The foolproof way to sequencer design.

Lockups, race conditions and unwanted states are eliminated when you follow this step-by-step approach.

Logic designers often spend frustrating hours trying to make hard-wired sequential logic circuits work. Hit-and-miss design techniques lead to system lockups, race conditions and other undesirable performance.

A systematic approach to sequencer design guarantees reliable performance. If you follow it, even a complex sequencer, with a number of branches and jumps, should present no problems. The approach is based on five broad steps before you get into the circuit design. Then, when you tackle the circuit, there are six rules to keep in mind.

Start with these steps:

1. Determine the required sequence. This depends naturally on the application.
2. Develop a flow chart for the sequence.
3. Rearrange the flow chart into a control-loop chart.
4. Prepare a Karnaugh map based on the control-loop chart.
5. Write equations for each Karnaugh state.

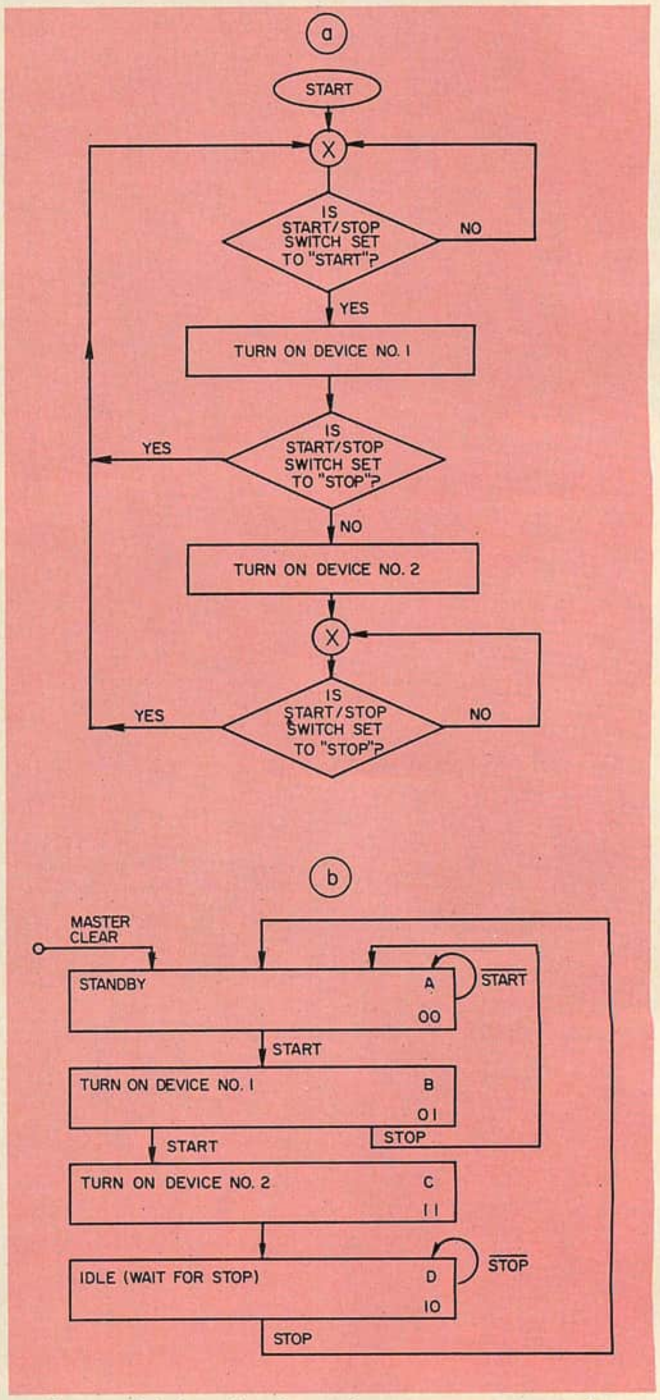
The flow chart in (Fig. 1a), drawn up the way a software programmer would do it, is an example of a simple sequencer. From the chart, we see that the program continues to loop until the Start/Stop switch is set to Start.

Then the program allows device No. 1 to turn on. If the Start/Stop switch is not immediately set to Stop, the program calls for turning on device No. 2. Thereafter, when the Start/Stop switch is set to Stop, the sequence returns to the beginning.

The logic designer, however, interprets the circuit's requirements as a control-loop sequence (Fig. 1b). This chart defines four discrete states that perform the sequence. The sequencer remains in State A until the Start/Stop switch is set to Start. Then the sequencer goes to State B and turns on device No. 1 and so on.

