Chapter 9
Scoring and Timekeeping

Virtually all fast-action video games call for automatic scoring, and of course it is nice if timed games have some provisions for displaying the elapsed time or time remaining for the play. The circuitry is practically identical in either case, a control circuit that generates binary numbers for scoring or time and a display circuit that generates the appropriate numeric figures on the screen.

The control and display circuits described in this chapter can be retrofit to most of the TV games already outlined in earlier chapters, and they can be fit into most of the game systems developed in the remainder of this book. After looking at the basic theory and suggesting some experiments, you will thus find some suggestions for expanding games you have already built to include scoring and timekeeping, or both.

Games described in the closing chapters of this book will include scoring and timekeeping boards as options. What is most important, however, is that you understand the circuitry thoroughly, making it possible to apply them effectively and efficiently to game designs of your own.

GENERATING NUMERIC CHARACTERS

The numerals used for these video games are built around the familiar 7-segment display that characterizes modern electronic calculators and digital clocks. Figure 9-1a shows this basic 7-segment format with the standard lower-case letter designations.

The a segment is always the one across the top. The alphabetical sequence then proceeds clockwise through f. Letter g is then reserved for the horizontal segment through the center of the figure.

The numeral 1 is thus generated by lighting segments b and c only. A 2 is generated by lighting segments a, b, g, e, and d, while a 3 is made up of a, b, c, d, and g. If you have never done so before, work your way through the standard 7-segment format to see how any numeral between 0 and 9 can be generated.

Seven-segment display characters (whether in a TV game, calculator, or digital clock) are normally originated by a BCD word, a binary-coded version of decimal numbers between 0 and 9. The table in Fig. 9-1b compares these 4-bit BCD words with their decimal counterparts and the segments in a 7-segment display that ought to be lighted or extinguished.

As mentioned in an earlier example, the 3 character is displayed by lighting segments a, b, c, d, and g. Note from the table that the BCD version of the number 3 is 0011. The corresponding 7-bit segment output in this particular case is 1111001, where the most-significant bit is the a segment and the least-significant is the g segment. The 1s in the segment word represent a lighted segment, and the 0s stand for a darkened segment—one that is not shown.

Study the table in Fig. 9-1b carefully, comparing the 10 BCD inputs with the segment equivalents and the 7-segment format in Fig. 9-1a.

The table in Fig. 9-1b is actually the truth table for a standard BCD-to-7-segment converter IC device. This 7448 device accepts BCD inputs A through D (a being the least-significant bit) and generates the corresponding 7-segment format at outputs a through g. Input G can be viewed as an enabling input that must be at logic 0 in order to enable the BCD-to-7-segment conversion process. Setting input G to logic 1 sets all segment outputs to logic 0, thereby blanking the figure altogether.

Assuming for the moment that you have a source of BCD numbers and a 7448 converter for translating those numbers into a 7-segment format, turn your attention to the drawing in Fig. 9-2. This figure shows how the segments are arranged for a display on the TV screen. The a segment is a bar that crosses the top of the figure, the b segment is a vertical bar that appears in the upper right-hand side, overlapping the a bar in that corner. The c segment is a vertical bar that is somewhat longer than the b-segment bar, overlapping the right-hand end of the d-segment bar.

The numeric figures appearing on the screen actually have no visible breaks between adjacent segments as the displays for most

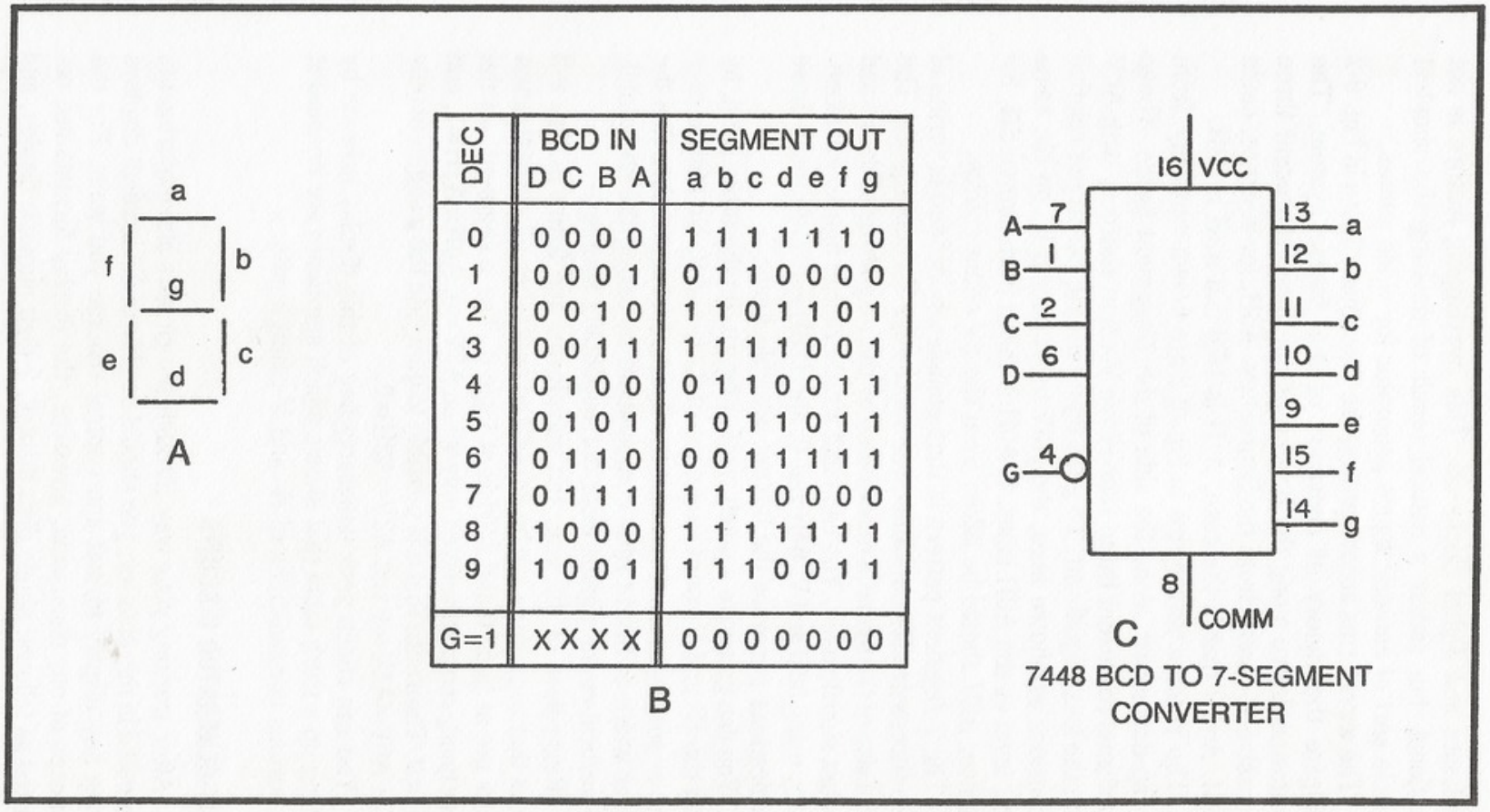


Fig. 9-1. A BCD to 7-segment converter. (a) The standard 7-segment digit format. (b) Conversion truth table. (c) The 7448 IC pin designations.

calculators and digital clocks do. This overlapping feature is not intentional, but rather a natural result of devising the simplest possible sort of numeric figure generator for video games.

The waveforms accompanying the 7-segment figure in Fig. 9-2 anticipate the theory of operation of the display system. The waveforms drawn along the bottom of the figure represent three successive H-counts from the Sourcebox. A0H, for instance, could be 8H. And if that is the case, A1H is 16H and A2H is 32H.

The V-count waveforms in Fig. 9-2 are drawn vertically, from the top downward, along the side of the 7-segment figure. These waveforms represent three successive V-count sources, with A0V being the least significant. To get a figure having the proper relative dimensions as shown here, the A0V input should be on the same count level as the A0H input. If A0H is taken from source 8H, for instance, A0V should be taken from the 8V count source.

The 7-segment pattern is thus completely defined in terms of three successive H-count and three successive V-count signals. The actual size of the figure is determined by the magnitude of the H- and V-count signals used. You can build a rather large figure, for example, by using 32H and V, 64H and V, and 128H and V. Actually those specifications generate the largest possible figure.

You can generate a much smaller figure, on the other hand, by using the H- and V-count sequence of 2, 4, and 8. No matter how large or small you want to make the figure, just remember that the proper width-to-height ratio comes about by taking the horizontal- and vertical-count signals from corresponding sources.

Regardless of the size, the waveforms and figure in Fig. 9-2 shows that segment a can be generated only when all vertical-count inputs are at logic 0 and A2H is at logic 0. The b segment, on the other hand, can be displayed only when A2V=0, A0H=A1H=1, and A2H=0. Then note that it is possible to generate the g segment only when A0V=A1V=1 and A2V=A2H=0.

You can check your understanding of this display scheme by making up a truth table that shows which segments are enabled at the various combinations of H- and V-count inputs.

DIGIT-GENERATOR CIRCUITS

After working your way through the circuits and experiments suggested in this chapter, you should be able to devise digit displays having any display format and control scheme you want. For the purposes of our discussion, however, the display formats are organized as follows: single digit display, 2-digit numeric display, and

