ADD **DOUBLES** TO YOUR COMPUTER

BY JAMES BARBARELLO

Peripheral allows two to play computer games without crowding the keyboard.

MICROCOMPUTERS can be programmed to play many interesting games, especially those involving two human players (as opposed to one person playing against the computer). Usually, this involves each player, in turn, operating a specific key on the keyboard. Some systems use two joysticks with associated pushbuttons so that two players can compete without getting in each other's way. However, many systems do not include joysticks, so two-person games played on a keyboard become a little difficult when the action gets hot and heavy.

The Doubles peripheral presented here (Fig. 1), consists of a battery-powered two-IC circuit with two independent pushbutton switches, each at the end of a length of slender two-conductor cable. When connected to a cassette port, for example, and used with a simple BASIC program, the two players can now participate without interference in any game that requires only a simple key closure as an input. Although designed for a TRS-80 Model I, the approach can be adapted to almost any system whose BASIC includes OUT and INP (or their equivalents), with the "called" port easily accessible.

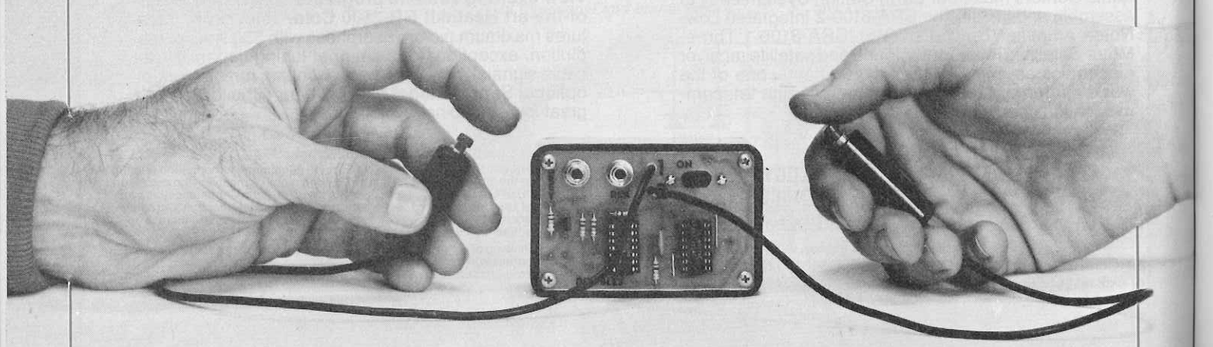
In the case of the TRS-80, when the BASIC command OUT 255,1 is issued, a logic-1 voltage level is present at output port 255 (cassette), while OUT 255,0 drops this voltage to the logic-0 level.

The command INP(255) senses the voltage level between the tip and ring of the cassette plug. If a logic 0 is present, the command will return the number 127. If a logic 1 is present at the cassette plug, the number 255 is returned.

Thus, by choice of OUT and INP commands, it is possible to communicate with an external circuit connected to the cassette port, and receive back its current status.

Circuit Operation. Two identical flip-flops, each having independent data, set, reset, and clock inputs, and Q and $\overline{Q}$ outputs, are contained in *IC1*. The logic level present at the D input is transferred to the Q output during the positive-going transition of the clock pulse.

When power is first applied (via *S3*), *C1* and *R3* produce a momentary pulse that loads input D of *IC1A* with



a logic 1 and input D of *IC1B* with a logic 0. This occurs because of the set/reset action. Each time a clock pulse is applied to the clock inputs, the data on pin 5 is shifted through the Q output to the data input of *IC2B*, and the Q output of *IC1B* is loaded into the D input of *IC1A*. The net result is that, with clock pulses, a logic 1 alternates between the $\overline{Q}$ outputs of *IC1*.

Clock pulses from the computer output port are fed via *J1* and *R1* to the base of *Q1*. The output of *Q1* is connected to *IC2D* which acts as an inverter. The output drives the clock inputs of *IC1* and pin 9 of NAND gate *IC2C*.

The two inputs of gate *IC2B* are tied high via *R4* and *R5*. If both *S1* and *S2* (the player switches) are open, the output of *IC2B* is low and *IC2C* is high. With both inputs of *IC2A* high, a low is present at *J2*.

Without connections to the computer, if either player closes his switch, the logic states change and a high appears at *J2*.