Once the control-loop chart has been drawn, the information is transferred to a Karnaugh map to show the inter-relationships of all the

James H. Bentley, Principal Development Engineer, Honeywell, 600 Second St. N., Hopkins, Minn. 55343.


a
START
X
IS START/STOP SWITCH SET TO "START"?
NO
YES
TURN ON DEVICE NO. 1
IS START/STOP SWITCH SET TO "STOP"?
YES
NO
TURN ON DEVICE NO. 2
X
IS START/STOP SWITCH SET TO "STOP"?
YES
NO
b
MASTER CLEAR
STANDBY A 00
$\overline{START}$
START
TURN ON DEVICE NO. 1 B 01
STOP
START
TURN ON DEVICE NO. 2 C 11
IDLE (WAIT FOR STOP) D 10
$\overline{STOP}$
STOP

1. **Sequential functional requirements,** as the software programmer sees them, form a flow chart (a), and as the logic designer needs them, with the logic states defined, become a control-loop chart (b).

states. After the Karnaugh state equations are written, we are ready to draw the logic diagram.

Follow the design rules

Here are the six rules for the circuit design:

1. Use only J-K type flip-flops to store the needed sequence-event codes. The number of codes equals 2^n, with n the number of flip-flops needed.

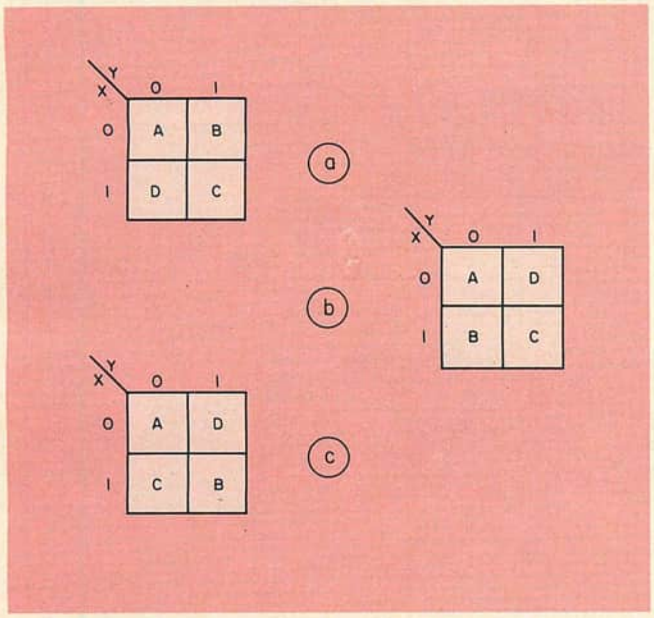
2. Always make the standby condition an all-ZERO event code. The Master-Clear signal must force the Sequence-Event control J-K flip-flops to standby.

3. Never permit more than one Sequence-Event flip-flop to toggle at a time when changing to the next event code.

4. Use a synchronous clock system, and strobe the Sequence-Event flip-flop changes of state with a clock pulse.

5. Store the occurrence of each external asynchronous event in any convenient type of flip-flop. Transfer the information to a D-buffer flip-flop in the time between the Sequence-Event strobe pulses. Clear both the external-store and the D-buffer flip-flops during a subsequent state time.

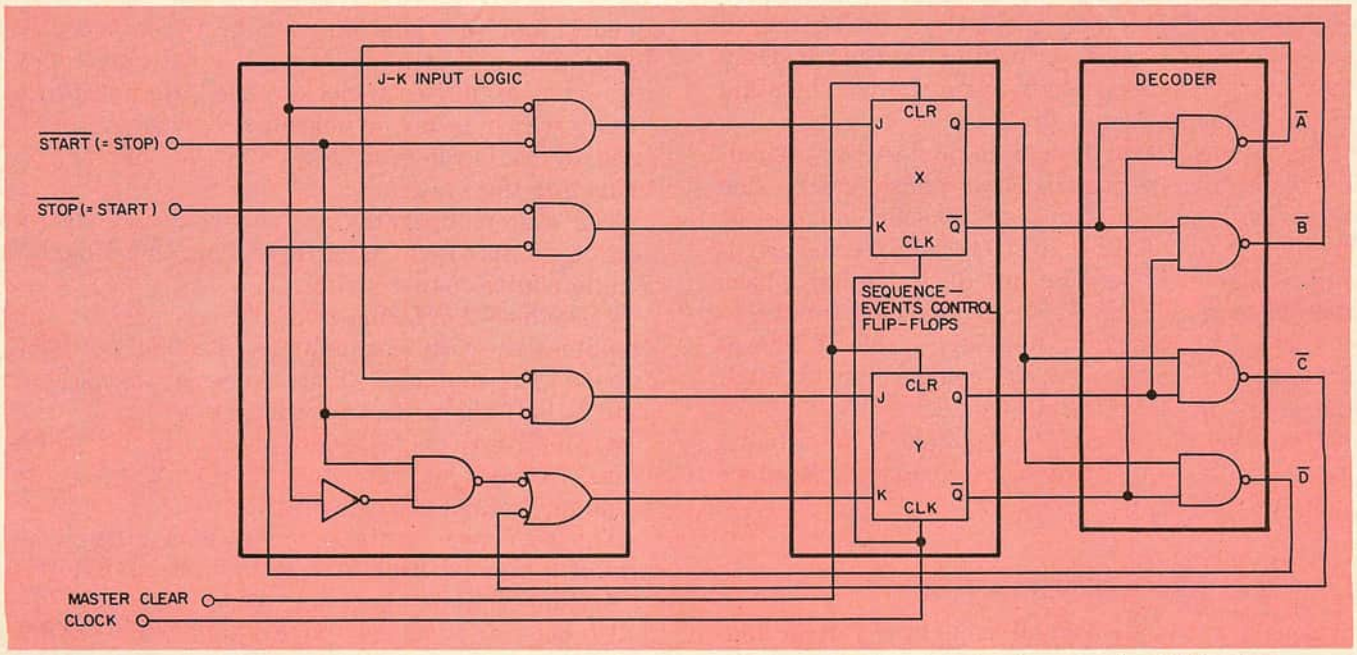
6. Force *completely* unused states to the standby condition. But some states, though they are not used to define a working state, may be needed to carry out Rule 3. Such unused states are called Idle states.



