the Q outputs, the sequence will be reversed, i.e. the counter will count 'down' from an initial value of 111 to a final value of 000. Thus, the counter can be used to count up or down just by the choice of where the outputs are taken from, either Q or $\overline{Q}$.

There is a small modification that can be made to this circuit that is worth looking into - especially as it is very easy to do. Leave FFA clock input as it is but connect the clock inputs for FFB and FFC to the $\overline{Q}$ outputs of the previous flip-flops instead of the Q outputs. Record the sequences at both the Q and $\overline{Q}$ outputs, as was done before. You might like to ask yourselves whether the results are what you expected.

Binary Synchronous Counters

A disadvantage of asynchronous counters is their slowness. It should be evident that they will be slow because it is necessary to allow all flip-flops to change state before a new clock pulse can be applied. Try to clock the circuit too quickly and it becomes confused. In a synchronous counter all clock inputs are clocked from a common source, the pulse train input. This presupposes that the counter knows in advance when certain flip-flops should change state and when they shouldn't. What is not wanted is the whole lot changing whenever a clock pulse appears. It would be somewhat tedious, not to say time consuming to go through the design method of synchronous counters. Suffice it to say that some extra gating is usually needed (quite a lot sometimes) in order to sort out when any given flip-flop should change state in the sequence. From the hobbyist's point of view, it is sufficiently interesting to hook up the circuit (shown in Figure 6) and pulse it to see that it does indeed work. However, it can be taken a stage further.

While the counter is being pulsed, a logic probe can be held on the J and K inputs of FFC (that is the output of the AND gate). The output of this gate can only be either logic 0 or logic 1, putting FFC into either the 'no-change' or 'toggle' modes. Determine when it is in one mode or the other and why. Investigation of this type teaches one a lot. Try to think of other aspects of the circuit to investigate. You may well have realised that this counter is not actually a full synchronous counter. Only FFA and FFC are truly synchronous; FFB is clocked from the output of FFA. Such a circuit is a compromise between speed and simplicity.

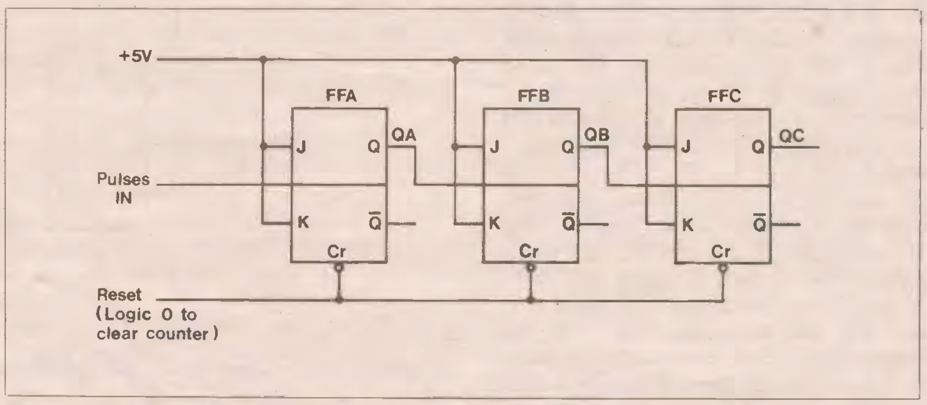


Figure 5. Scale-of-eight asynchronous (ripple-through) counter.

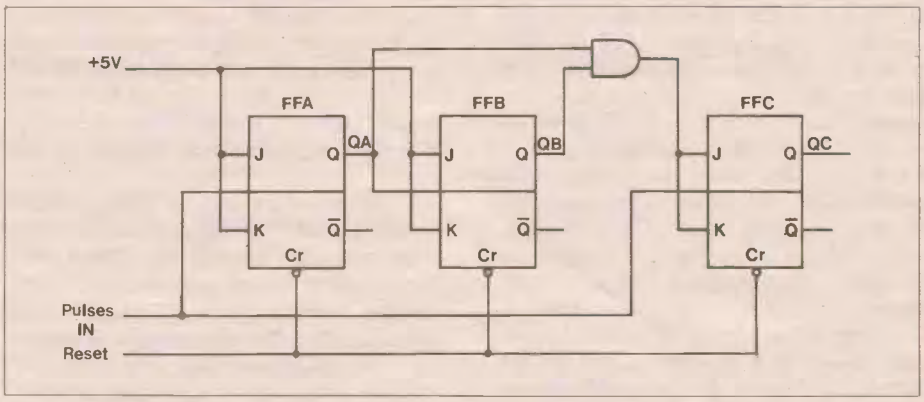


Figure 6. Scale-of-eight synchronous counter.