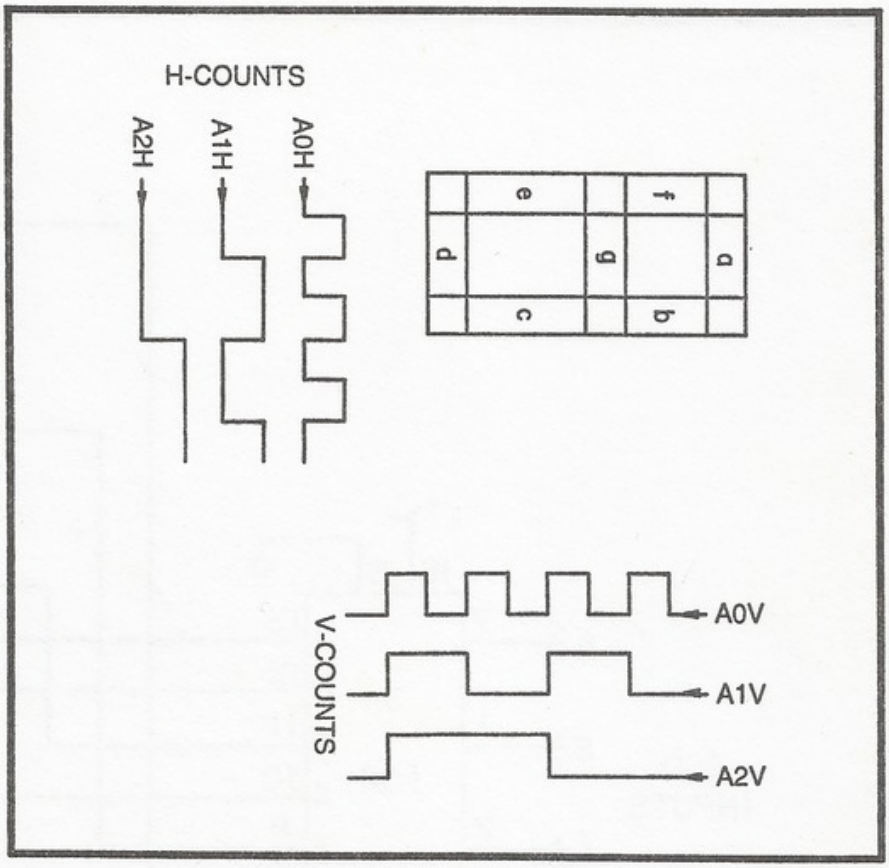


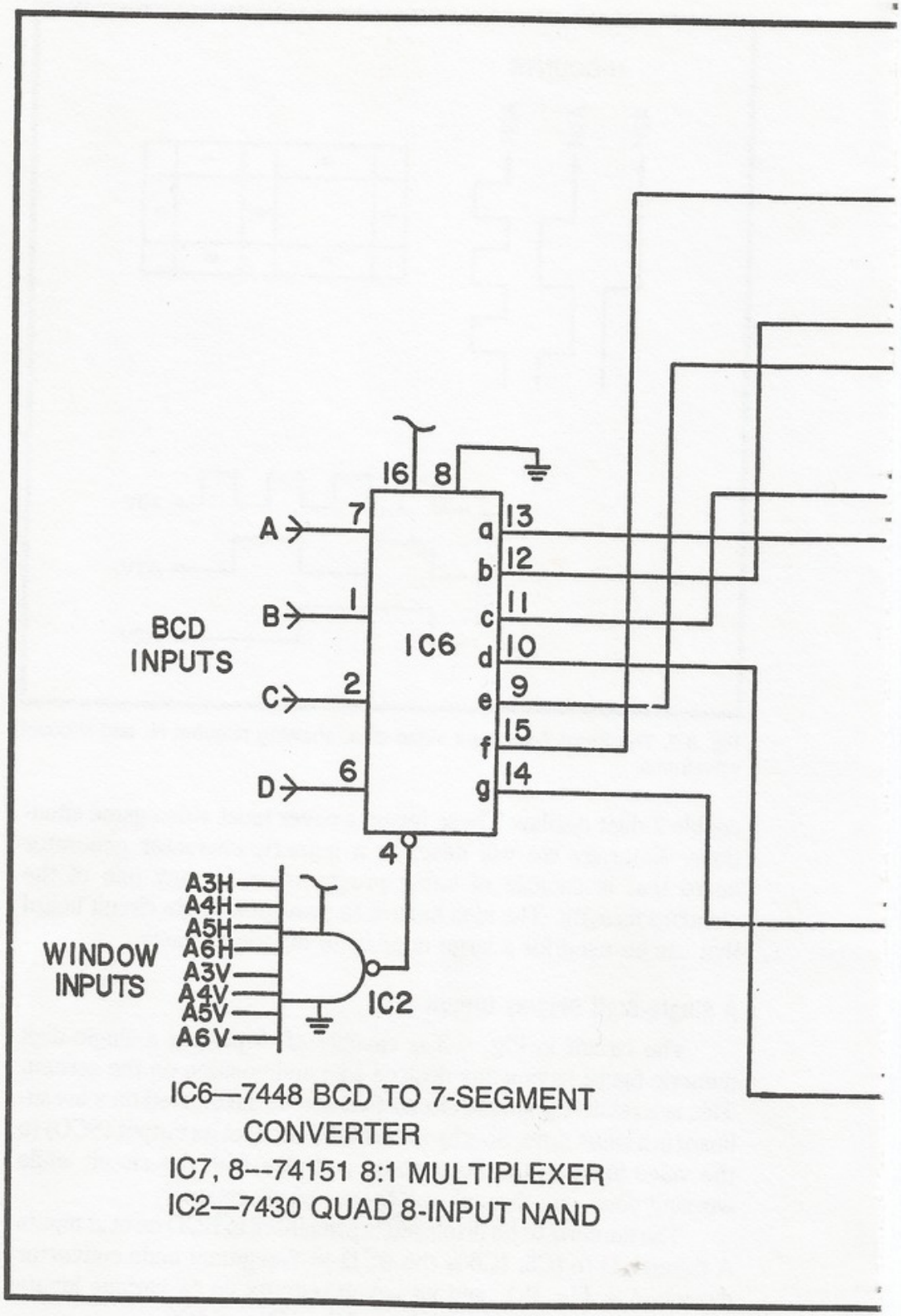
Fig. 9-2. The basic 7-segment video digit, showing relevant H- and V-count waveforms.

double 2-digit display. These formats cover most video game situations. Naturally we will describe a numeric character generator board that is capable of being programmed for any one of the standard formats. The idea here is to provide a single circuit board that can be used for a large number of different games.

A Single-Digit Display Circuit

The circuit in Fig. 9-3 is capable of displaying a single-digit numeric figure having any desired size and position on the screen. This is a relatively simple circuit that can be assembled on a breadboard in a short time. So why not build it, connect its output (SCO) to the video input of the Sourcebox, and play with the circuit while working your way through the theory of operation.

The numeral to be displayed is presented in BCD form at inputs A through D to IC6. IC6 is the BCD-to-7-segment code converter described in Fig. 9-1, and its seven outputs go to various inputs points on a pair of 8:1 multiplexer ICs, IC7 and IC8.



IC6—7448 BCD TO 7-SEGMENT CONVERTER

IC7, 8--74151 8:1 MULTIPLEXER

IC2—7430 QUAD 8-INPUT NAND

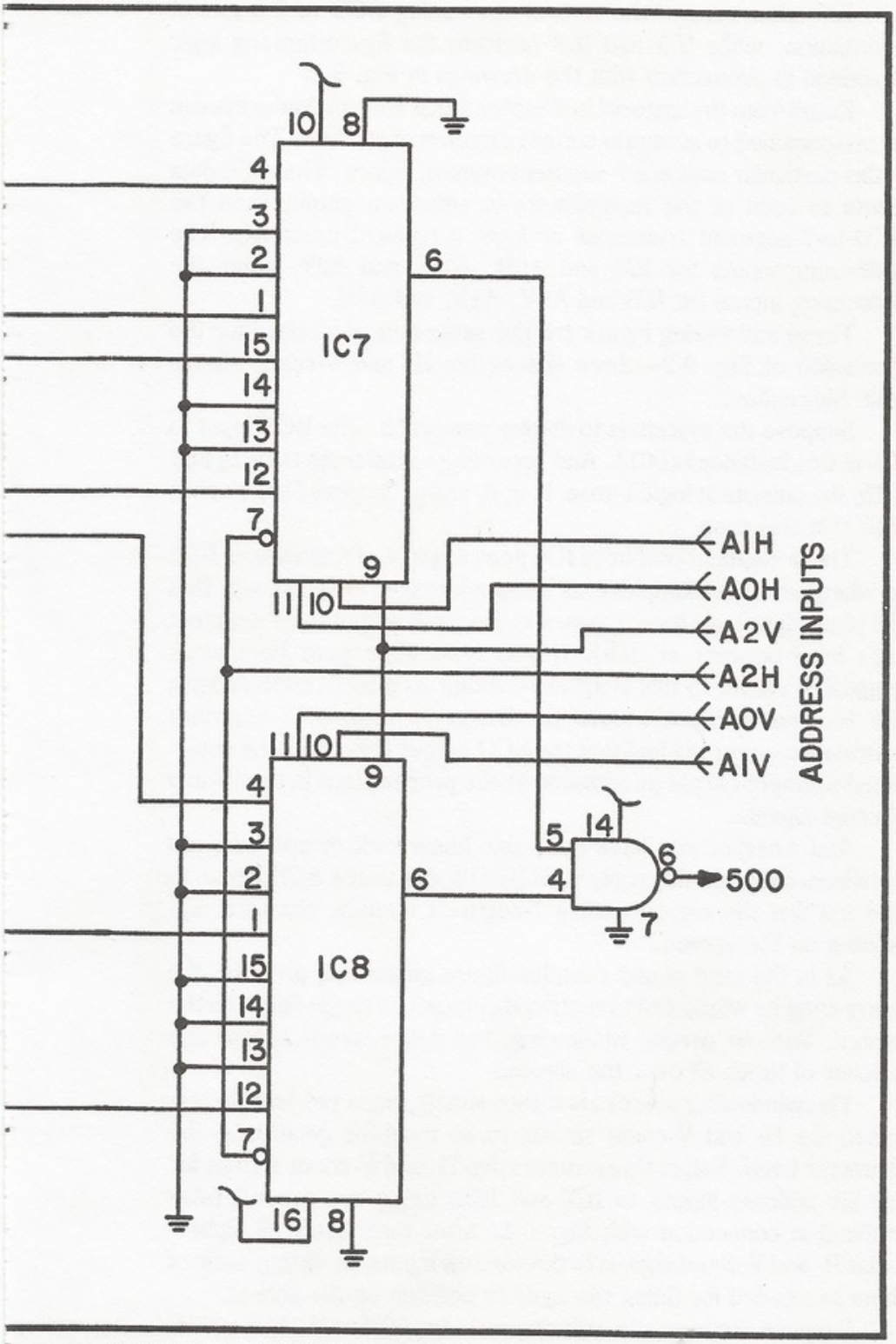


Fig. 9-3. A single-digit numeric character generator

IC6 takes care of the task of converting BCD to 7-segment information, while IC7 and IC8 perform the figure-forming logic described in connection with the drawings in Fig. 9-2.

Recall from the material in Chapter 4 that such multiplexers can be programmed to generate complex figures of all sorts. The figure in this particular case is a 7-segment numeric figure. The eight data inputs to each of the multiplexers is either an output from the BCD-to-7-segment converter or logic 0 (ground potential). The addressing inputs for IC7 and A0H, A1H, and A2V, while the addressing inputs for IC8 and A0V, A1V, and A2V.

These addressing inputs are the same ones described for the discussion of Fig. 9-2—three successive H- and V-count signals from Sourcebox.

Suppose the system is to display numeral 3. The BCD input to IC6 in this instance is 0011. And according to the truth table in Fig. 9-1b, the outputs at logic 1 are a, b, c, d, and g. Outputs f and e are at logic 0 in this case.

The a-segment output of IC6 goes to pin 4 of multiplexer IC8; so whenever that multiplexer is being addressed in such a way that the pin-4 input logic level appears at the pin-6 output, an a-segment logic level appears at SCO. If you have done your homework suggested earlier in this chapter—making up a truth table relating the H- and V-count inputs to the generation of 7-segment information—you will find that the SCO output generates the designated segment output information at the proper place in the H- and V-count signals.

And whether you have done this homework or not, you can breadboard this circuit, apply valid BCD codes to the BCD inputs to IC6 and find the corresponding 7-segment numeric character appearing on the screen.

As in the case of any complex-figure generating process, the figure must be windowed to restrict its image to a single figure on the screen. Without proper windowing, the figure would appear any number of times all over the screen.

The windowing process is rather simple, once you have determined the H- and V-count signals to be used for generating the character itself. Select three successive H- and V-count signals for the six address inputs to IC7 and IC8, using the general rules outlined in connection with Fig. 9-2. After that, apply all higher-order H- and V-count signals to the window inputs, inverting some of them as needed for fixing the figure's position on the screen.

Suppose, for example, you choose to let A0H=8H, A1H=16H, and A2H=32H. This decision makes it necessary to set A0V=8V,

A1V=16V, and A2V=32V. That takes care of the addressing inputs for the multiplexers. Now the remaining higher order Sourcebox signals—64H, 128H, 256H, 64V, 128V, and 265V—must go to the window inputs of IC2. You will want to invert some of these signals before applying them to IC2 thereby placing the figure at some convenient position on the screen. Review the material concerning figure windowing in Chapter 4 if you find you are getting lost at this point.

After fixing the address and windowing inputs, you can generate some figures by applying logic 1 and 0 levels to the BCD inputs. Connect all four BCD inputs to COMM, for instance, and you should find a 0 on the screen. The size of the figure is determined by your selection of address inputs, and its position is fixed by your selection of inverted or noninverted inputs at the windowing NAND gate, IC2.

Disconnect the grounding wire from input A, leaving the other three grounded, and you should see a figure 1 on the screen. If you connect nothing at all to the four BCD inputs, they automatically assume logic-1 states, resulting in no figure on the screen. You have most likely discovered this little fact for yourself while setting up the circuit initially. Just remember that this system responds only to valid BCD inputs, binary versions of numbers 0 through 9. Any input number larger than 9 results in either a meaningless figure or none at all.

If you want to have some more fun with the circuit in Fig. 9-3, remove all your programming jumper wires from inputs A through D and replace them with a BCD counter as shown in Fig. 9-4a.

This counter circuit can be clocked, using either the circuit in Fig. 9-4a or 9-4b. The circuit in Fig. 9-4b is a 555 timer wired as a monostable multivibrator. Each time the experimenter depresses the COUNT pushbutton, the timer generates a 10-ms pulse that increments the counter, ultimately advancing the count appearing on the screen. The count can be cleared to 0 at any time by simply depressing the CLEAR pushbutton. A numeric display system used in this fashion is typical of game-scoring operations. Every time a score pulse (COUNT operation) takes place, the numeral on the screen advances one unit.

The 555 timer in Fig. 9-4c is connected as a free-running multivibrator that has a frequency between one pulse every 6 or 8 seconds and 10 Hz, depending on the value of R2. The larger the value of R2, the slower the oscillator runs.