2. **When the designer transfers the data to a Karnaugh map,** he must strictly observe the principle of adjacency, as in maps a or b. There are four other configurations that are not permissible. One example is map c.

In the simple sequencer defined in Fig. 1 only four states are needed. Fig. 2a represents the conditions of two J-K Sequence-Event flip-flops, X and Y which are sufficient to establish four states (Design Rule 1).

State A is assigned to Standby, since the standby state must always be all-ZERO, in accordance with Rule 2. Furthermore, Rule 3 requires that states adjacent to each other in the control loop be separated by only one bit change. Thus they also are adjacent to each other on the Karnaugh map. The term "adjacent" means that as the sequence proceeds from one state to the next via an allowed path of the control loop, one —and only one—J-K flip-flop may toggle.



3. **Three basic circuits**—the J-K Input logic, the Sequencer-Event logic and the Decoder—implement the requirements for the simple example of Fig. 1. A more complex system would need an External-Events circuit.

Fig. 2b is another acceptable Karnaugh map for the control-loop chart. All other arrangements (four more are possible) are unacceptable, since they do not comply with the adjacency rule. In the unacceptable cases two or more control flip-flops are required to simultaneously toggle to produce a change of state. Since flip-flops rarely toggle at identical speeds, even when clocked by the same clock pulse, many undesired effects can occur.

The control-loop chart and its corresponding Karnaugh map require that flip-flop Y toggles to a ONE state for the sequencer to proceed from state A to B. Therefore the steering input, J, of flip-flop Y—designated JY—must equal ONE. This happens when the sequencer is in state A and the Start/Stop switch goes to Start. The equation for this is

$$JY = A \cdot \text{Start}.$$

The subsequent state equations are

$$JX = B \cdot \text{Start}.$$
$$KY = C + B \cdot \text{Stop}.$$
$$KX = D \cdot \text{Stop}.$$

Note that in the third equation, $KY = 1$ each time the sequencer is in state C. This results in an immediate jump to state D. However, when the sequencer is in state B, and the Start/Stop switch is set to Stop, $KY = 1$ again. This resets the Y flip-flop, but now the system goes to state A.

The final step is to draw the logic diagram. An implementation with NAND/NOR logic is shown in Fig. 3. Note that the Master-Clear pulse sets the sequencer to the standby condition, state A.

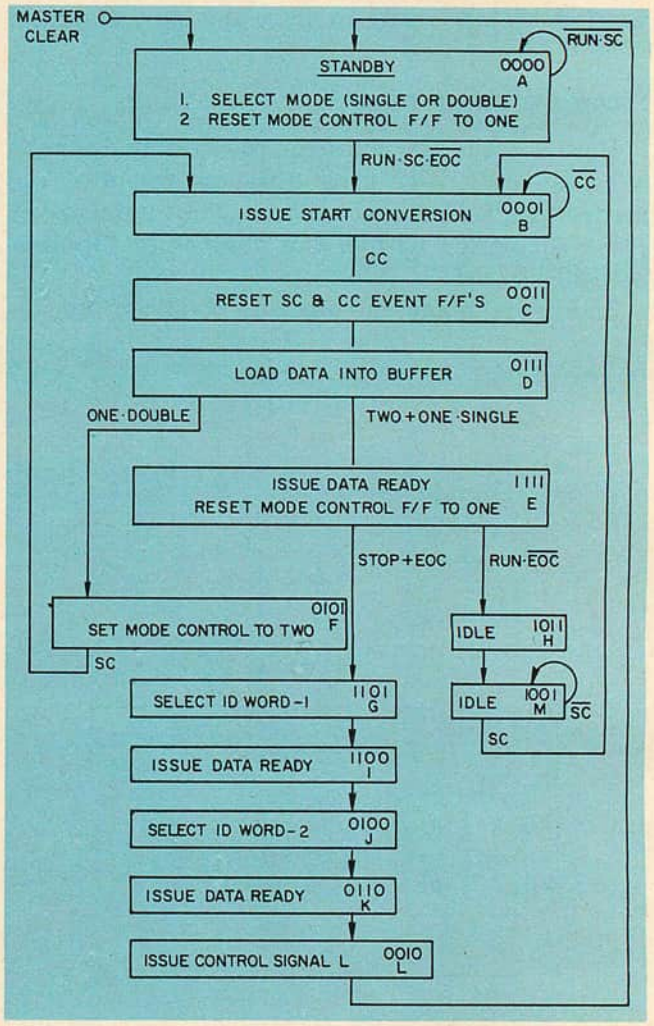
Note also that Rule 4 requires Sequence-Event state changes to occur synchronously with a free-running clock—not at the time some external event occurs. The external occurrences must first establish and stabilize themselves or be stored between clock pulses, according to Rule 5. Only then should the next clock pulse change the state of the Sequence-Event flip-flops.