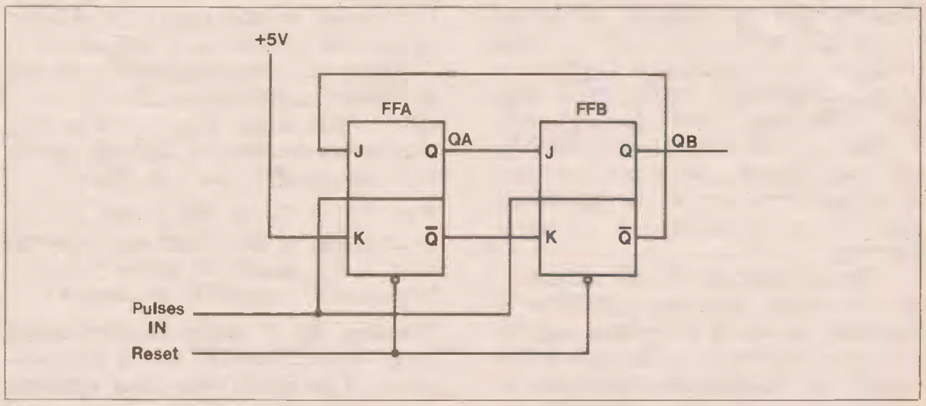


Figure 7. Scale-of-three feedback counter.

Counters to other Bases

The basic binary dividing element, the JK flip-flop, is only capable of division by two, yet there are times when division by numbers that are not a power of two is needed. As an example of this, consider the simple non-binary counter of Figure 7. This is a 'scale-of-three' counter, the three states being 00, 01 and 10; the state 11 is avoided. The circuit is forced to reset to 00 on the fourth pulse by means of the feedback between the $\overline{Q}$ output of FFB and the J input of FFA. The action of this feedback is as follows.

Assuming that the counter is initially reset, so that its first state is FFA = 0, FFB = 0, the $\overline{Q}$ output of FFB will be at logic 1 which is fed back to the J input of FFA; the K input of FFA is wired to logic 1 anyway, so FFA is in the toggle mode. The J and K inputs of FFB are fed from the complementary Q and $\overline{Q}$ outputs of FFA, so that FFB will always be in either the SET mode or the RESET mode; it has no other choice. The first clock pulse will cause FFA to toggle, its Q output going to logic 1; this same pulse will have no effect on FFB since its inputs, at the moment of clocking were J = 0 and K = 1 (RESET mode) and the flip-flop is already reset. The state after the first clock pulse is, therefore, FFA = 1 and FFB = 0. But after this first clock pulse, FFB's J and K inputs will have reversed because of the toggling of FFA and we can anticipate that FFB will become SET after the next clock pulse. Sure enough, this is what happens, FFA toggling at the same time, so that the state after the second clock pulse is FFA = 0 and FFB = 1. This is where the feedback comes in. The $\overline{Q}$ output of FFB has now gone down to logic 0; this means that FFA is no longer in the toggle mode, but in the reset mode (J = 0, K = 1). What is the mode of FFB? Look at its J and K inputs, J = 0 and K = 1. This means that it is also in the reset mode. Therefore, after the next pulse, FFA 'stays' reset, FFB 'becomes' reset and the final state of the counter is also its initial state for a new sequence, namely 00.

Taking the idea of feedback further and combining it with gating, we get the circuit of Figure 8, the 'scale-of-five' counter. If you followed the discussion about the 'scale-of-three' counter, you shouldn't have much difficulty in proving to yourself how this one works. However, there's nothing like putting theory into practice so it is suggested that you wire up this counter, reset it and follow it through its sequence (000, 001, 010, 011, 100, 000, etc). Make use of LEDs at the Q outputs and a logic probe on the gate inputs and output. You should find it a useful exercise.

The 'scale-of-five' counter converts to an even more important type if it is preceded by a single JK flip-flop wired in the toggle mode, as shown in Figure 9. The result of the two successive divisions is to produce an overall result of 'divide-by-ten', that is a decade counter. This has obvious practical applications. It is worth wiring this up, especially if you've already breadboarded the 'divide-by-five' counter. Then try reversing the order of the two component parts, that is drive the 'divide-by-two' section from the 'divide-by-five' circuit. Do you think the order will matter? You may (or may not) be surprised at the result!