The circuit in Fig. 9-4c can be used as a CLK source for the counter, making the display increment automatically at the CLK frequency. The clock can be stopped and started by means of the

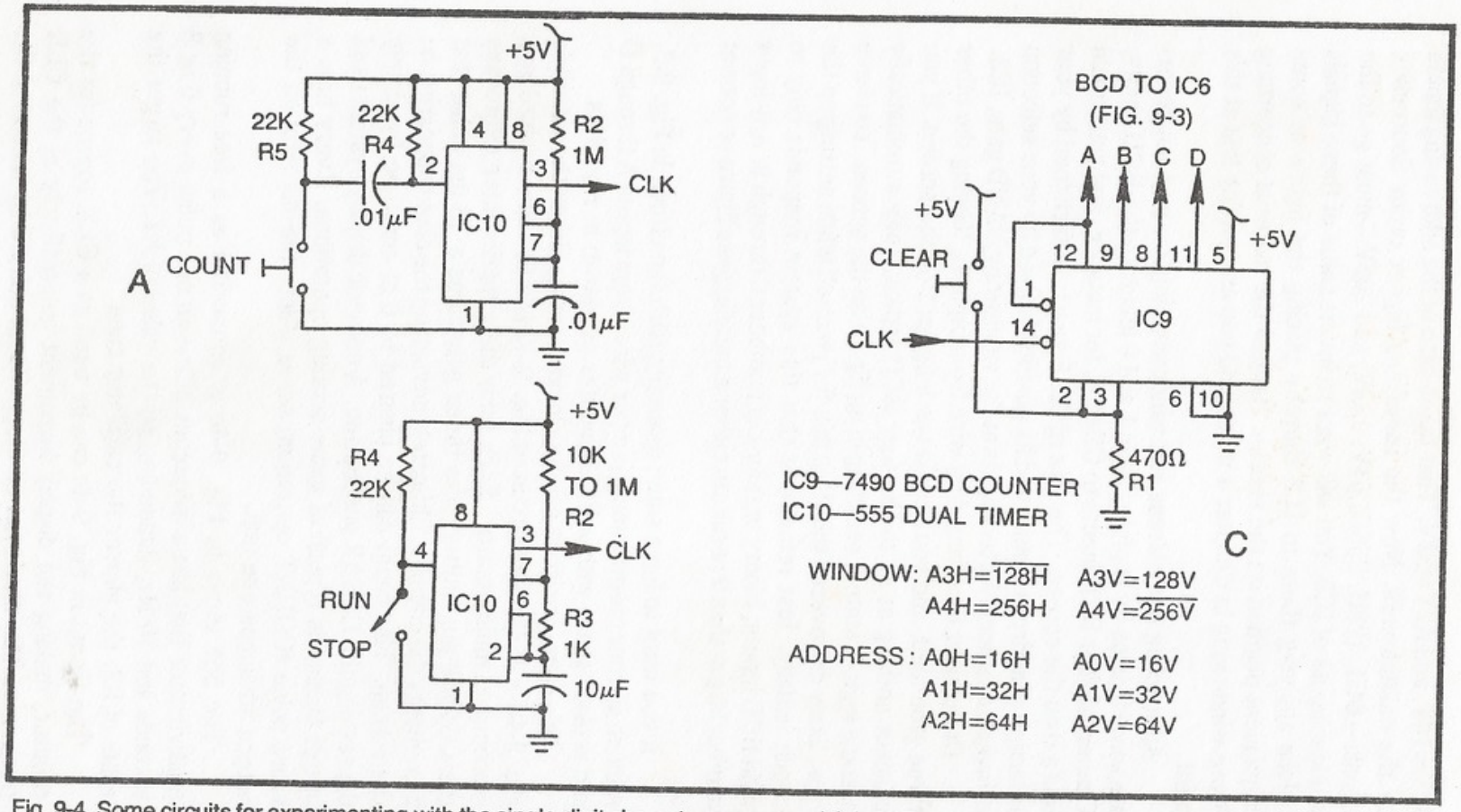


Fig. 9-4. Some circuits for experimenting with the single-digit character generator. (a) A BCD counter input. (b) A monostable clocking source. (c) A free-running clocking source.

STOP/RUN switch, or the display can be cleared to zero by depressing the CLR pushbutton on the counter.

Free-running display counters of this sort are used for displaying game times.

A Two-Digit Display Circuit

The single-digit display works fine as long as the score or game time never exceeds a figure 9. It is capable of registering single digits between 0 and 9, but many games call for scoring and timing larger numbers.

Figure 9-5 shows the general display scheme for a 2-digit video display. The two digits have identical segment designations, but are separated by a space that is one digit wide. For most counting and timing applications, the digit on the left is a 10s digit, while the other is a units digit.

The scheme for generating the figures is identical to that used for a single-digit dispaly, except for an A3H signal that is used for

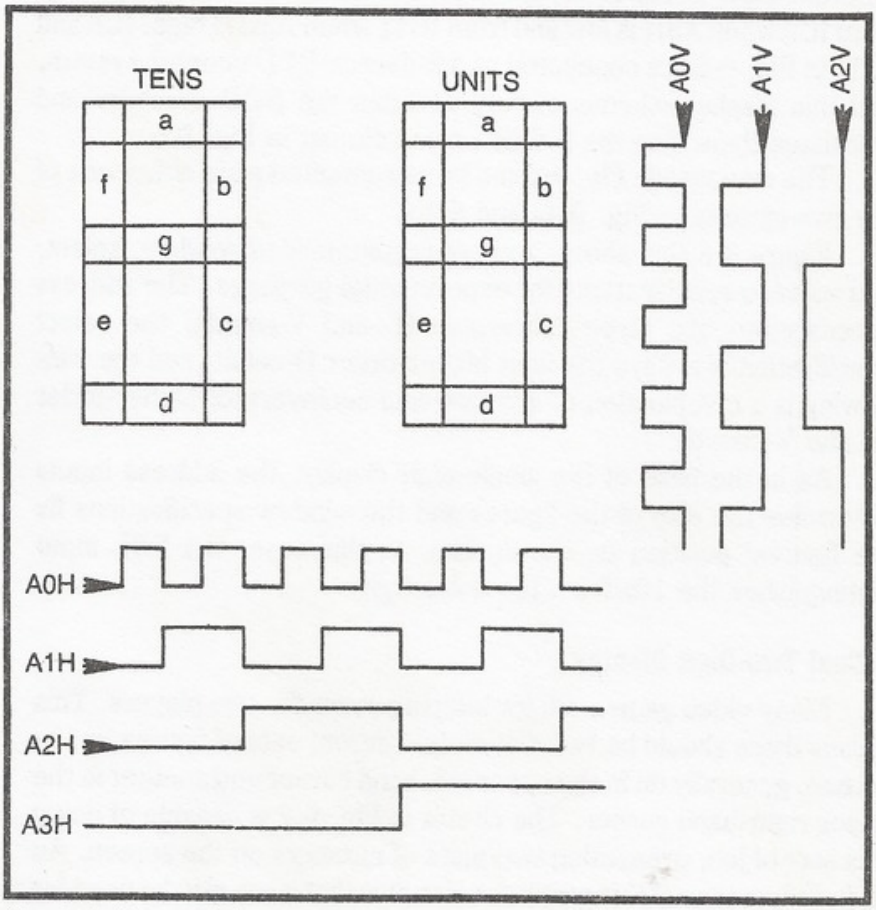


Fig. 9-5. A two-digit numeric display, showing relevant H- and V-count waveforms.

distinguishing one digit from the other. Whenever A3H is at logic 0, the system generates the 10s digit, but when this H-count input rises to logic 1, the system uses the same circuitry to generate the units digit.

The circuitry for decoding BCD numbers and generating the 7-segment figures on the screen is identical to the single-digit output circuit in Fig. 9-3. The windowing circuit is also the same.

Figure 9-6, however, shows how the basic output circuit can be modified to generate the 2-digit format. IC12 in Fig. 9-6 is a quad 2:1 multiplexer circuit that must be used for selecting the BCD words for the two digits. Whenever the SEL (SELECT) input is at logic 0, this multiplexer feeds IC6 in Fig. 9-3 the BCD word from counter IC9, the units counter. As the A3H signal to SEL rises to logic 1, however, IC12 provides the BCD word from the 10s digit counter, IC11.

The output circuit in Fig. 9-3 is thus sampling one of two different BCD words at any given time. It samples and displays data from IC9 when A3H is low and from IC11 when A3H is high. IC9 and IC11 in Fig. 9-6 are connected as a 2-decade BCD counter system, and this display scheme merely decodes the BCD numbers and translates them into the 2-digit screen format in Fig. 9-5.

The counters in Fig. 9-6 can be incremented from either one of the two circuits in Fig. 9-4a and 9-4b.

Figure 9-6 also shows a set of recommended window, select, and address specifications for experimental purposes. The address specifications are three successive H- and V-counts, the select specification is always the next higher-order H-count, and the windowing is a combination of inverted and noninverted, higher-order H- and V-counts.

As in the case of the single-digit display, the address inputs determine the size of the figures and the window specifications fix the figures' position on the screen. In this case, the SEL input distinguishes the 10s from the units digit.

A Dual Two-Digit Display

Many video games call for keeping score for two players. This means there should be two distinctly different sets of figures on the screen, generally on in the upper left-hand corner and another in the upper right-hand corner. The circuit in Fig. 9-7 is capable of doing this sort of job, presenting two pairs of numbers on the screen. An added advantage of this particular circuit is that it can also be used for generating simpler numeric displays. This one circuit board, in other

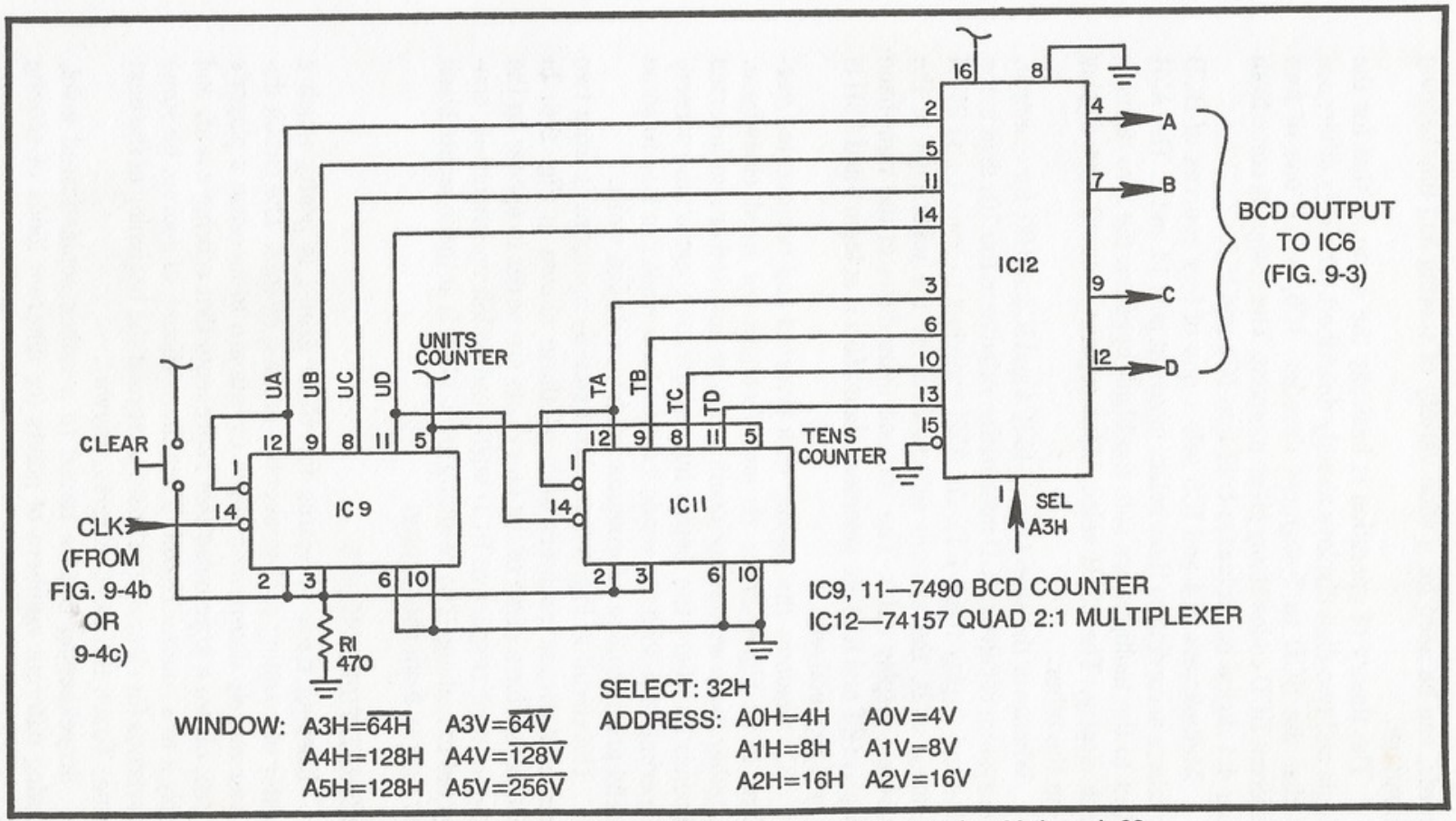


Fig. 9-6. A 2-digit counter input for the two-digit numeric display. This system generates number 00 through 99.

words, can be used for a wide variety of scoring and timekeeping situations.

The theory of operation is basically the same as that for the single and two-digit displays already described. The only difference is that the BCD to 7-segment decoder, IC6, is fed one of four different BCD codes at any given moment. These signals come from four 4:1 digital multiplexers built into IC4 and IC5.

Multiplexers IC4 and IC5 select one of four sources of BCD numbers according to their select inputs at pins 14 and 2. The A3H input to the multiplexers sets the blank between the two digits in each display. The 256H select input separates one 2-digit display from the other.

Whenever the select inputs to IC4 and IC5 and 00, for example, the system receives BCD information relative to the 10s digit in the left-hand display (digit LT). As A3H switches to logic 1 and 256H remains at 0, the system selects data for the units digit in the left-hand display (LU). The 10s and units digits in the right-hand display (RT and RU) are selected when 256H is at logic 1 and A3H is 0 or 1 respectively.

IC2 windows the display with inverted and noninverted versions of all higher-order H- and V-counts not used elsewhere. Whether these windowing inputs are to be inverted or non-inverted depends on where the display figures are to appear on the screen. Experimenting with the window inputs for a while can lead to some useful programming information for future design work.

The circuit in Fig. 9-7 can be tested by operating it from two sets of 2-decade counters such as those shown in Fig. 9-6. In instances where some of the four digits are never to appear on the screen, their respective BCD inputs can be left uncommitted, connected to nothing. The resulting logic-1 levels at these inputs blank the figures from the display.