But to implement Rule 5, logic designers usually use a nonoverlapping two-phase clock. The Sequence-Event flip-flops are clocked on one of the phases, say Phase 2, after the external asynchronous events are set up at the other phase time, Phase 1. This assures that all inputs to the Sequence-Event flip-flops are stable at Phase-2 time. The clock rate, of course, must match the needs of the external events.

However, the preceding example is too simple to adequately illustrate Rules 5 and 6. A more complex system is needed.

Examining a more complex system

Consider a mode-control sequencer for an analog-to-digital converter. The a/d converter samples inputs from an external device at a fixed frequency and provides outputs to a computer in bytes. The a/d-conversion rate is determined by an external device's clock, called Signal Clock (SC), which is not synchronized with the clock used by the sequencer. Fig. 4 is the control-loop chart for the system.



MASTER CLEAR
$\overline{RUN \cdot SC}$
STANDBY 0000 A
1. SELECT MODE (SINGLE OR DOUBLE)
2 RESET MODE CONTROL F/F TO ONE
$RUN \cdot SC \cdot \overline{EOC}$
$\overline{CC}$
ISSUE START CONVERSION 0001 B
CC
RESET SC & CC EVENT F/F'S 0011 C
LOAD DATA INTO BUFFER 0111 D
ONE·DOUBLE
TWO+ONE·SINGLE
ISSUE DATA READY 1111
RESET MODE CONTROL F/F TO ONE E
STOP+EOC
$RUN \cdot \overline{EOC}$
SET MODE CONTROL TO TWO 0101 F
SC
IDLE 1011 H
SELECT ID WORD-1 1101 G
IDLE 1001 M
$\overline{SC}$
SC
ISSUE DATA READY 1100 I
SELECT ID WORD-2 0100 J
ISSUE DATA READY 0110 K
ISSUE CONTROL SIGNAL L 0010 L

4. **A control-loop chart** for an a/d control system illustrates the use of idle states to attain a single-bit change at a time, when going from one state to another. Idle states, H and M, serve this function between B and E.

The system operates in two modes. Mode selection is handled, as follows, by the Double/Single control-panel switch:

- Double—Performs two a/d conversions and combines the two output bytes into one word for input to the computer. Thus the word throughput rate is half the a/d conversion rate.
- Single—Provides each byte as a single word. Thus throughput rate to the computer is the same as the a/d conversion rate.

The sequencer system's conditions under control of a manual Run/Stop switch are as follows:

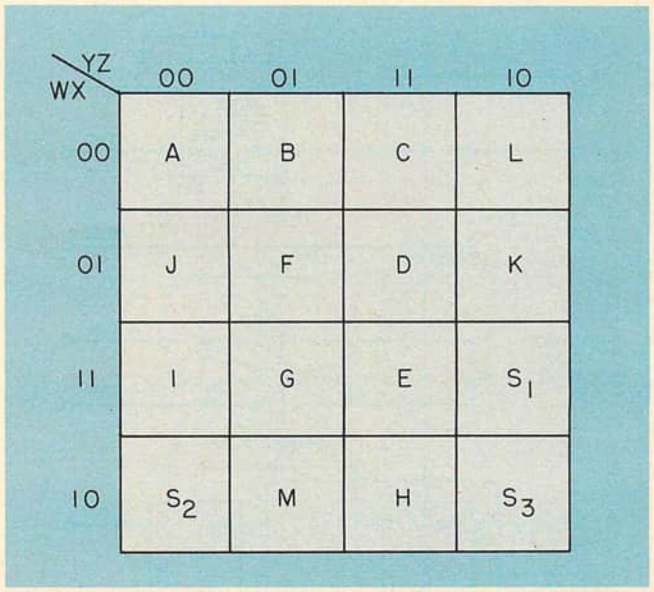
- Run—Starts operating when the next SC pulse occurs, provided the End-of-Count (EOC) condition is inactive (low condition). If the EOC condition occurs before the Run/Stop switch is set to Stop, the sequencer cycles through the shutdown routine described under Stop. EOC comes from the computer to indicate that the computer is not ready for a new set of data.