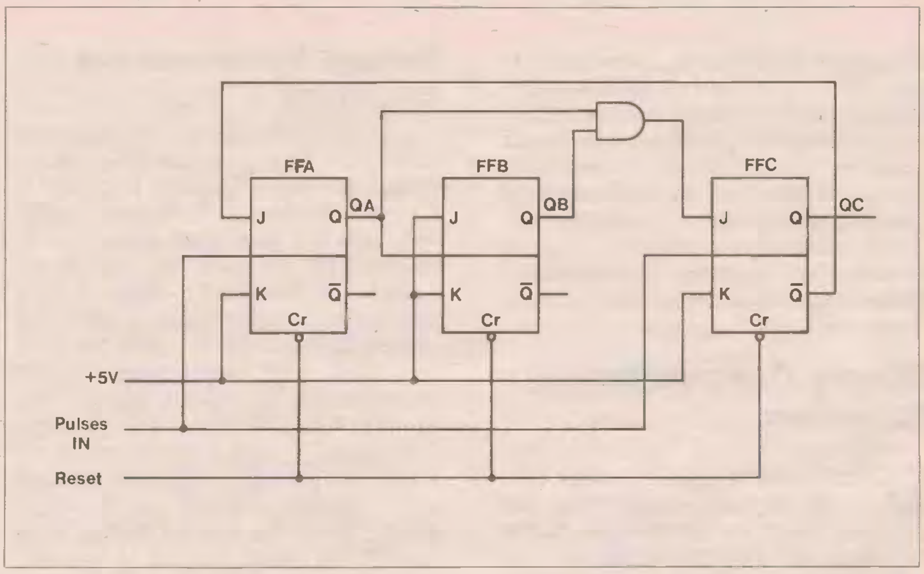


Figure 8. Scale-of-five feedback counter.

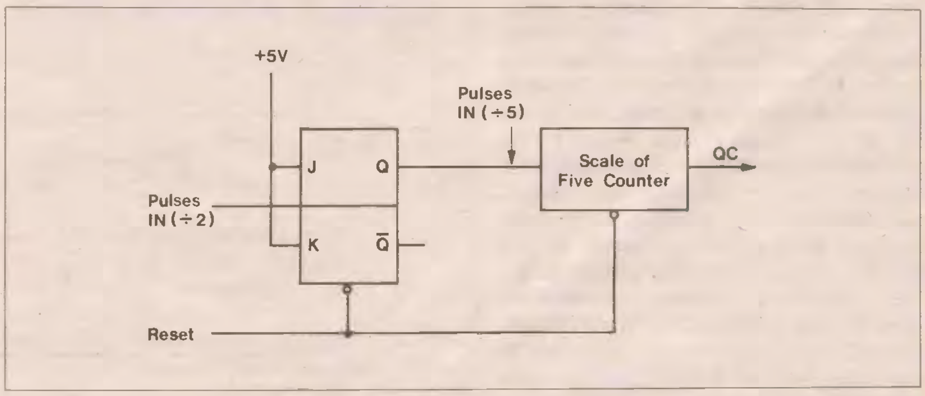


Figure 9. Decade counter constructed from basic divide-by-two stage and the circuit of Figure 8.

Of course, if you really want to build a circuit around a decade counter, you don't have to build it up in this way. There are a number of single-chip circuits, of which the one shown in Figure 10, the 7490, is an example. This actually contains the separate 'divide-by-two' and 'divide-by-five' circuits, which can be accessed separately or wired in series by connecting pin 12 (Q_1) to pin 1 (pulses in for 'divide-by-five'). Pins 2 and 3, or 6 and 7 can be used to set up an initial state of 0 or 9 (0000 or 1001).

Another interesting circuit is the so-called 'programmable' series of counters, 74160-3, shown in pin-out form with sample waveforms in Figure 11. A resumé of the circuit operation is as follows:

Features: A 16-pin DIL IC that can be programmed to start at any required initial state in the binary sequence 0-15 and terminate at any point in the sequence.

Clearing the Counter: Some counters clear 'asynchronously', that is independently of the clock; some clear 'synchronously', that is on the next positive clock transition.