SCOREKEEPING CIRCUITS

Keeping track of scores for video games is pretty much a matter of counting certain events—scoring events. The circuit designer merely determines which events are to increase a player's score, obtain a logic pulse that represents that scoring event, and apply it to a counter circuit. Then there should, of course, be some provisions for clearing the score to zero at the beginning of the next game. That's the basic process, anyway.

Scorekeeping can be raised to a rather sophisticated level, scoring different numbers of points for different kinds of scoring

events, scoring more than one player, and making the scoring process part of the game control.

The Simplest Scoring Scheme

The simplest scoring scheme is one that merely advances a player's score one count whenever a particular scoring event occurs. Consider the Golf game in Chapter 8, for instance. A simple counter circuit can be triggered by the SWING pulse, thus keeping track of the number of swings in a game sequence.

Figure 9-8a shows a pair of binary counters cascaded to provide a BCD count from 00 through 99. The scoring event, whatever it might be, is presented as a brief positive-going pulse. The count increments on the negative-going edge of that particular pulse. The circuit is likewise cleared to zero by a positive pulse on the CLEAR bus. These counters, incidentally, can be cleared and held at zero, in spite of scoring events, by holding the CLEAR bus at logic 1.

The U outputs in Fig. 9-8a are the units BCD code that is to operate either the left or right units input to the score pattern generator in Fig. 9-7. The T outputs from IC2 in Fig. 9-8a are the 10s units.

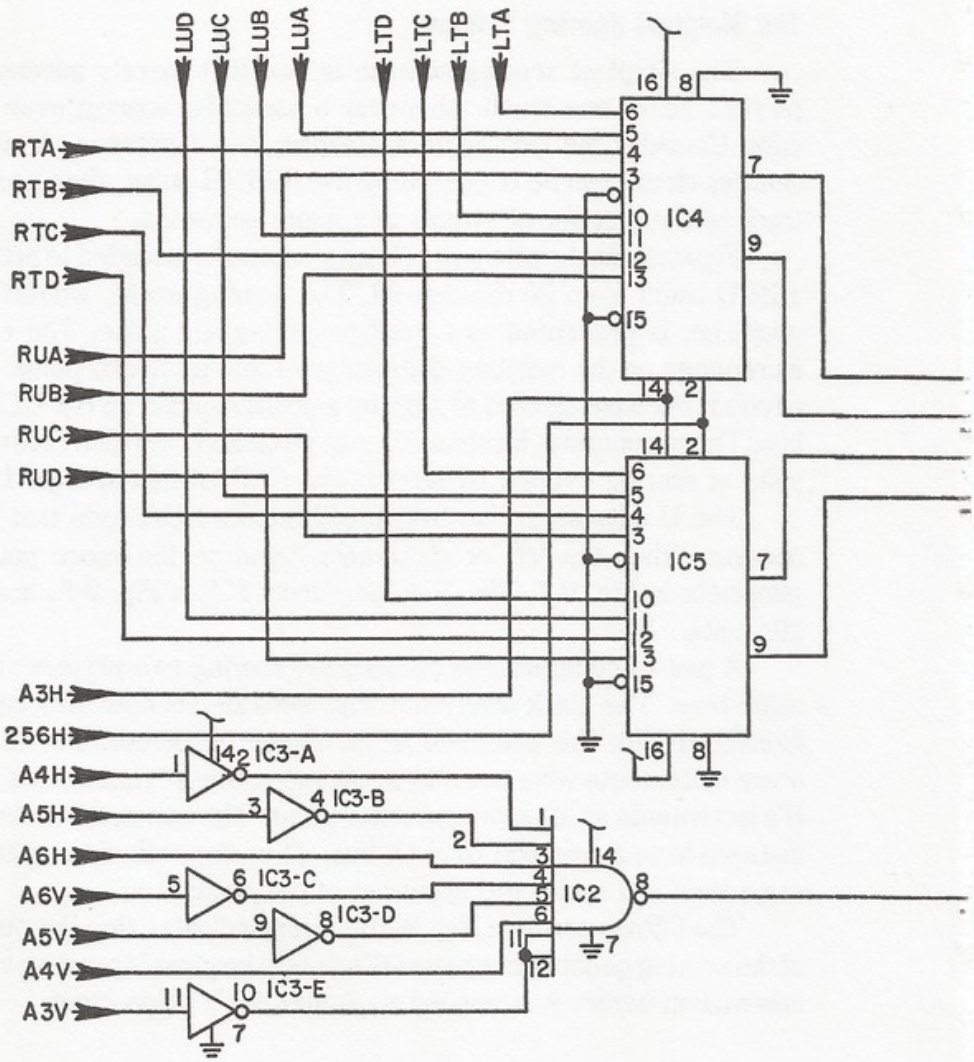
A pair of counters can be used for scoring two players at the same time. The block diagram in Fig. 9-8b shows how they can be interfaced with the complete score figure generator. Player A's score increments whenever his particular scoring event occurs, and B's increments as his scoring events occur. The two counters in this instance have a common CLEAR bus. They are both cleared at the same time and under the same set of circumstances.

The ORing circuit in Fig. 9-8b merely indicates that the output of the scoring generator is to be ORed with the game's primary video information before it is applied to Sourcebox's video input.

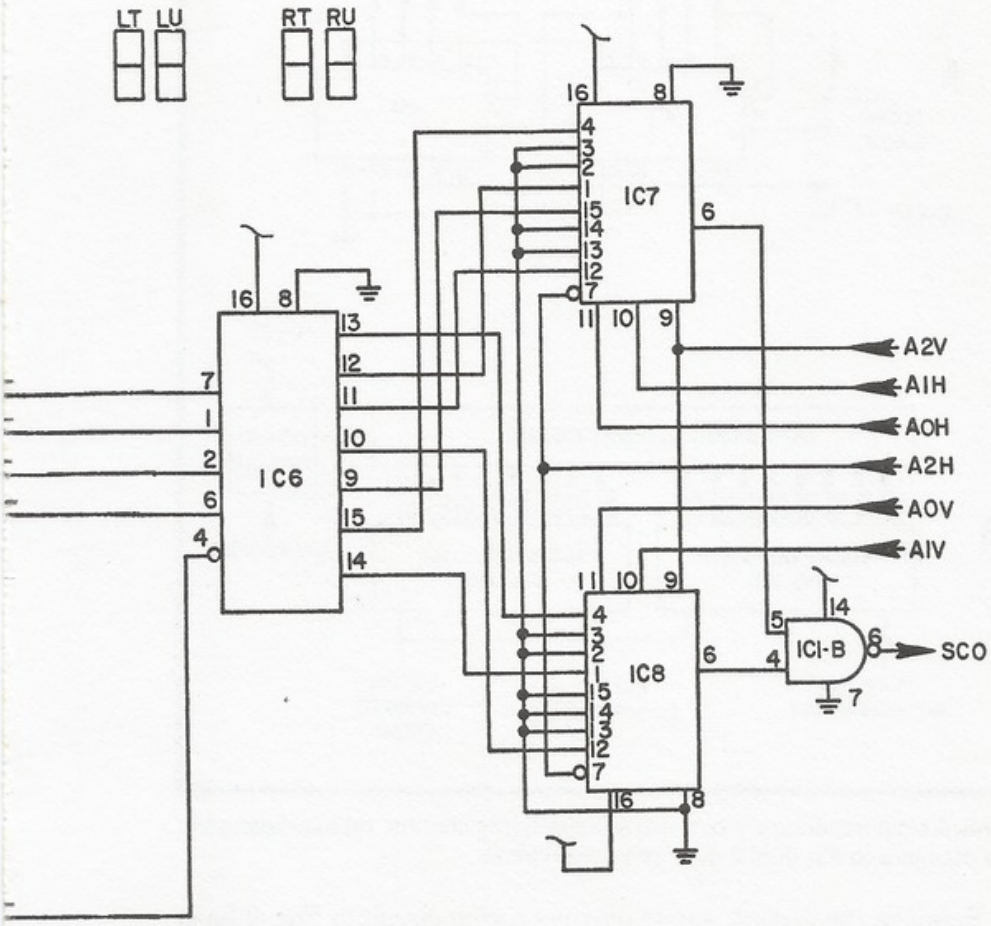
Scoring and Game Control

Many kinds of TV games call for ending a game sequence or changing the game pattern whenever the score reaches a certain point. The scoring is an integral part of the game control.

Figure 9-9a shows a typical control circuit for a TV game. It is mainly an $\overline{R}$-$\overline{S}$ flip-flop that is manually set by depressing a START pushbutton, and it is reset by an active-low $\overline{STOP}$ pulse. Depressing the START button begins the game, and the occurrence of the $\overline{STOP}$ pulse ends it. Scoring becomes an integral part of the game control when $\overline{STOP}$ is generated by a scoring circuit.



IC1—7400 QUAD 2-INPUT NAND
IC2—7430 8-INPUT NAND
IC3—7404 HEX INVERTER
IC4, 5—74153 DUAL 4:1 MULTIPLEXER
IC6—7448 BCD TO 7-SEGMENT
CONVERTER
IC7, 8—74151 8:1 MULTIPLEXER



RECOMMENED SPECIFICATIONS

WINDOW: A4H=$\overline{32H}$ A3V=$\overline{16V}$
A5H=$\overline{64H}$ A4V=32V
A6H=128H A5V=$\overline{64V}$
A6V=$\overline{128V}$

SELECT: A3H=16H AND 256H

ADDRESS: A0H=2H A0V=2V
A1H=4H A1V=4V
A2H=8H A2V=8V

Fig. 9-7 A complete dual 2-digit display generator

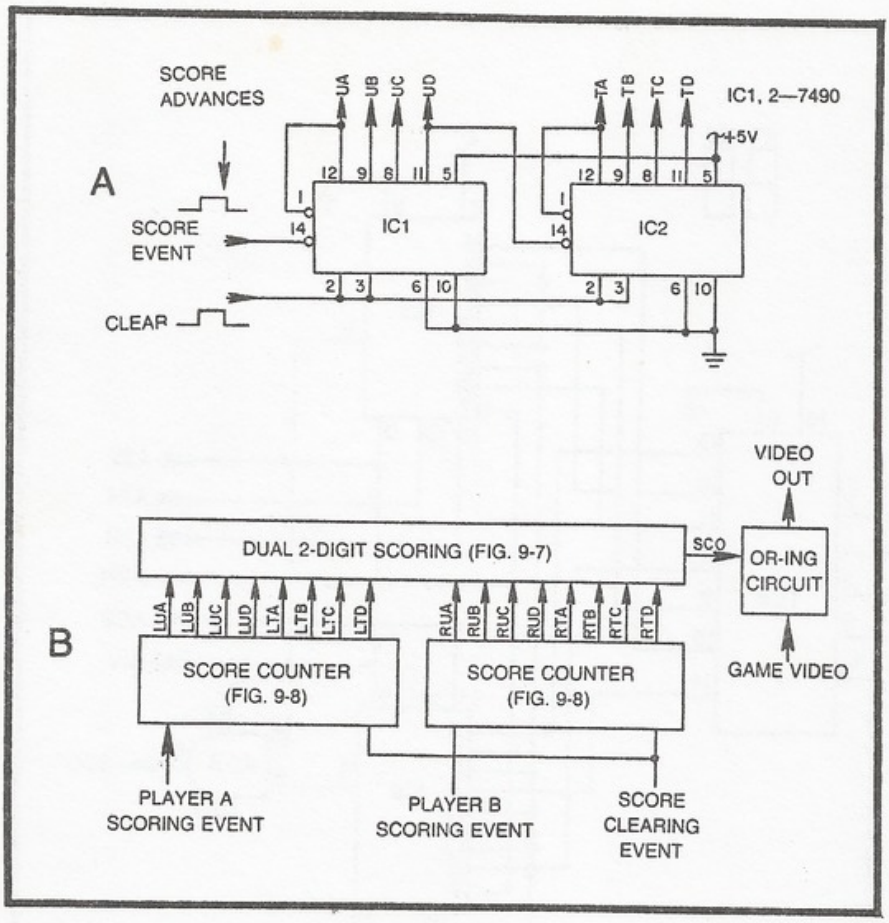


Fig. 9-8. A simple scoring scheme. (a) A 2-digit score counter. (b) Interfacing two score counters to the dual 2-digit generator circuit.

Suppose the 2-digit, single player scoring circuit in Fig. 9-8a is to control this game. Depressing the START button causes the output of IC3-B in Fig. 9-9a to drop from logic 1 to 0, and the pulse generator produces a brief pulse that is inverted to a positive-going pulse from IC4-B. This pulse is applied to the counter's CLEAR bus to clear the two digits to zero.

As the game progresses, the score presumably increments, advancing the count from IC1 and IC2 in Fig. 9-8a toward some game-ending score. Now suppose that game-ending score is 25. So when the 10s counter reaches the binary version of 2 (0010) and the units counter reaches 5 (0101), the game should stop. In other words, this game should stop the first time TB=1, UA and UC=1.

The 3-input NAND gate in Fig. 9-9c is wired to sense this particular count. The output of that gate is normally at logic 1, but drops to 0 when the counters show the BCD version of 25. At that moment, the little pulse generator in Fig. 9-9c produces a negative-

going pulse that is fed to the $\overline{\text{STOP}}$ input of the game control circuit in Fig. 9-9a, thus stopping the game action and holding the 25 scoring figure on the screen.