When the circuit is connected to the computer and power applied, the circuit defaults with a logic 1 in *IC1A*. The simple software has made the computer aware that this condition has occurred, and from then on, each time the computer issues a clock pulse, it "knows" which flip-flop is being used. Since the computer can issue clock pulses very fast, both operators appear to be on-line at once.

Construction: The Doubles circuit can be built up on a perfboard, or a printed-circuit board constructed using the pattern of Fig. 2. Before handling the CMOS ICs, touch a ground point (such as the screw holding an outlet cover) to remove any static charge you might have. Mount jacks *J1*, *J2*, and switch S3 on the pc board as shown in Fig. 3. Connect battery clip *B1* at this time.

The final step is to construct and attach two switch assemblies. Each of these consists of a normally open pushbutton switch, about 18″ of thin (1/8″ diameter) coaxial cable and a 1/4″ plastic phono plug cover. Strip one end of the coax cable and solder the center conductor to either switch lug. Solder the outer braid to the remaining switch lug. Make a knot in the cable about 1″ from the switch lugs and pass the free end of the coax cable through the phono plug cover so that it exits the smaller end. Tighten the nut on the switch and then coat the sides of the switch with epoxy or a similar adhesive and push it into the large end of the cover as far as it will go. Wipe off any excess adhesive and let set. Repeat for the other switch.

When both switch assemblies are complete, pass the cables through the two 1/8″ holes in the pc board (so that the cable ends are on the *foil* side). Strip the cable ends and attach to the board as shown in Fig. 3.

Attach a 9-volt battery to the battery clip. The unit can be used "as is" or can be mounted in a cabinet. (The pc board has been designed to replace the top cover of Radio Shack's Experimenter Box #270-231. The battery is taped to the inside of the box.)

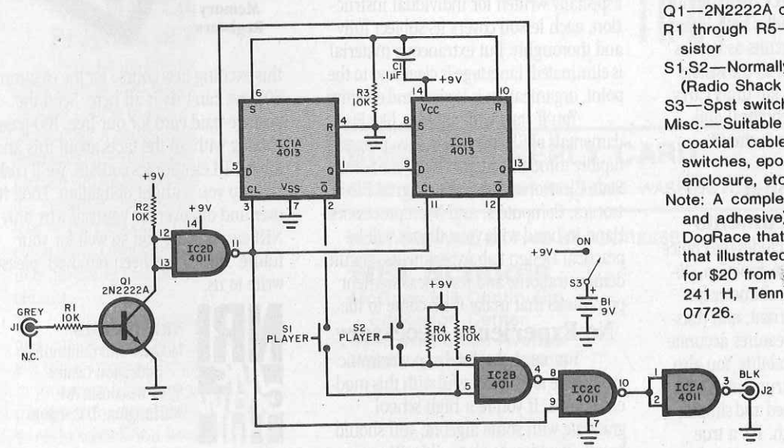


Fig. 1. The circuit consists of two IC flip-flops with the pushbuttons attached.

Testing. Using the TRS-80 Model I or III as an example, connect the grey (large) cassette plug to *J1*, and the black plug to *J2*.

Enter the following short program into the TRS-80:

```
10 CLS
20 OUT 255,1: OUT 255,0
30 OUT 255,1: OUT 255,0:
  IF INP(255)=255 THEN PRINT "1",
40 OUT 255,1: OUT 255,0:
  IF INP(255)=255 THEN PRINT "2",
50 GOTO 30
```

Line 10 clears the screen, and line 20 initializes the Doubles circuit. Lines 30 and 40 alternately check for a closure of *S1* and *S2* respectively. Line 50 loops back to line 30, thus creating an endless loop that continuously checks the status of the switches. Turn switch *S3* ON and RUN the program.

Press switch *S1* and note that a series of 1s appears on screen as long as this switch is depressed. Release *S1* and close *S2*. Note that a series of 2s now appear. Holding both switches down simultaneously will cause alternating 1s and 2s. You can now identify and mark *S1* and *S2* for future use in playing.

Using the Circuit. To use the Doubler effectively, the computer must know which switch was operated, which was depressed first, and whether a switch was depressed when it should not have been.