- Stop—Completes transmitting to the computer the final a/d word together with two ID words and returns to standby.
- Busy—Generates a Conversion-Complete (CC) signal when the a/d converter's Busy goes from high to low. The Busy remains high for 5 μs after the Start-Conversion signal to the converter resets. Upon completing the CC function the sequencer resets CC prior to the next SC pulse.
- ID Word—Replaces a/d data words with ID-1 and ID-2 words after an EOC or Stop condition.
- Data Ready—Issues a Data-Ready signal to the computer when data are available on the computer input bus.

With the functions defined, charted (Fig. 4) and mapped (Fig. 5), you can now follow the six design rules. Note that this more complex example requires 13 states. But three control flip-flops are not enough, and four control flip-flops provide 16 states—more than needed. You must now apply Rule 6 to handle the unused states.

Treating unused and idle states

If possible, unused states should be avoided. Sometimes a judicious rearrangement of the states on the Karnaugh map can reduce the number of control flip-flops needed. An optimal arrangement may require several trials. On the other hand, the need for future growth and change may make providing for unused states the wiser choice.

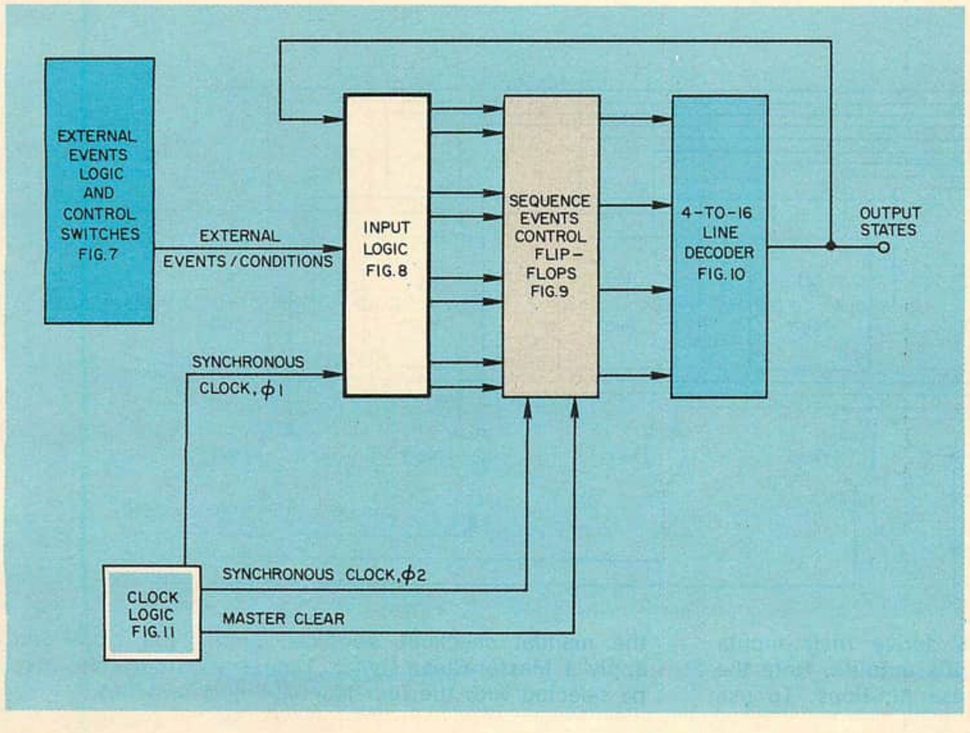


WX \ YZ	00	01	11	10
00	A	B	C	L
01	J	F	D	K
11	I	G	E	S_1
10	S_2	M	H	S_3

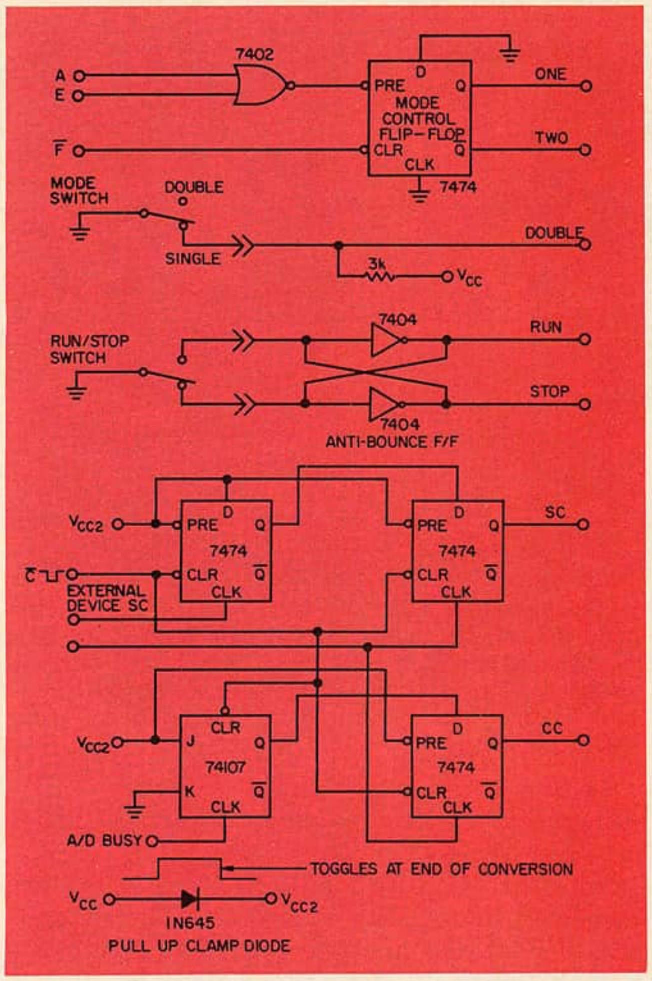
5. **The Karnaugh map** for the a/d controller must include the unused states, S_1, S_2 and S_3. These states must also appear in the J-K input equations. Should the system fall into one of these unused states, the circuit must automatically transfer to Standby (state A) on the next clock pulse.

Note that states S_1, S_2, and S_3 (Fig. 5) are completely unused. States H and M, however, are labeled Idle; they are used only to go from State E to B, in single steps as required by Rule 3. The J-K Sequence-Event flip-flop input equations must include all the states, and this means the unused states too. The complete set of input equations are:

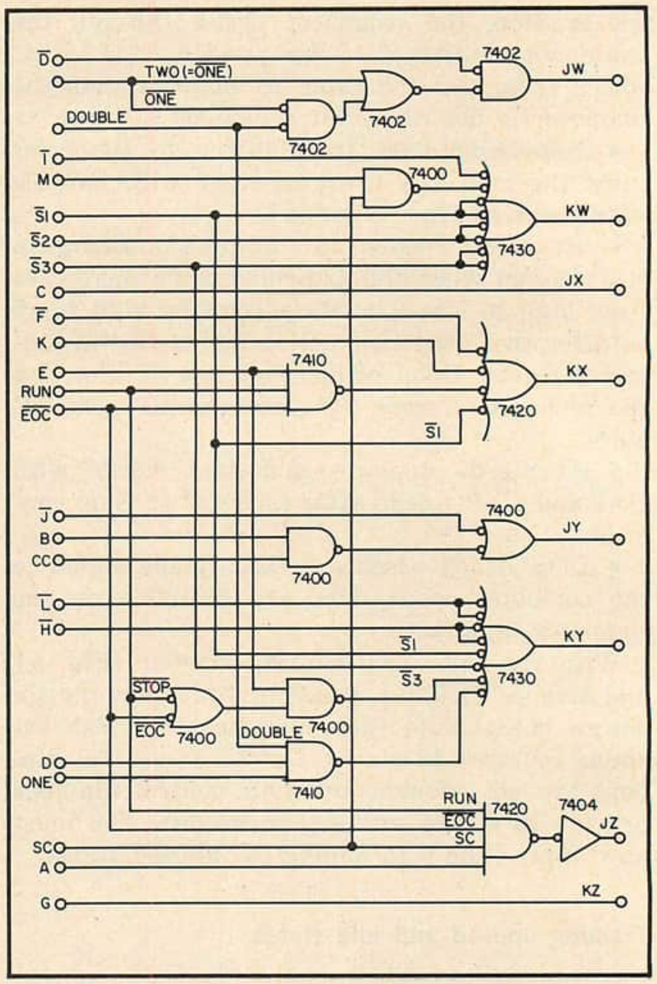
$$JW = D \cdot (TWO + ONE \cdot SINGLE)$$


EXTERNAL EVENTS LOGIC AND CONTROL SWITCHES FIG.7
EXTERNAL EVENTS / CONDITIONS
INPUT LOGIC FIG.8
SEQUENCE EVENTS CONTROL FLIP-FLOPS FIG.9
4-TO-16 LINE DECODER FIG.10
OUTPUT STATES
SYNCHRONOUS CLOCK, φ1
SYNCHRONOUS CLOCK, φ2
MASTER CLEAR
CLOCK LOGIC FIG.11