The score is cleared and the game is restarted by depressing the START button once again.

The basic idea behind the score-stopping control is to sense the score at which the game is to stop, connecting the counter outputs that are at logic 1 to the input of a NAND gate. The chart in Fig. 9-9b

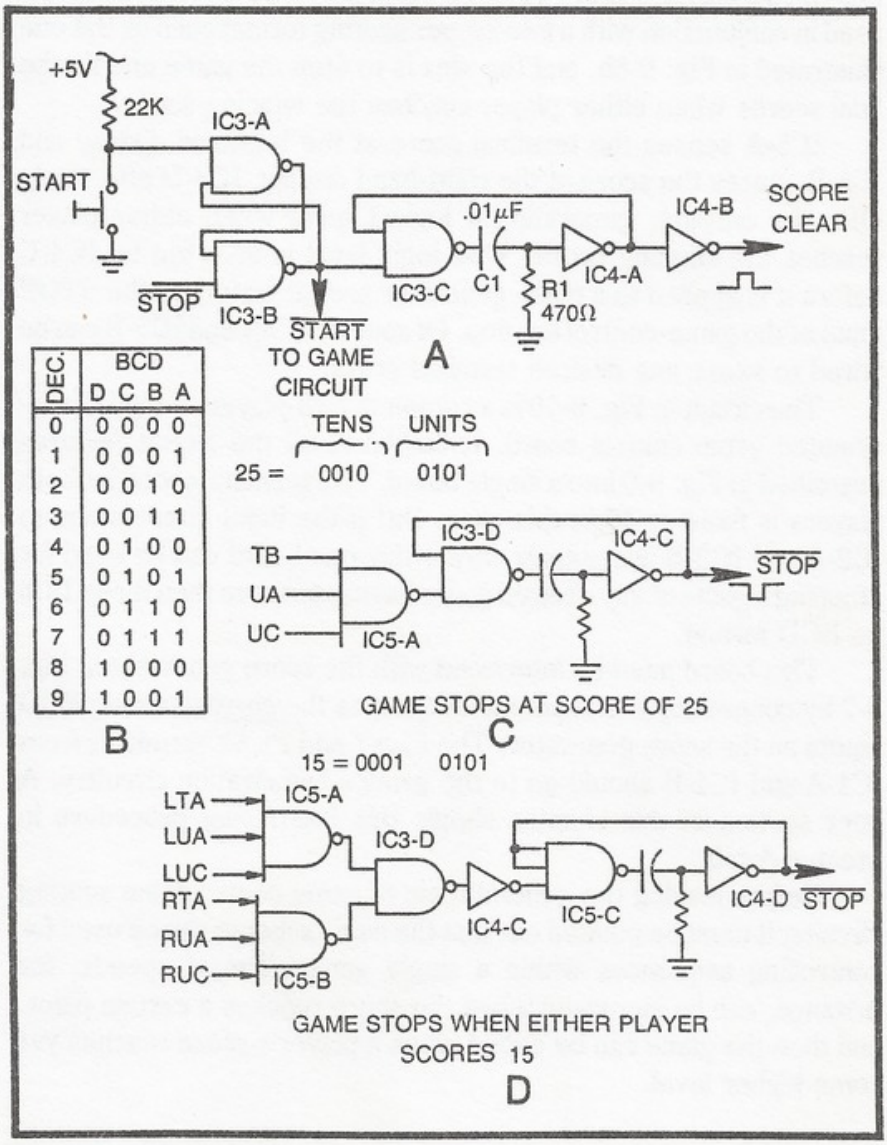


Fig. 9-9. Circuits for controlling the game from the scoring. (a) Automatic score clearing at the start of a game. (b) Score counter truth table. (c) Stopping the game when a single-player score reaches 25. (d) Stopping the game when either of two players' score reaches 15.

summarizes the output status of a counter at its 10 different counts. The NAND gate that senses the stop count must be large enough to handle all the 1s from both the units and 10s counter. A 3-input NAND gate is normally adequate, but a 4-input gate would be required in the rather odd case where one might want to stop the game at score of 47 (0100 from the tens counter and 0111 from the units counter).

The circuit in Fig. 9-9d shows how to stop a game whenever one of two player's score reaches 15. This score-sensing circuit is used in conjunction with a two-player scoring format such as the one illustrated in Fig. 9-8b, and the idea is to stop the game and fix the final scores when either player reaches the winning score.

IC5-A senses the terminal score at the left-hand display and IC5-B senses the score at the right-hand display. IC3-D effectively ORs the outputs, generating a logic-1 level when either player reaches the winning score. That logic level is inverted by IC4-C before it is applied to a pulse generator and ultimately to the STOP input of the game-control flip-flop. Of course IC5-A and IC5-B can be wired to sense any desired terminal score.

The circuit in Fig. 9-10 is a complete two-player, 2-digit score-oriented game control board. It combines all the circuit features described in Fig. 9-9 into a single board. The terminal score for both players is fixed at 45 in this case. But if the input connections to IC2-A and IC2-B are jumper wires, this one board can be used for stopping a game at any desired score having no more than three 1s in its BCD format.

This board must be interfaced with the score generator in Fig. 9-7 by connecting the counters' outputs to the corresponding BCD inputs on the score generator. The PLAY and $\overline{\text{PLAY}}$ terminals from IC1-A and IC1-B should go to the game's initialization circuitry. A later section in this chapter shows this interfacing procedure in greater detail.

Before leaving the general topic of game control from scoring circuits, it must be pointed out that the same scheme can be used for controlling sequences within a single game. Target speeds, for instance, can be increased when the score reaches a certain point, and then the game can be ended when a player's score reaches yet some higher level.

Weighted Scoring

Many games in the real world call for weighted scoring, making some game scores count more than others. A touchdown in football,

for instance, is worth 6 points, a field goal is worth 3, and a point after touchdown is good for 1 point. We can do the same thing with video games.

The circuit in Fig. 9-11 shows a rather simple weighted scoring circuit. It has nine scoring inputs labeled S1 through S9, and a CLK output. The CLK output clocks a scorekeeping counter such as any of those described in the previous two sections of this chapter. The S inputs accept active-low scoring-event pulses from the game system.

Suppose a certain game event calls for scoring 2 points for one of the players. The game system generates a negative-going pulse that is applied to the S2 input, and the ultimate result is that two clock pulses emerge from the CLK output. An event in the same game might then call for a 4-point score, and in this case the event-scoring pulse is applied to S4, causing the CLK to generate four output pulses.

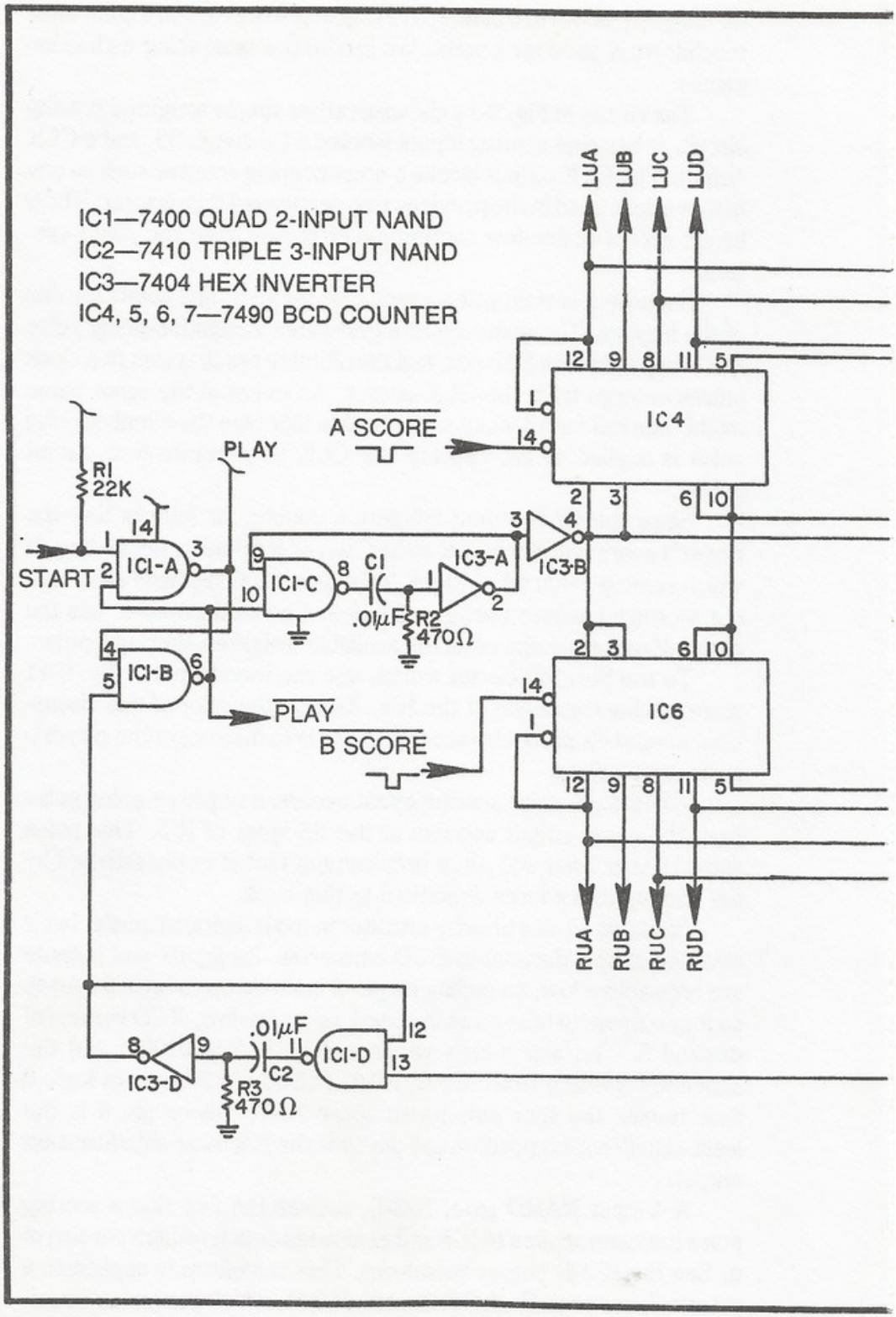
Since the CLK output triggers a counter, it follows that the player's score will increment either two or four units, depending on which scoring event takes place. This circuit actually handles weighted scores anywhere between 1 and 9. The experimenter has the option of using any one or all the available weighted-scoring inputs.

To see how this circuit works, use the waveforms in Fig. 9-11 while tracing the action of the ICs. And for the sake of this discussion, suppose a particular scoring event is to increment the player's score by 5 points.

When the 5-point scoring event occurs, a negative-going pulse from the game circuit appears at the S5 input of IC5. This pulse should last at least 100 μs, a requirement that is easily satisfied by the scoring procedures described in this book.

IC5 is listed as a priority encoder in most digital manuals, but it also works as a decimal-to-BCD converter. Its inputs and outputs are both active low, so pulling its pin-2 input down to logic 0 causes its four outputs to take on an inverted, or active-low, BCD version of decimal 5. The active-high version of BCD 5 is 01001, and the active-low version from IC5 is 1010. Pulling pin 2 down to logic 0 thus causes the four outputs to show 1010, where pin 9 is the least-significant-bit position and pin 14 is the ICs most-significant-bit output.

A 4-input NAND gate, IC2-B, senses the fact that a scoring pulse has been applied to IC5 and generates an active-high version of it. See the IC2-B output waveform. This waveform is applied to a resettable monostable multivibrator, IC3-A, which generates a positive pulse having a duration of about 25 μs.


