AN INTEGRATED CIRCUIT CW ID GENERATOR

*P. J. Ferrell W7PUG
6021 S. 119th St.
Seattle WA 98178*

A project to develop an automatic, solid-state CW ID generator was recently initiated by members of the Seattle repeater group. Although there have been a number of recent articles concerning such devices,[1-2-3-4] our starting point was the FM Magazine article by Woore.[2] The outcome of this project must be classified as an engineering overkill. The resulting CW identification generator features a clocked character generator (for flawless CW with variable speed), inexpensive RTL integrated circuitry, and a computer-designed diode read-only memory matrix. Also included are "pulse" starting, a discrete "hold" voltage available during ID execution, and a continuously adjustable keying speed (from far too slow to far too fast). Not only are these generators ideal for repeater identification, but they may be used to identify any amateur station such as RTTY, ATV, etc.

The block diagram in Fig. 1 shows the major divisions of the ID generator. Many excellent articles covering RTL logic design have appeared in amateur literature[5-6] and the reader is referred to them for background material.

Program Counter and Start/Stop Flip-Flop

This six-stage ripple counter consists of three dual JK flip-flips. The first five cascaded stages are the program counter and count from 0 to 31. The last stage is employed as the start/stop flip-flop. Each stage of a ripple counter is arranged to toggle (change state) on the output of the preceding stage. A five-stage program counter has 32 distinct stages ($2^5=32$). When arranged as shown in Fig. 2, the program counter advances under the control of gate G1 which derives its input from the character generator. Each dot, dash, or blank character advances the program counter by one count. The last character (number 32) resets or clears the first five stages, but toggles the start/stop flip-flop to the "stop" or set position, thereby halting the character generator.

A positive pulse into the "preclear" input of FF6 clears the halt and allows the character generator to run, thereby initiating a cycle of operation. A five-stage program counter was chosen since virtually all amateur calls can be encoded in 32 characters worth of dots, dashes, and blanks. RTL JK flip-flops are adversely affected by capacitive output loads, and will not toggle reliably if the capacitance is too high. This fact precludes the use of silicon diodes in the *and* portion of the diode memory.