PARTS LIST

B1—9-volt battery
C1—0.1-μF disk capacitor
IC1—4013 dual D flip-flop
IC2—4011 quad 2-input NAND gate
J1,J2—1/8″ phono jack
Q1—2N2222A or similar
R1 through R5—10,000-ohm, 1/4-watt resistor
S1,S2—Normally open pushbutton switch (Radio Shack 275-1547 or similar)
S3—Spst switch
Misc.—Suitable lengths of 1/8″ diameter coaxial cable, plastic enclosures for switches, epoxy, battery holder, suitable enclosure, etc.
Note: A complete kit of parts (less case and adhesive), including a game called DogRace that uses a program similar to that illustrated in this article, is available for $20 from J.J. Barbarello, RD 1, Box 241 H, Tennent Rd., Englishtown, NJ 07726.

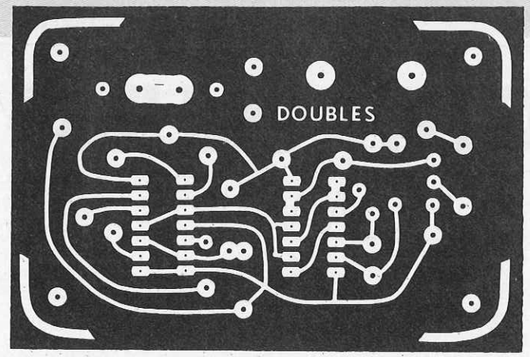


Fig. 2. Use this same-size foil pattern for the pc board.

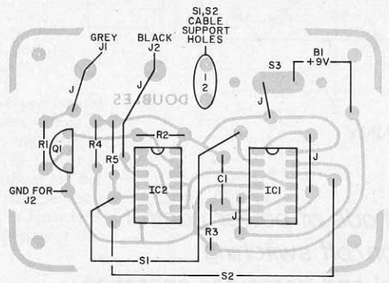


Fig. 3. Component layout for the pc board.

Enter the following sample program:

```
10 CLS: OUT 255,1: OUT 255,0
20 PRINT @ 408, "** GO **"
30 FOR J = 1 TO 20
40 FOR I = 1 TO 2: OUT 255,1: OUT
  255,0
50 IF P(I)<>255 THEN P(I) = INP(255)
60 IF P(I)=255 AND P(0)=0 THEN
  P(0)=I
70 NEXT I,J: PRINT @ 408,
  STRING$(8,32)
80 IF P(1) = 255 THEN PRINT "#1",
90 IF P(2) = 255 THEN PRINT "#2",
100 PRINT "#";P(0);"FIRST"
110 FOR I=0 TO 2: P(I)=0: NEXT I
120 FOR J = 1 TO 10 + RND(30)
130 FOR I = 1 TO 2: OUT 255,1: OUT
  255,0
140 IF P(I)<>255 THEN P(I) = INP(255)
150 NEXT I
160 IF P(1)=255 THEN PRINT@408,
  "FOUL #1":P(1)=0
170 IF P(2)=255 THEN PRINT@408,
  "FOUL #2":P(2)=0
180 NEXT J
190 FOR I = 0 TO 2: P(I)=0: NEXT I
200 GOTO 20
```

Lines 20 through 110 are the scoring lines, after line 20 prints ** GO **, line 30 causes the computer to loop through lines 40 through 70 twenty times. Each time line 40 "clocks" the Doubles circuit, line 50 "looks" for a switch closure. If a switch closure has been made, line 60 checks to see if it is the first. If it is, that switch number is stored as variable P(0). Lines 80 through 100 print out the number of the depressed switches, as well as which was closed first. Line 110 clears the variable P(0), which was first; P(1), switch 1 indicator; and P(2), switch 2 indicator.

Lines 120 through 180 are similar to lines 30 through 70, but these check for foul. The number of circuit scans is randomly chosen in line 120. Since there is no need to know which switch closed first in a foul situation, P(0) is not used. On each scan of the circuit, lines 160 and 170 check for a foul. If so, this information is printed. Line 190 clears variables P(0), P(1), and P(2), while line 200 loops the program back to the beginning.

To use the program, enter and run it. When ** GO ** appears on the video display, either or both switches may be operated. The switch number as well as which switch was operated first will appear on screen. If a switch was depressed before the ** GO ** appeared, then the "foul" indicator will appear. This short program can be used as the basis for many game programs, with scoring and graphics added if desired. ◇