IC1—7400 QUAD 2-INPUT NAND
IC2—7410 TRIPLE 3-INPUT NAND
IC3—7404 HEX INVERTER
IC4, 5, 6, 7—7490 BCD COUNTER
LUA
LUB
LUC
LUD
IC4
A SCORE
PLAY
R1
22K
START
IC1-A
IC1-C
C1
.01μF
R2
470Ω
IC3-A
IC3-B
IC1-B
PLAY
B SCORE
IC6
RUA
RUB
RUC
RUD
.01μF
IC3-D
C2
R3
470Ω
IC1-D

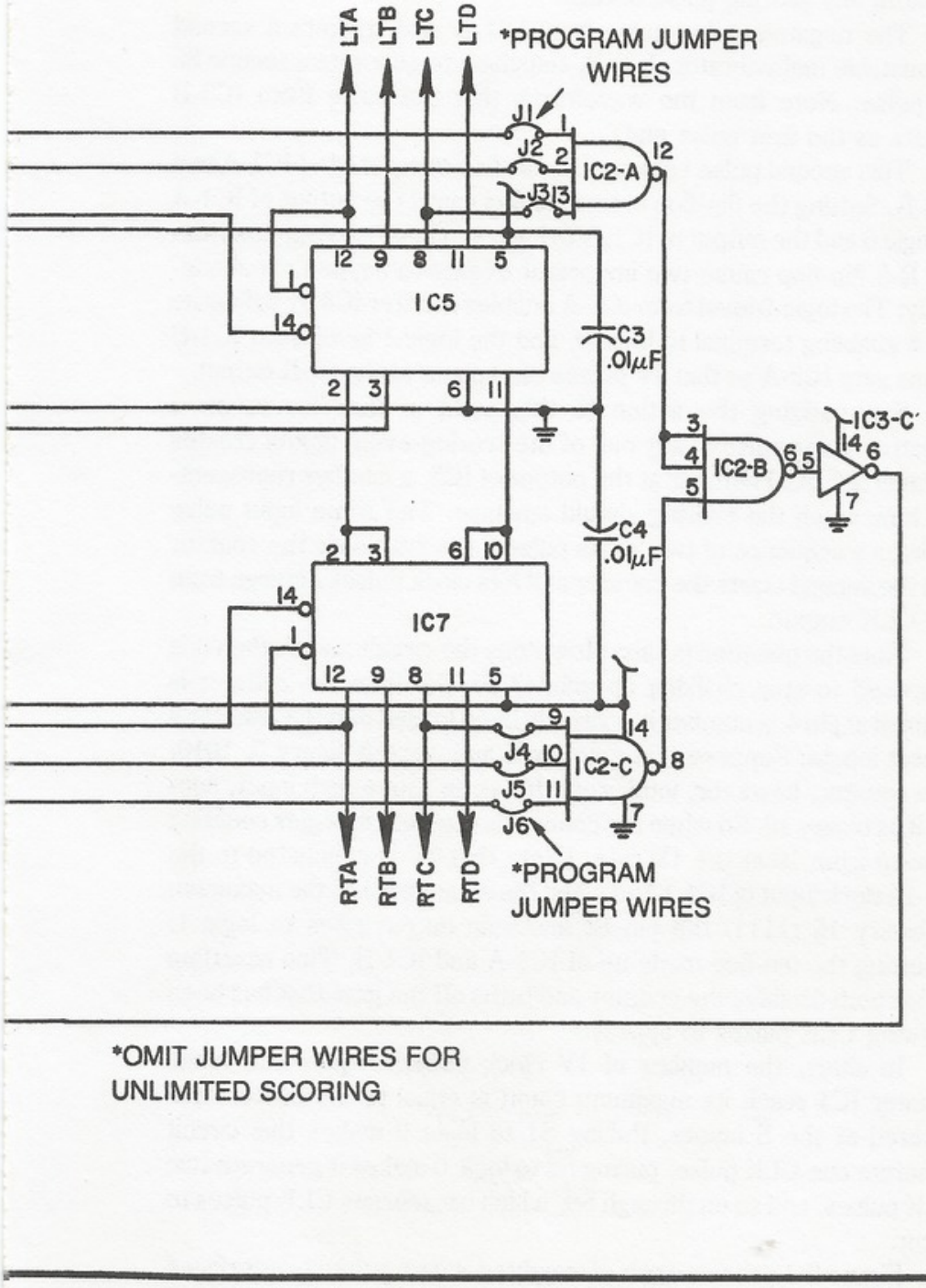


Fig. 9-10. A complete two-player game control circuit

The pulse from IC3-A always begins at the start of the scoring waveform. And after being inverted by IC1-D, it is applied to the pin-11 loading input of IC4, a presettable binary counter. This counter, in other words, is loaded with the binary output of IC5 the moment any scoring pulse occurs.

The negative-going pulse from IC1-D also triggers a second monostable multivibrator, IC3-B, causing it to generate a second 25 μs pulse. Note from the waveforms that the pulse from IC3-B occurs as the first pulse ends.

This second pulse sets an R-S flip-flop composed of IC1-A and IC1-B. Setting the flip-flop in this fashion snaps the output of IC1-A to logic 0 and the output of IC1-B to logic 1. These two outputs from the R-S flip-flop cause two important events to happen simultaneously: The logic-0 level from IC1-A enables counter IC4 by pulling its pin-4 enabling terminal to logic 0, and the logic-1 level from IC1-B opens gate IC2-A so that 1V pulses can appear at the CLK output.

Summarizing the action to this point in the discussion, a negative-going pulse at any one of the scoring-event inputs creates an inverted BCD number at the output of IC5, a number representing how much the scoring should advance. The same input pulse initiates a sequence of two 25 μs pulses, the first loads the counter and the second starts the counter and lets clock pulses emerge from the CLK output.

Now the question is this. How does the circuit know when it is supposed to stop chalking up points? By the time the counter is enabled at pin 4, a number has already been loaded into the counter's preset inputs. Suppose that number is an inverted binary 5, 1010. The counter, however, interprets this as an active-high input, seeing it as binary 10. So when the counter is enabled, it begins counting upward from 10 at the 1V rate. (Note that 1V is connected to the pin-14 clock input of IC4.) And when the count reaches the maximum of binary 15 (1111) the pin-12 max/min output rises to logic 1, resetting the flip-flop made up of IC1-A and IC1-B. This resetting action both disables the counter and turns off the gate that has been allowing CLK pulses to appear.

In short, the number of 1V clock pulses required to make counter IC4 reach its maximum count is equal to the score value entered at the $\overline{S}$ inputs. Pulling $\overline{S1}$ to logic 0 makes this circuit generate one CLK pulse, pulling $\overline{S2}$ to logic 0 makes it generate two CLK pulses, and so on through $\overline{S9}$, which causes nine CLK pulses to occur.

Figure 9-12 shows a pair of weighted-scoring circuits interfaced with the two-player score counters and figure generators described

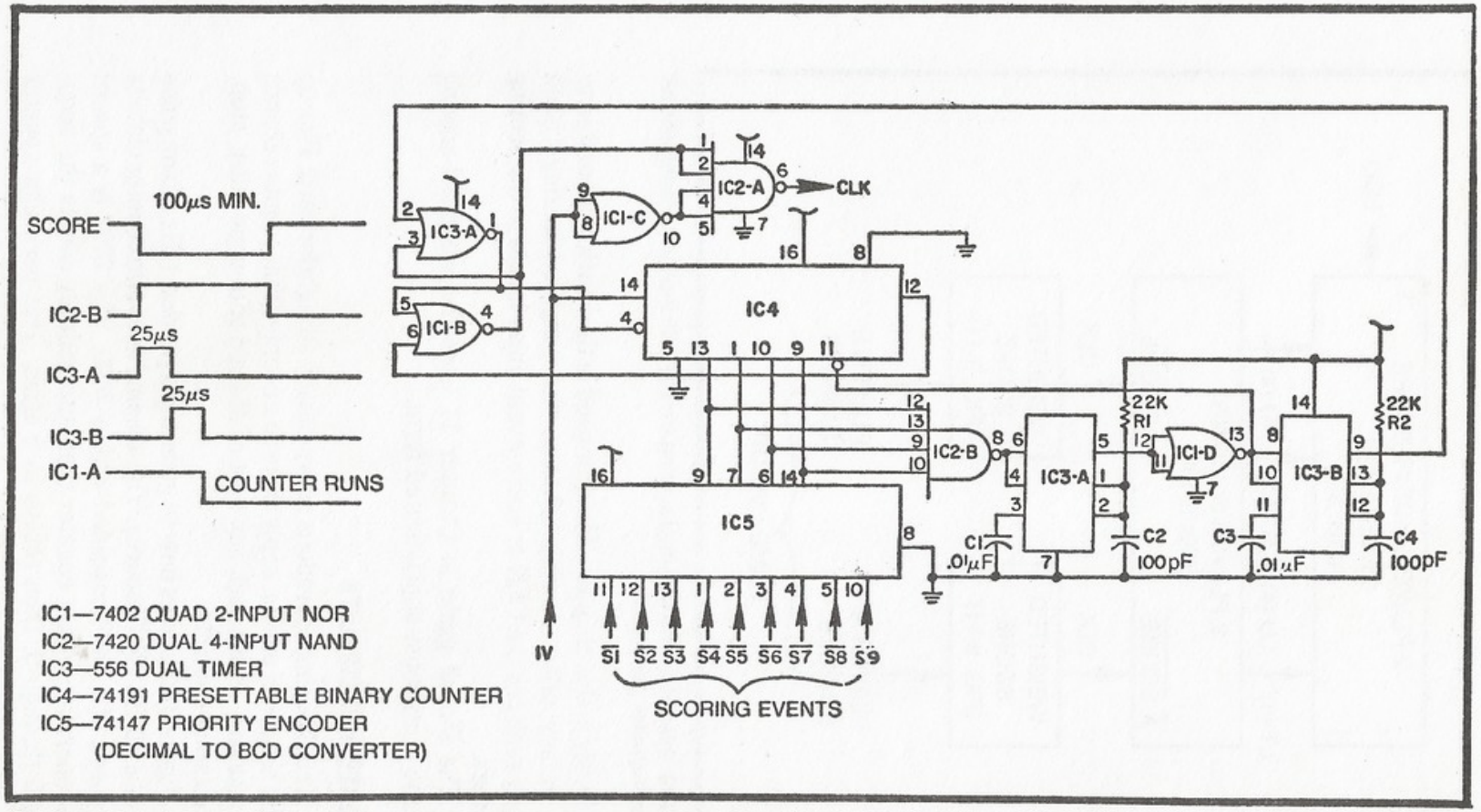


Fig. 9-11. A weighted scoring circuit for giving different game events different scores between 1 and 9 points.

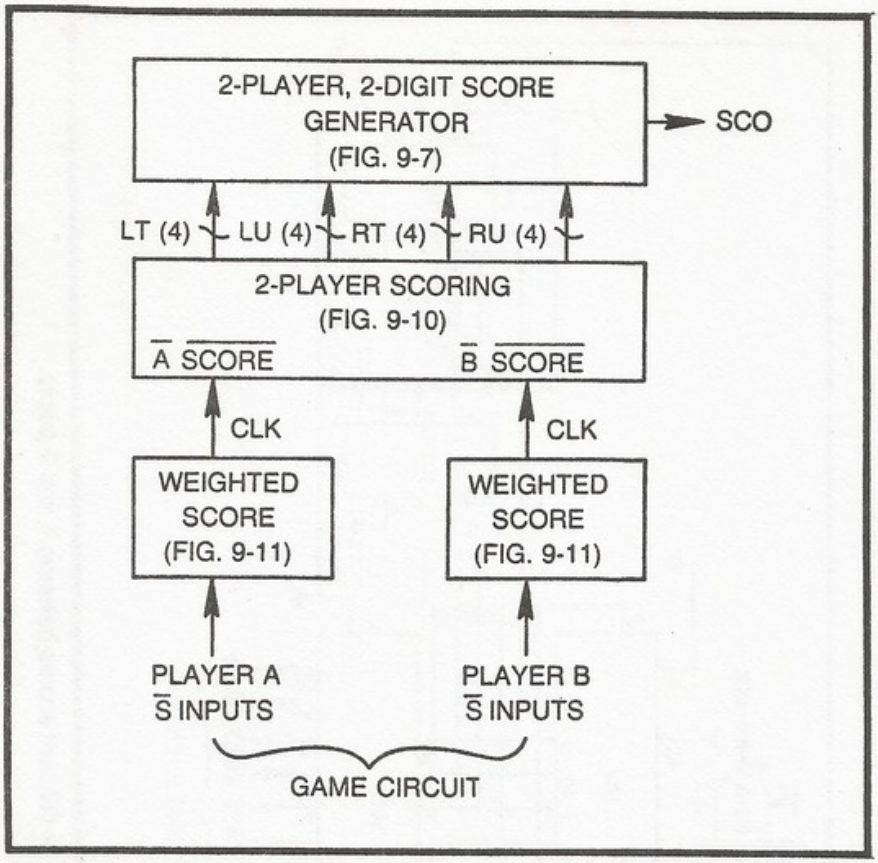


Fig. 9-12. Interfacing the weighted scoring circuits with 2-player scoring and full score character generator.

previously in this chapter. Whether used in this particular configuration or any other scoring scheme, the weighted-scoring circuit merely replaces the CLK or score-event pulse inputs to the scoring counters.

The Pinball game in Chapter 11 uses the weighted-scoring scheme in its most sophisticated form.

TIMEKEEPING CIRCUITS

Timekeeping operations are relatively straightforward: Pick up a 1-Hz source of pulses, apply them to a counter that counts seconds and minutes, then attack any sort of desired automatic start, stop, and clearing controls.

Figure 9-13 shows how to generate a precise 1-Hz timing pulse from the 60-Hz VRST source. The scheme uses two counter circuits composed of a 7492 cascaded with a 7490. The 7492 is a special clock-operation binary counter that automatically counts the sequence 0 through 5, then resets to 0 again. This particular counter

serves the dual function of a divide-by-6 frequency counter and a binary counter for 10s of seconds. In Fig. 9-12, the 7493 is used as a divide-by-6 circuit.

VRST pulses applied to the pin-1 input of the 7492 thus emerge from pin 8 at the rate of 10-Hz (60 Hz divided by 6). The 10-Hz pulses are then applied to the pin-14 input of a 7490 BCD counter which is connected as a symmetrical divide-by-10 counter. The 7490 thus divides the 10-Hz signal from the first counter to a precise 1-Hz waveform having a duty cycle of exactly 50%. This 1-Hz signal serves as the main clocking source for all timekeeping operations.

The circuit generates its 1-Hz output only as long as the START input is at logic 0. Whenever START input goes to logic 1 for any reason, the 1-Hz pulses no longer appear at the output because the counters are effectively cleared and stopped. Pulling the START input back down to zero makes the 1-Hz pulses appear once again.