If the program counter output lines were buffered (isolated from the flip-flops

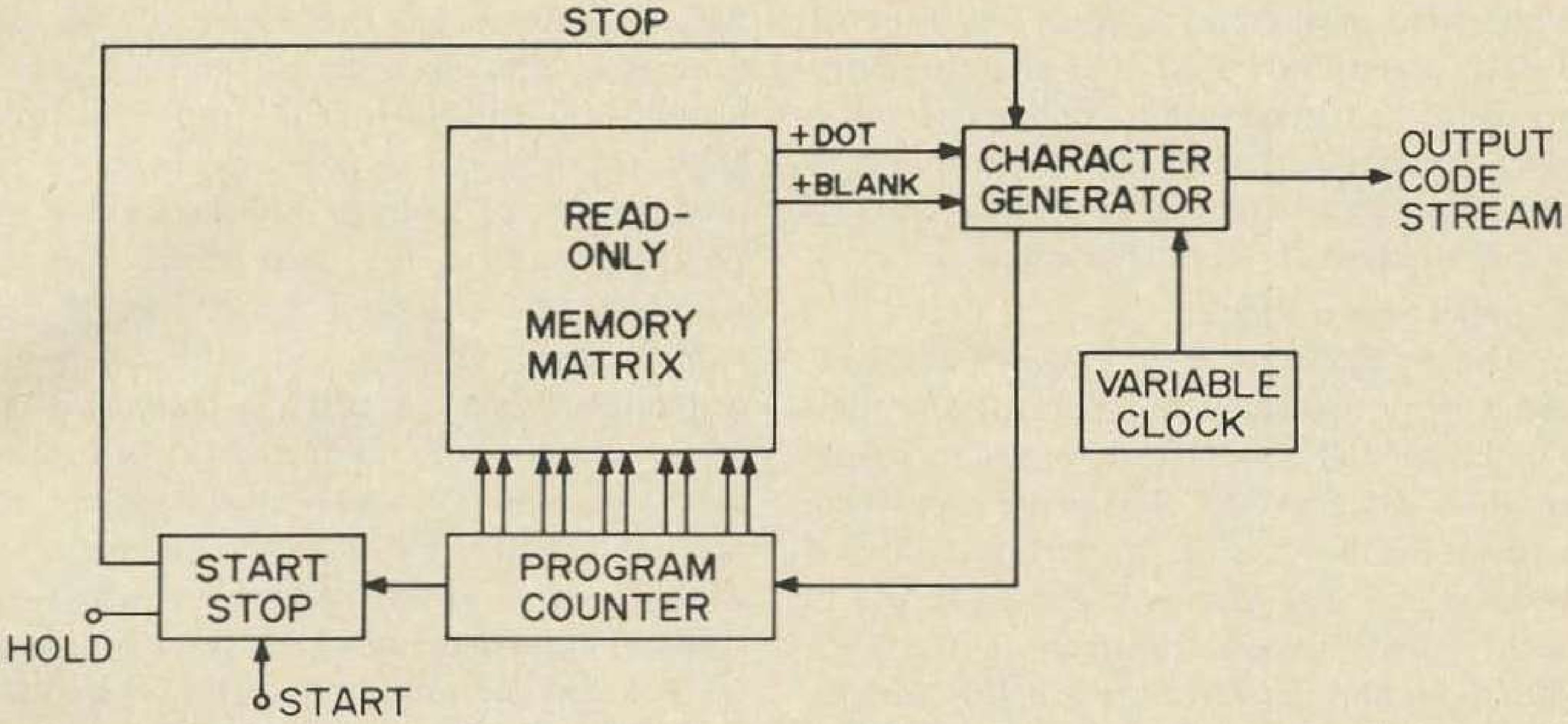


Fig. 1. CW ID generator block diagram.

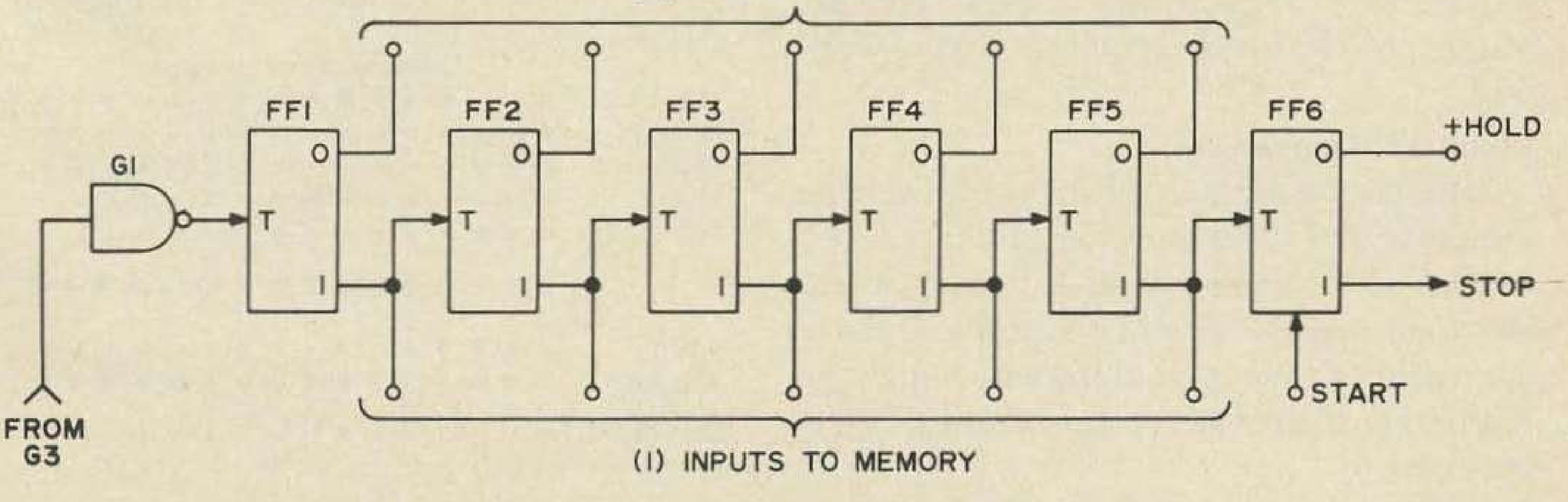


Fig. 2. Program counter.