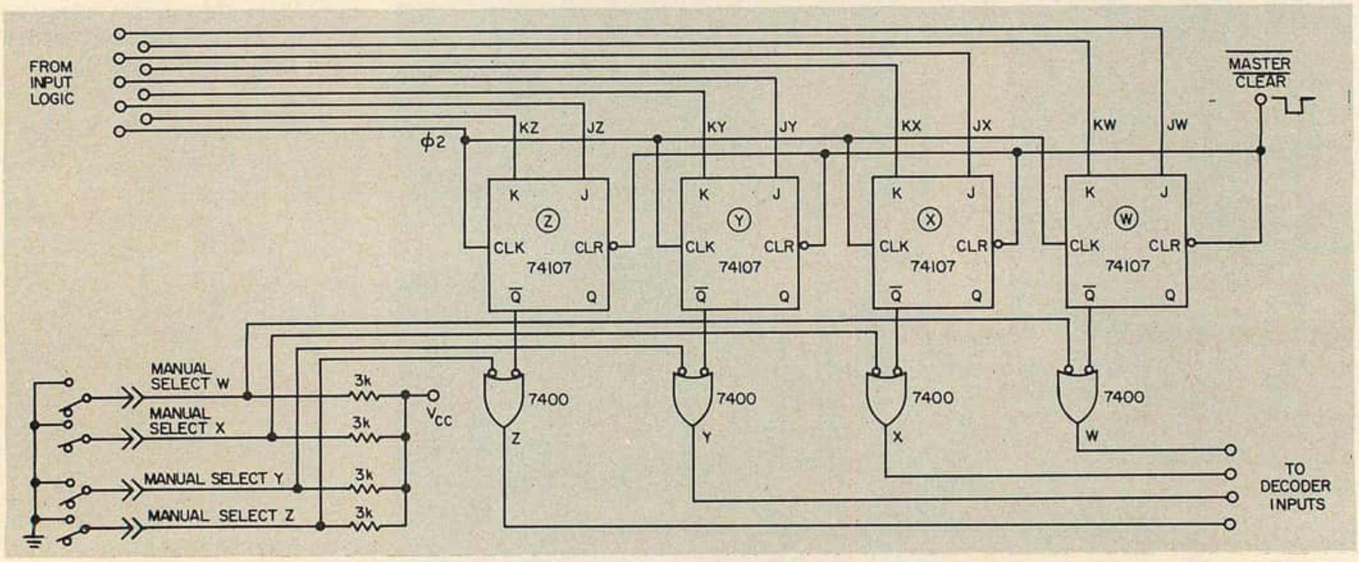
6. **The a/d controller block diagram** has two circuits not needed in the simple example of Fig. 3—the external-events logic and a two-phase clock.



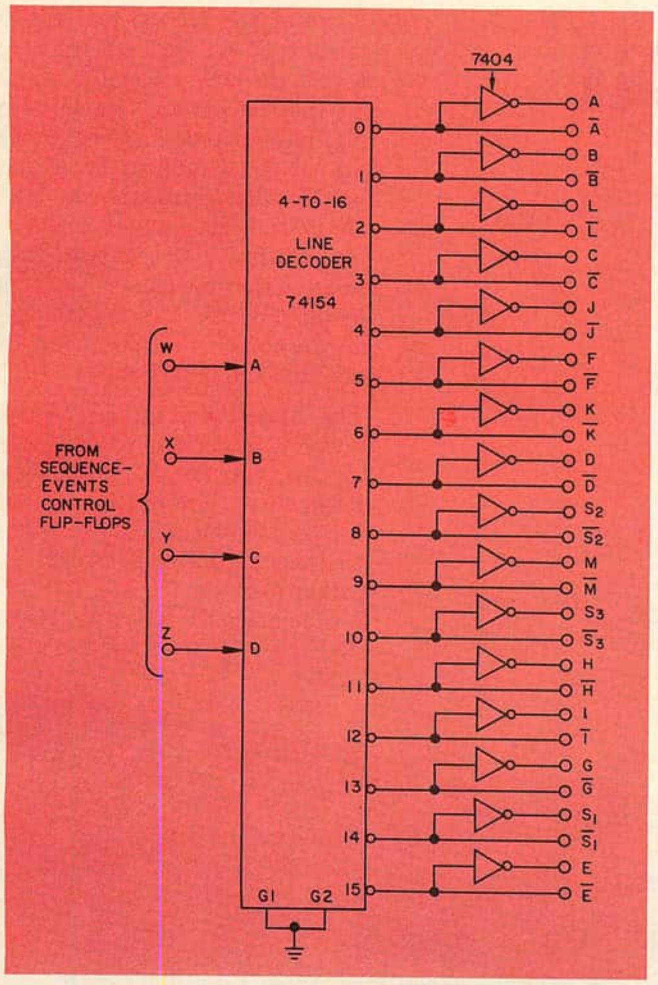
7. **External events** such as the external device signal clock (SC) and the a/d Busy signal, must be temporarily stored and then strobed synchronously into the sequencer by clock $\phi 1$. The four external-events store flip-flops are cleared when state C occurs. The external EOC signal goes directly to the input logic circuits from the computer, since it always occurs and stabilizes before it is needed by the sequencer.