This 60-Hz-to-1-Hz converter, or frequency divider, is incorporated in the most useful kind of timekeeping circuit, shown in Fig. 9-14. VRST is applied to pin 1 of IC3, and as long as the game is in its PLAY mode, the precise 1-Hz pulse appears at pin 11 of IC5.

IC6 in Fig. 9-14 serves the dual purpose of a BCD counter and a divide-by-10 frequency divider. The 1-Hz pulses appearing at its pin-14 clocking input increment this counter at the 1-Hz rate, making its four outputs generate the appropriate BCD code for units of seconds.

The most-significant-bit output of IC6 changes at a rate of once every 10 seconds, the input clock rate of 1 Hz divided by ten. This output pulse clocks IC4, another 7492 divide-by-6 counter. The three outputs of IC4 increment once every 10 seconds, generating the 10s-of-seconds cycle of 0 through 5.

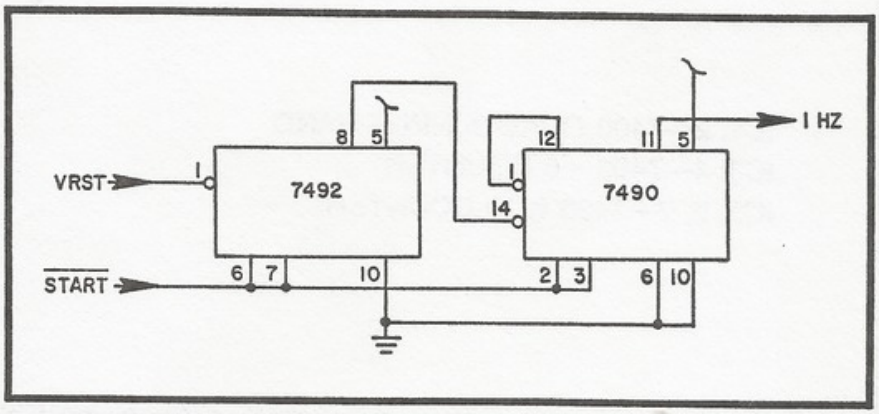
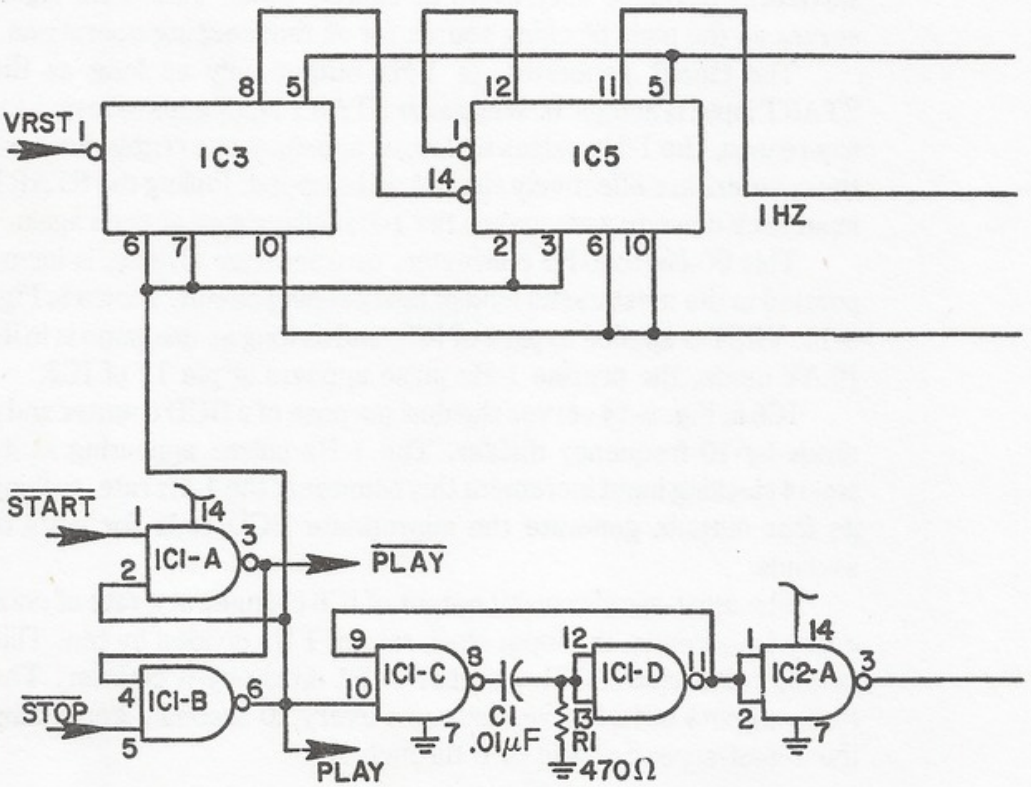
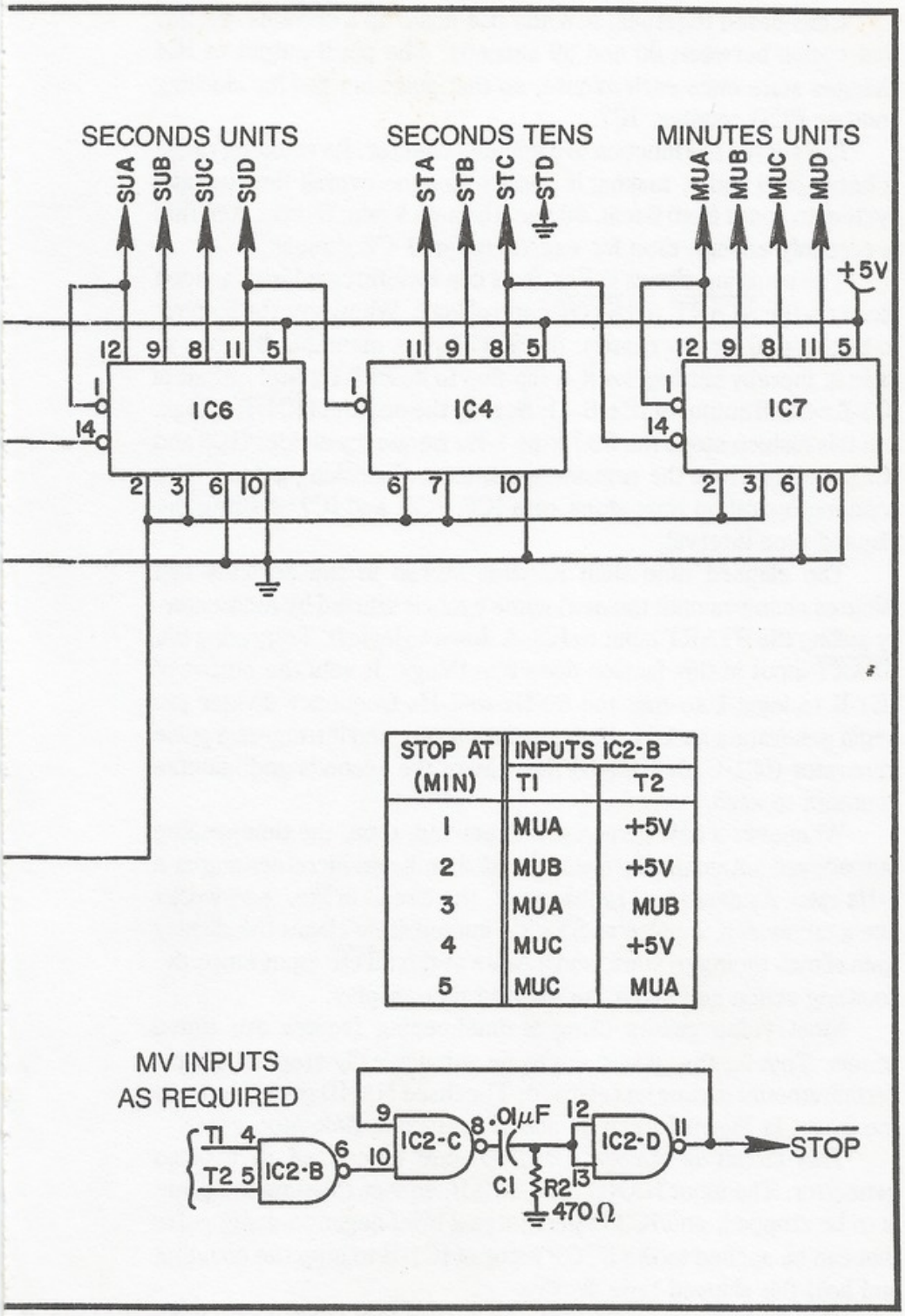


Fig. 9-13. A simple frequency divider circuit that converts the 60 Hz VRST pulse from Sourcebox into 1 Hz timing pulses.



IC1, 2—7400 QUAD 2-INPUT NAND
IC3, 4—7492 ÷6 COUNTER
IC5, 6, 7—7490 BCD COUNTER



STOP AT (MIN)	INPUTS IC2-B	
	T1	T2
1	MUA	+5V
2	MUB	+5V
3	MUA	MUB
4	MUC	+5V
5	MUC	MUA

Fig. 9-14. A timekeeping circuit that can measure elapsed game time from 0 min. 00 sec. to 9 min. 59 sec.

Considered together, IC6 and IC4 make up a seconds counter that cycles between 00 and 59 seconds. The pin-8 output of IC4 changes state once each minute, so that pulse is used for clocking another BCD counter, IC7.

IC7 serves the function of a minutes counter. Its counting range is between 0 and 9, making it possible for the overall timekeeping system to count from 0 min. 00 sec. through 9 min. 59 sec. And that is certainly enough time for executing good TV games.

The counting circuit in Fig. 9-14 can be interfaced with a video game via the $\overline{\text{START}}$ and $\overline{\text{STOP}}$ control logic. Whenever the game is to be stopped for any reason, the STOP input momentarily goes to logic 0, thereby setting the $\overline{\text{R}}$-$\overline{\text{S}}$ flip-flop to its PLAY state: output of IC1-A=0 and output of IC1-B=1. Setting the output of IC1-B to logic 1 in this fashion stops the 60-Hz-to-1-Hz frequency divider (IC3 and IC5), and deprives the remaining counters of clocking pulses. The counting operation thus stops with IC6, IC4, and IC7 showing the elapsed time interval.

The elapsed time then remains stored in the seconds and minutes counters until the next game cycle is started by momentarily pulling the START input to IC1-A down to logic 0. Triggering the $\overline{\text{START}}$ input in this fashion does two things: It sets the output of IC1-B to logic 1 so that the 60-Hz-to-1-Hz frequency divider can begin generating its 1-Hz timing pulses again, and it triggers a pulse generator (IC1-C and IC1-D) that clears the seconds and minutes counters to zero.

Whenever a new game cycle is started, then, the timekeeping outputs are automatically cleared and then begin incrementing at a 1-Hz rate. As described to this point, the circuit in Fig. 9-14 works like a stopwatch, a pulse at $\overline{\text{START}}$ immediately clears the display then allows timing to start, and a pulse at the $\overline{\text{STOP}}$ input stops the counting action and holds the elapsed time display.

Most video games using a timekeeping feature are timed games. That is, the games are to be automatically stopped after a certain amount of time has elapsed. The three NAND gates shown in the insert in Fig. 9-14 can be added to make a time-stop game.

This circuit is simply a NAND gate connected to a pulse generator. The input NAND gate, IC2-B, senses the time the game is to be stopped, and IC2-D generates a brief negative-going pulse that can be applied to the STOP input of IC1-B to stop the counting and hold the elapsed-time display.

The chart accompanying the circuits in Fig. 9-14 show which timing outputs should be connected to the inputs of IC2-B. If the game is to be stopped after 3 minutes, for instance, the inputs to

IC2-B should be MUA and MUB from IC7. Once the game is started by a negative-going pulse at the $\overline{\text{START}}$ input of IC1-A, it runs until MUA and MUB both go to logic 1 at the end of 3 minutes. The resulting negative-going pulse from IC2-D then stops the clocking operation.

If the game, itself, is controlled by the PLAY and $\overline{\text{PLAY}}$ outputs of IC1-A and IC-1B, the game operation is tied to the clocking operations, as the timer goes, so goes the game.

Figure 9-15 shows how the timekeeping circuit can be interfaced with the figure generator from Fig. 9-7 and the game logic system.

RETROFITTING SCORING AND TIMEKEEPING TO EXISTING GAMES

Many of the video games described in earlier chapters can be retrofitted with certain versions of the scoring and timekeeping circuits presented in this chapter. Retrofitting the circuits is generally a matter of adding two more circuit boards, calling for little, if any, surgery on existing systems.