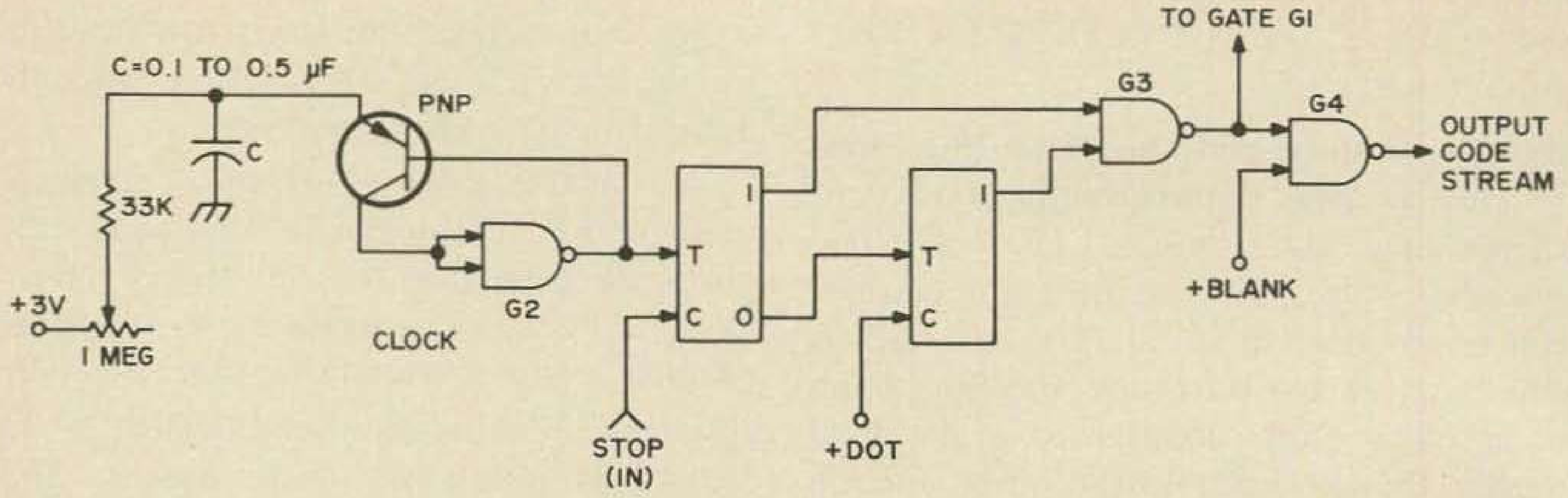


Fig. 3. Clock and character generator.

with gates or inverters), then any type of diode could be used in any memory position; but germanium diodes have very low capacitance and may be connected directly to the flip-flop outputs, thereby saving the cost of 10 buffer stages.

Variable Speed Clock

The clock circuit must deliver a negative-going pulse with leading edge of less than a microsecond duration in order to toggle an RTL JK flip-flop. The pulse repetition rate should be variable to permit choice of code speed. The circuit is shown in Fig. 3 with a PNP silicon transistor paired with gate G2. The net effect is a PNPN switch. Capacitor C charges to about 2V and then discharges through the gate with a leading edge which is very abrupt.

Nearly any of the new PNP silicon transistors will work in this circuit. The minimum value for R is about 33 kΩ else sufficient current is available to hold the switch in conduction (just like a neon relaxation oscillator). For R much above 1 MΩ insufficient current is available to initiate the regenerative "snap" action. Values of R between these limits work well.