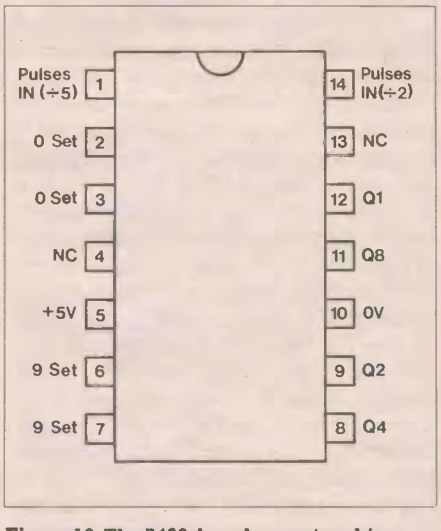


Figure 10. The 7490 decade counter chip.

Programming the Counter: 'Data in', corresponding to the required initial state (ABCD) is applied to pins 3-6. The 'load' line is taken low and the counter presets on the next positive clock edge.

Enabling and Disabling the Counter: Two control lines, P and T, are provided. They are both taken high to start the counter. Taking either low will disable (inhibit) the counter.

Carry 'look ahead' output: Used for successive cascading of stages without extra gating being needed. A carry pulse

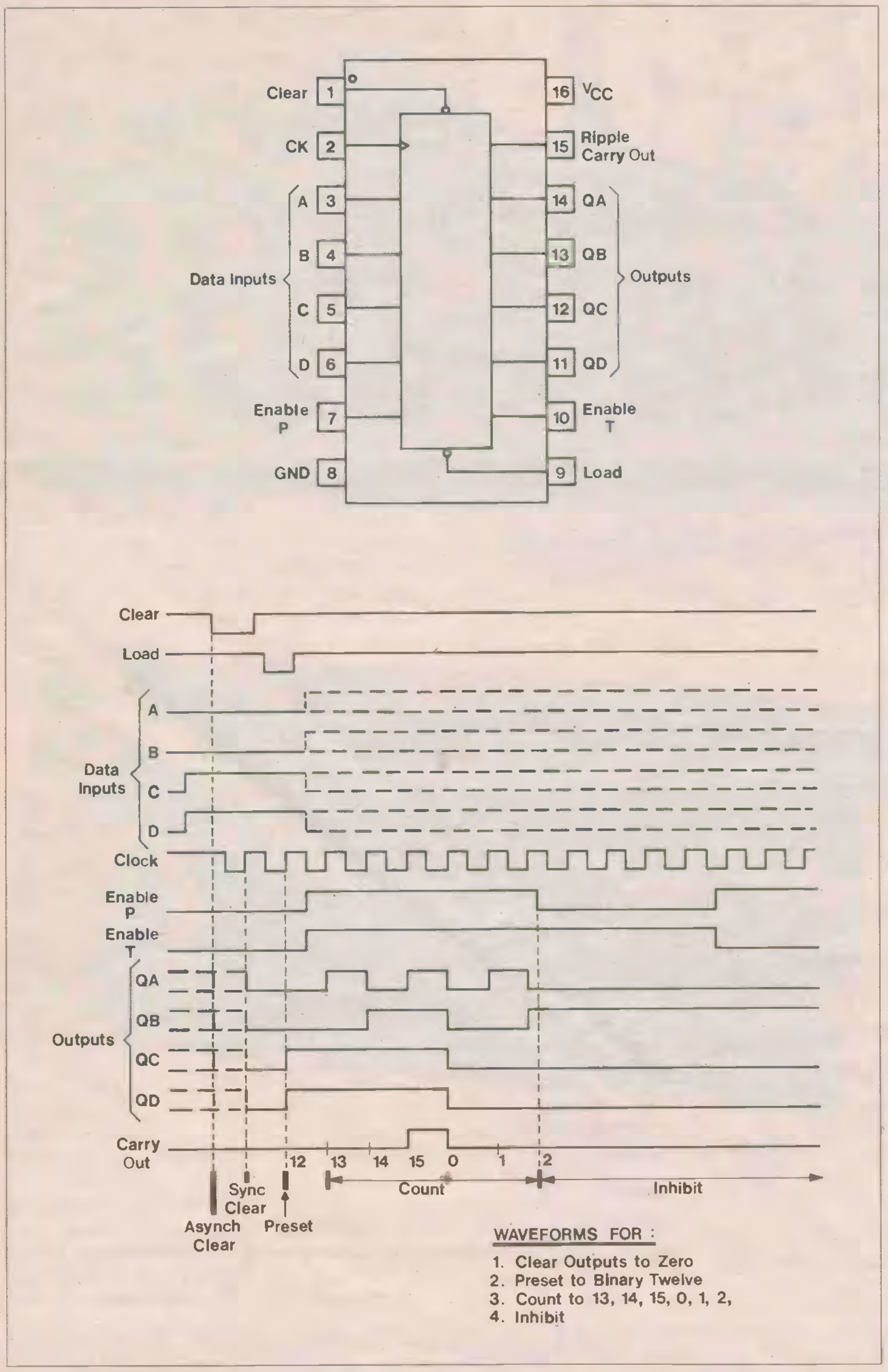


Figure 11. The SN74160-3 programmable counters.

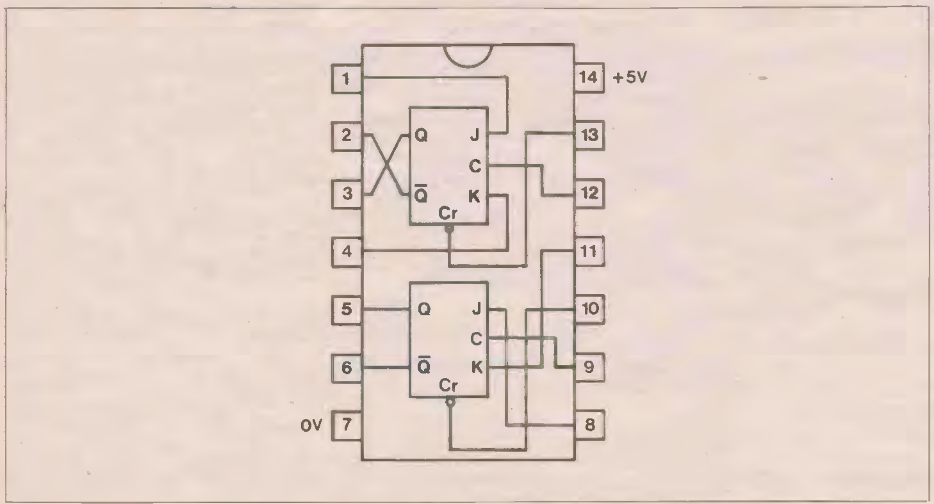


Figure 12. Pin-out for 74107 dual JK level-triggered flip-flops.

is generated and is enabled by the T enable line (must be high).

The set of waveforms of Figure 11 should help to make the operation clear. A negative-going pulse is required to clear the counter, the asynchronous and synchronous 'clears' being shown at the bottom of the diagram. The load pulse is also negative going and, at its leading edge, the 'data in' (binary 12, i.e. 1100 in order DCBA) is preset into the counter (Q_D to Q_A) - see output waveforms lower down. The clock pulse is a regularly recurring square wave that governs the counting rate - the changes at Q_D to Q_A, 1100, 1101, 1110, 1111, etc., can be seen quite clearly to occur at the leading edges of the clock pulses. The action of the enable lines, P and T, is also seen. During the count period both are high but, at a count of 2, the P line goes low, stopping the count. If both were taken high at any subsequent instant, counting would resume from the last value. In fact, in the example shown, when P goes high again, T goes low, so the counter remains inhibited. The point that is made is that the operation of the counter is under the control of two independent lines, P and T, and how these are actually used is up to the individual.

There should be enough practical work there to while away a few evenings. Building and testing the circuits described will help in gaining a good understanding of a subject that is often imperfectly understood. As already said, a similar look at registers will be the subject of the next part of this series.

Appendix - Problems with TTL

Because TTL gates often use a type of output stage known as a 'totem pole', it is possible to generate, in normal use, short duration current spikes while conduction is changing over from one half of the totem pole to the other, a process that occurs when the output switches between logic levels. Thorough decoupling of TTL circuits is, therefore, advised. The following guidelines are suggested to cure or avoid problems of this type.

(i) Use one 10nF - 100nF short-lead disc ceramic capacitor across the supply lines for every four gate packages.

(ii) Use one similarly for every two MSI packages (e.g. counters and shift registers).

(iii) Use a separate such capacitor for every package that is further away than 3" (75mm) from the nearest bypass capacitor.

(iv) Use a 10μF 6V tantalum electrolytic capacitor where the +5V supply lines enter the board.