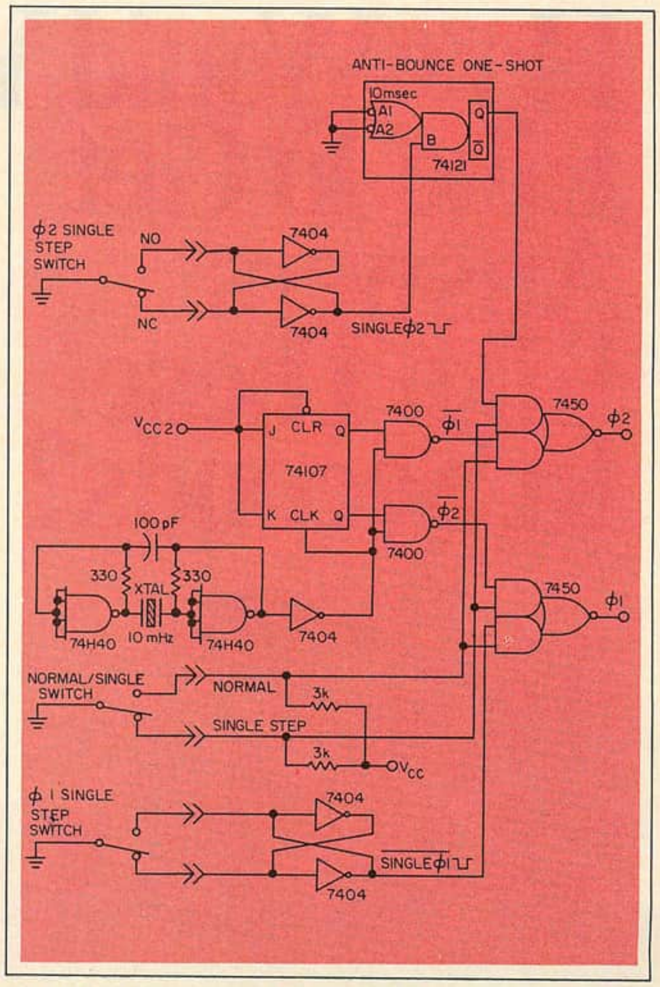
8. **The Input-Logic circuit implements the equations** for the J-K steering inputs of the Sequence-Events flip-flops. Note that most of the input variables to the gates come from the decoder's output, which represents the sequencer's present state. The present state, together with the signals from external events and from the control switches, determines the system's next state. Changes in the system would be made mainly in this circuit.



9. **The Sequence-Events flip-flops** derive their inputs directly from the Input-Logic circuit's outputs. Note the extra features for checkout of these flip-flops. To use the manual checkout switches, inhibit the clock and apply a Master-Clear signal. Then any output state may be selected with the four Manual-Select switches.



10. **The simplest part of the sequencer is the decoder.** Inputs A, B, C and D are tied to control flip-flop outputs W, X, Y and Z. Though all available states and their complements may not be used in an initial design, inevitable design changes make it prudent to include the full capability at the outset. Thus changes can be confined to the Input-Logic circuit. The decoded outputs are assigned from the Karnaugh map of the system.



11. **A two-phase clock** is required for operation of the sequencer. The frequency of the master oscillator must be high enough to service the asynchronous external inputs. In this example, the clock frequency is determined by the 5 μs conversion speed of the a/d converter. A 10 MHz frequency provides an ample margin. And for checkout purposes, three control-panel switches provide a single-step capability.

$$
\begin{aligned}
KW &= I + M \cdot SC + S_1 + S_2 + S_3 \\
JX &= C \\
KX &= F \cdot SC + K + E \cdot RUN \cdot \overline{EOC} + S_1 \\
JY &= B \cdot CC + J \\
KY &= E \cdot (STOP + EOC) + L \\
&\quad + D \cdot ONE \cdot DOUBLE + H \\
&\quad + S_1 + S_3 \\
JZ &= A \cdot RUN \cdot SC \cdot \overline{EOC} \\
KZ &= G
\end{aligned}
$$

Implementing the a/d control

The logic designer must not leave out any terms. To check on the number of terms, count the lines leaving each state in the control-loop chart. The total must equal the number of terms in the input equations. In this example, the count is 15. This count does not include the three unused states S_1, S_2 and S_3, since they do not appear on the control-loop chart.

In accordance with Rule 6, these three unused states are forced back to state A—Standby—on the next clock pulse.

The block diagram of the a/d controller is shown in (Fig. 6). Self-explanatory circuit details are covered in Figs. 7 through 11. And beyond bare-bones requirements, the sequencer includes some refinements, such as self-checking and some peripheral device checking. Note the use of a two-phase non-overlapping clock. All the logic used in this example belongs to the TTL-7400 series.

Besides providing a fool-proof approach to sequencer design, you will find that this method is very adaptable. To make design changes, usually only the Input-Logic circuit need be altered; the remaining circuits, such as the Sequence-Event flip-flops and decoder, remain untouched. A few minutes work with a soldering iron or wrapping gun completes the changes. ■■