Golf Score and Hole Designation

Perhaps the simplest game to retrofit is the Golf game featured in Chapter 8. Adding the circuit in Fig. 9-16 to the Golf game

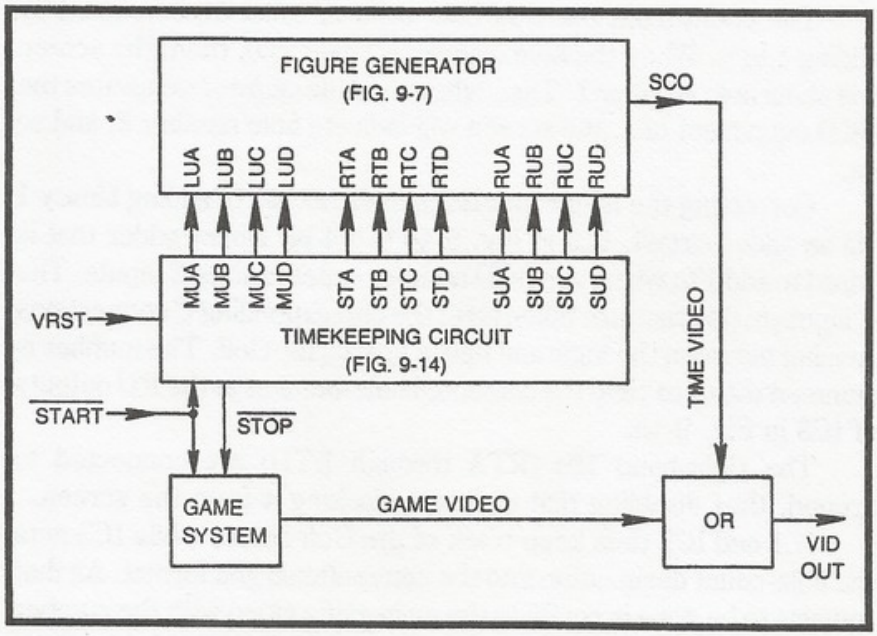


Fig. 9-15. Interfacing the timekeeping circuit with the character generator and some sort of video game that must be ended after a certain amount of time has elapsed.

provides a nine-hole score between 00 and 99, as well as a numeric designation of the hole being played.

IC1 and IC2 make up the scoring portion of the game. These two BCD counters are cascaded to count binary numbers 00 through 99, and they increment each time the pin-14 clock input of IC1 sees a positive-going pulse from BT, the ball timer in Fig. 8-18. Recall that this timer outputs a positive pulse each time the player depresses the SWING pushbutton. So the counter increments one unit each time the player takes a swing at the ball.

The tally can be reset to zero at the beginning of each game by depressing the SCORE RESET pushbutton. This pushbutton, of course, should be added to the Golf control panel.

The BCD outputs of IC1 and IC2 go to the numeric-figure generator circuit board shown in Fig. 9-7. Using the connections shown here, the Golf score appears on the screen in the left-hand set of digits.

The right-hand units numeral on the screen will show which hole is being played. IC9 in Fig. 8-18 is the hole counter for the Golf game. Its output is a BCD number between 0 and 8; so if its "C" outputs were applied directly to the RU inputs on the score figure generator, the player would see the holes being numbered between 0 and 8, instead of the proper 1 through 9.

The count from the Golf hole counter must be corrected by adding 1 to it. When the hole counter is reset to 0, then, the screen will show hole number 1. Then when the hole counter generates the BCD equivalent of 1, the screen will indicate hole number 2, and so on.

Correcting the hole count is a simple matter of adding binary 1 via an adder circuit. IC3 in Fig. 9-16 is a 4-bit binary adder that is wired to add 1 to whatever BCD number appears at its C inputs. The C inputs in this case are taken form the corresponding C connections running between the logic and figure boards for Golf. The number is summed with 1 to yield the corrected hole-number at the RU outputs of IC3 in Fig. 9-16.

The right-hand 10s (RTA through RTD) are connected to ground, thus disabling that digit and blanking it from the screen.

IC1 and IC2 thus keep track of the Golf score, while IC3 puts the hole-count designation into the conventional golf format. All that remains to be done is combine the main game video with the number video. IC4-A, IC4-B, and IC4-C in Fig. 9-16 take care of this job by effectively ORing together the game and number video.

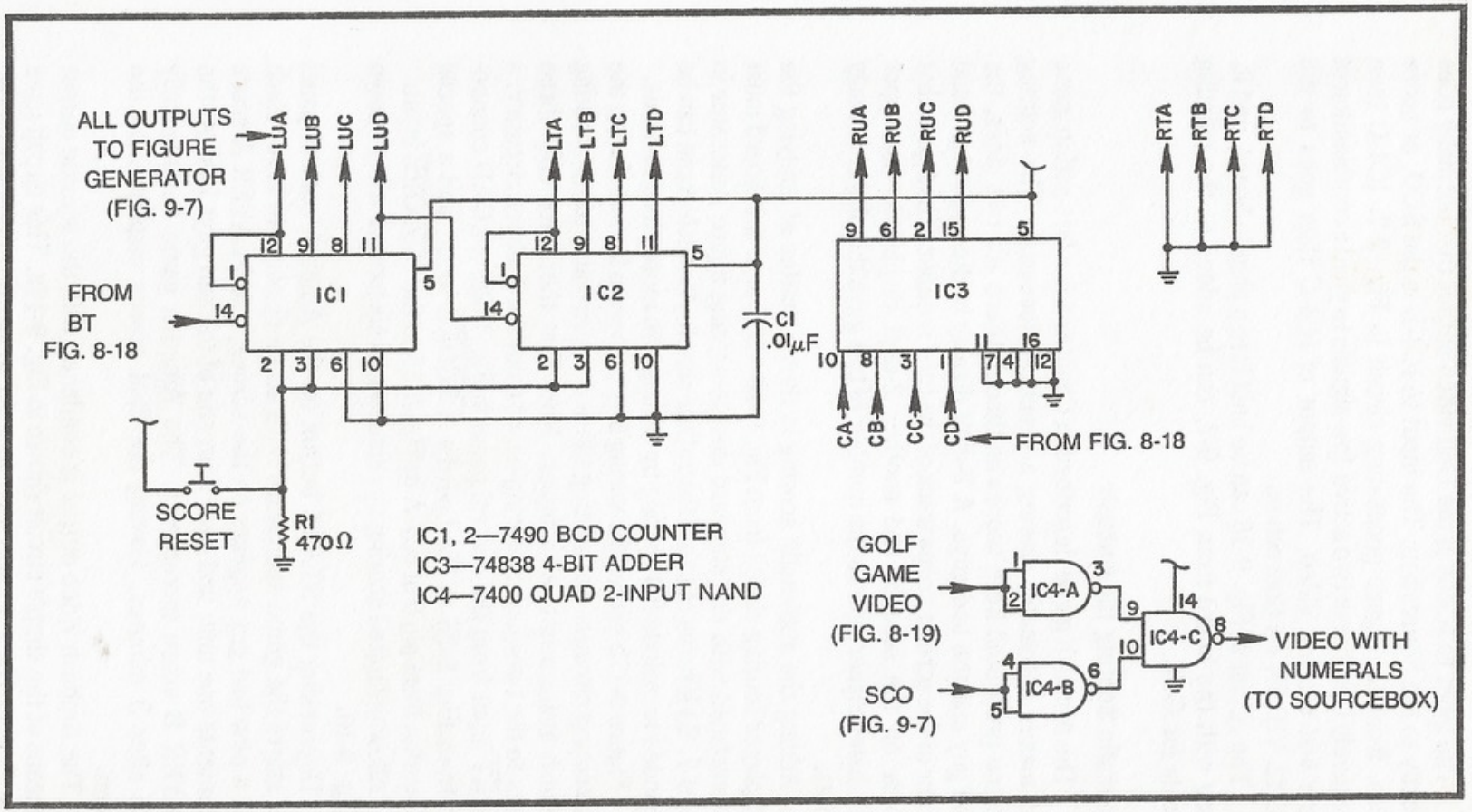


Fig. 9-16. Scoring and hole-numbering circuit for the game of Golf.

The input to IC4-A is the Golf video output that normally goes directly to the Sourcebox. The input to IC4-B is the SCO, or score video, from the figure-generating circuit in Fig. 9-7. IC4-C then effectively ORs these two active-low signals to yield a combination of game and scoring video. The output of IC4-C thus goes to the GAME VID IN on Sourcebox.

The circuit in Fig. 9-16 can be built into a plug-in board, and it, along with the board from Fig. 9-7, can be added to the existing boards for Golf.

Automatic Scoring for Ambush

The Ambush game described in Chapter 8 is a fast-action game that makes automatic scoring a practical necessity. The scoring scheme presented here scores as though there are two sides, the good guy and the bad guys. A 2-digit display in the upper left-hand corner of the screen keeps track of the total number of bad guys that appear on the screen, and another 2-digit display in the upper right-hand corner scores the number of bad guys the player actually shoots.

Adding the automatic scoring is thus a matter of applying the two-player scoring circuit from Fig. 9-10. Of course this board must be interfaced with the standard double-scoring figure generator in Fig. 9-7. So it turns out that the rather simple Ambush game can be expanded to include full scoring by adding two more circuit boards.

Figure 9-17b shows the wiring block diagram for interfacing the two scoring circuits. Connecting the scoring circuitry to the existing Ambush system is even simpler. Note that there are only three inputs to the two-player scoring board from the Ambush circuitry: a $\overline{\text{START}}$ input from the control panel in Fig. 8-22, a LOAD connection from Fig. 8-25 to the board's $\overline{\text{A SCORE}}$ input, and a special connection from pin 5 of IC7-A in Fig. 8-25 to the $\overline{\text{B SCORE}}$ input.

Allow unlimited scoring by removing all six jumper wires shown in Fig. 9-10.

Depressing the START button on the Ambush control panel thus starts the game and clears both score displays to zero. Each time a new bad guy appears on the screen, the PLAYER A score increments one unit; and each time one of the bad guys is shot, the PLAYER B score increments. The Ambush game automatically ends after 3 minutes, leaving the final scores displayed on the screen.

The Ambush video output is combined with the scoring output by means of the simple circuit shown in Fig. 9-17a. This circuit uses three 2-input NAND gates that happen to be uncommitted on the

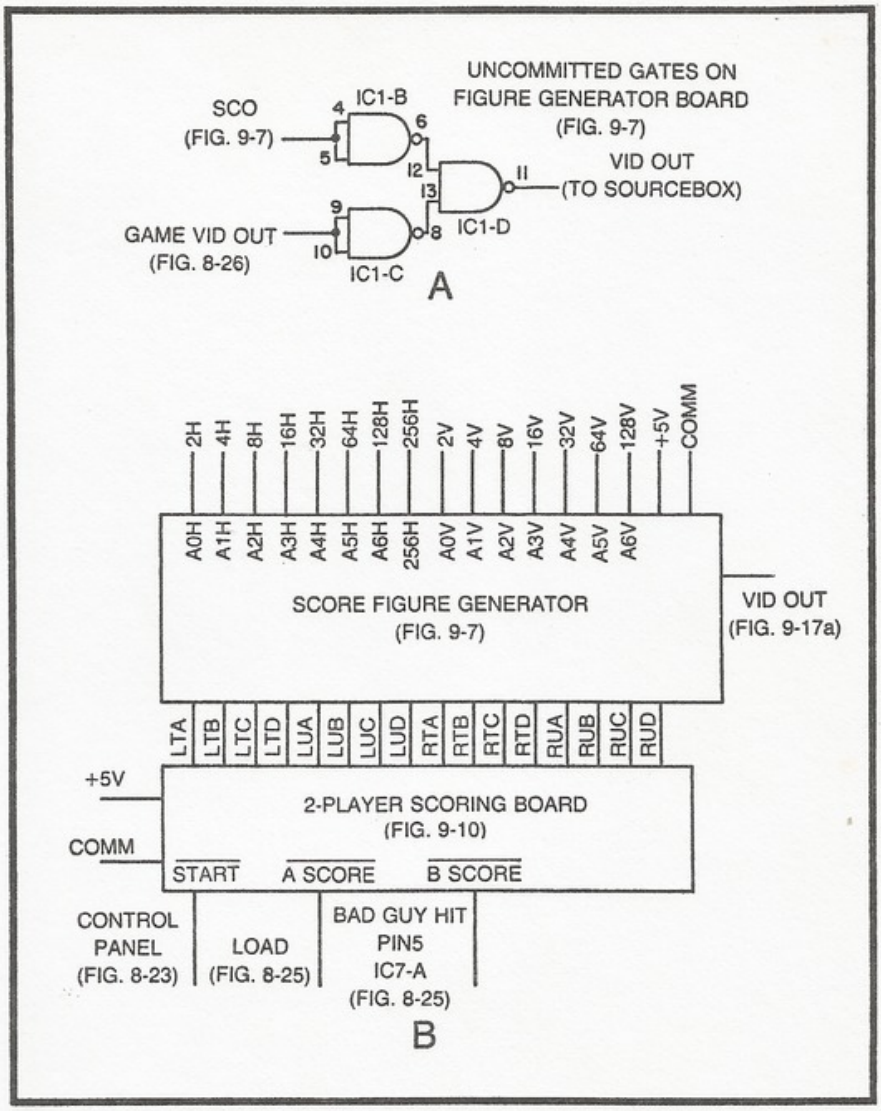


Fig. 9-17. Scorekeeping circuit for Ambush.

score-figure generator. Wire IC1 on the score-figure generator as shown here, applying the SCO terminal from Fig. 9-7 to the inputs of IC1-B and the GAME VID OUT from Fig. 8-26 to the input of IC1-C (instead of to the Sourcebox).

Connecting the pin-11 output of IC1-D to the GAME VID IN of the Sourcebox completes the operation. Ambush with full scoring is then ready to go to work for you.