Charácter Generator

The electronic generation of Morse code requires the creation of dots, spaces, dashes, and blanks which have a precisely specified relative length. The dot and space are each of one unit duration, while the dash and blank are each of three units duration.

An extremely clever character generator was borrowed from the Micro-Ultimatic Keyer[7] and forms the heart of the ID generator. The character generator consists of two JK flip-flops (FF7 and FF8) and gates G3 and G4 as shown in Fig. 3. A positive (stop) voltage on terminal C of FF7 holds it in the clear state, thereby stopping the character generator. If the stop voltage is removed, the character generator toggles in such a manner as to produce a string of dashes at the output of gate G4. If a positive (+DOT) voltage on terminal C of FF8 changes the string of dashes into a string of dots. A dash (or blank) requires four clock pulse intervals while a dot requires two. Gates G1 and G4 each invert the output of gate G3.

The output of gate G1 must be either a dot or dash character (never a blank), and is used to advance the program counter at the end of each character. The second input to gate G4 will blank out the output code stream, and is used to produce a blank character. If a blank is required, then a positive "+BLANK" input from the diode memory causes the dash generated by the character generator to be blanked

polarity of the (I) output

FF5	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
FF4	o	o	o	o	o	o	o	o	+	+	+	+	+	+	+	+
FF3	o	o	o	o	+	+	+	+	o	o	o	o	+	+	+	+
FF2	o	o	+	+	o	o	+	+	o	o	+	+	o	o	+	+
FF1	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+DOT	o	+	+	o	+	o	o	+	o	o	o	o	o	+	+	+
+BLANK	o	o	o	+	o	+	+	o	o	o	+	o	o	o	o	o
CHARACTER	—	·	·	x	·	x	x	·	—	—	x	—	—	·	·	·
		D			E				W					7		

Fig. 4. Diode memory inputs and outputs.

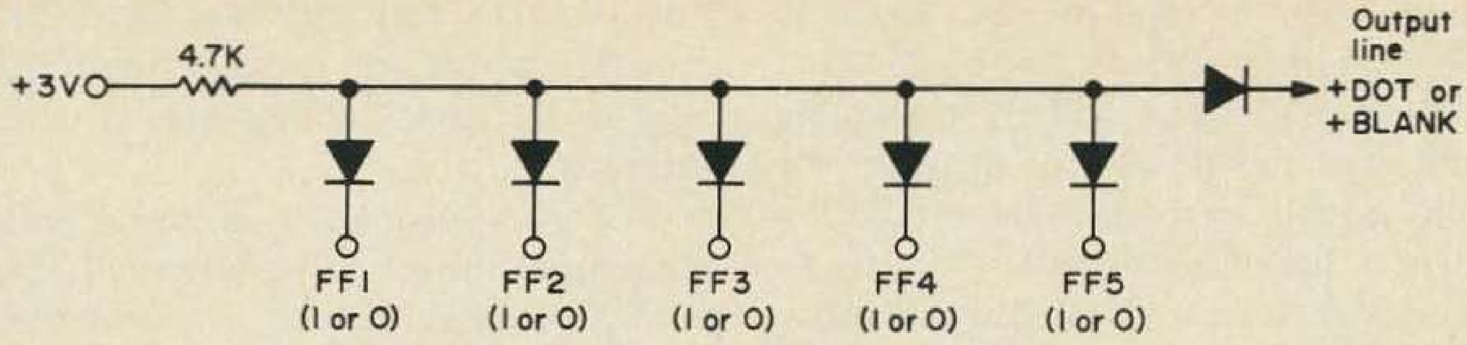


Fig. 5. Diode decoder for program counter.

out – which results in the transmission of a blank character.

Thus, the role of the diode read-only memory matrix is to provide just that sequence of +DOT and +BLANK inputs to the character generator which results in the transmission of the desired code stream. Gate G4 is the output with a plus representing "key down" and a zero representing "key up."

Diode Read-Only Memory Matrix

This is the hard part! Each desired code stream requires a distinct and different read-only memory design. A +DOT voltage must be produced by the diode memory for each program counter state that corresponds to a dot in the desired code stream, and a +BLANK is required for each blank character. An example is presented in Fig. 4.

FF5	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
FF4	o	o	o	o	o	o	o	o	+	+	+	+	+	+	+	+	o	o	o	o	o	o	o	o	+	+	+	+	+	+	+	+
FF3	o	o	o	o	+	+	+	+	o	o	o	o	+	+	+	+	o	o	o	o	+	+	+	+	o	o	o	o	+	+	+	+
FF2	o	o	+	+	o	o	+	+	o	o	+	+	o	o	+	+	o	o	+	+	o	o	+	+	o	o	+	+	o	o	+	+
FF1	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+	o	+
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	x	—	·	·	x	·	x	x	x	·	—	—	x	—	—	·	·	·	x	—	·	·	x	—	·	·	·	x	·	·	—	·

D E W 7 D B F

				●	●	●	●
	●		●	●	●	●	●
●		●			●		
●				●			

DOT MAP

0	4	12	8	24	28	20	16
1	5	13	9	25	29	21	17
3	7	15	11	27	31	23	19
2	6	14	10	26	30	22	18

5-INPUT KARNAUGH MAP

X	X	X	X				
	X			X			
	X					X	X

BLANK MAP

Fig. 6. Using the Karnaugh map.

Suppose that the first letters of the desired code stream were "DE (blank, blank)W7". The program counter states are shown corresponding to the required outputs from the diode memory matrix. A dash is seen to be the "default" condition; i.e. if neither a +DOT nor a +BLANK occurs, a dash results. Each of the 32 program counter states must be accounted for since they all appear on each ID execute cycle.

Figure 5 shows the method for decoding the program counter. The five terminals marked FF1 through FF5 are connected to either the **1** or **0** side of the respective flip-flop. If *any* of the five connections is low, then the whole common line is low. The only time the common line can be high is when all five input connections are high. For any given connection, this will occur exactly once during each program counter cycle. This type of diode decoder is often called an *and* gate since it has a high output only when all inputs are high.

Because of the diode output from the common line, these decoders may be paralleled to obtain the required +DOT and +BLANK functions. This paralleling is often referred to as an *or* gate since *any* high input results in a high output.

In the example of Fig. 6, the desired code stream has 15 dots and 9 blanks. If we employ a separate diode decoder for each one, then a total of 24 decoders would be required: 15 would be paralleled to give the +DOT signal and the remaining 9 would be paralleled to provide the +BLANK signal. Each decoder requires 6 diodes for a grand total of 144 diodes to build a straightforward read-only memory using this technique.

Fortunately for us, the English philosopher George Boole published his *Investigation of the Laws of Thought,* in which he resolved the ambiguity of the words *and* and *or* by means of a kind of algebra. In 1938, eighty-four years later, Prof. D. E. Shannon (the *Information Theory Shannon*) put Boole's algebra or Boolean Algebra to use in the *Symbolic Analysis of Relay and Switching Circuits.* This classic paper has revolutionized switching design, and has led to the development of minimization techniques which can dramatically reduce the diode count of our ready-only memory. The details of these methods and the underlying theory are beyond the scope of this article, but for those who are fascinated by this stuff, standard texts are available which will quickly dispel the aura of "black magic" that seems to surround this area.[8]

For our purpose, a graphical reduction technique known as a *Karnaugh map* will be employed. Figure 6 illustrates the process for the code stream *DE W7DBF.* The polarity of the **1** output levels of flip-flops FF1 through FF5 are shown as the program counter steps from **0** (all FFs clear) to state 31 (all FFs set). The sample code stream begins with a blank and has three blanks separating the **DE** from the **W**

A Karnaugh map organization of program counter states is presented, flanked on the left by the DOT map and on the right by the BLANK map for the desired code stream. Reduction is accomplished by "folding" the map about any of the dividing lines and pairing the marks (dots or blanks) which overlap. For example, folding the DOT map about the vertical centerline pairs dot 5 with dot 21, and dot 9 with dot 25, and dot 15 with dot 31.

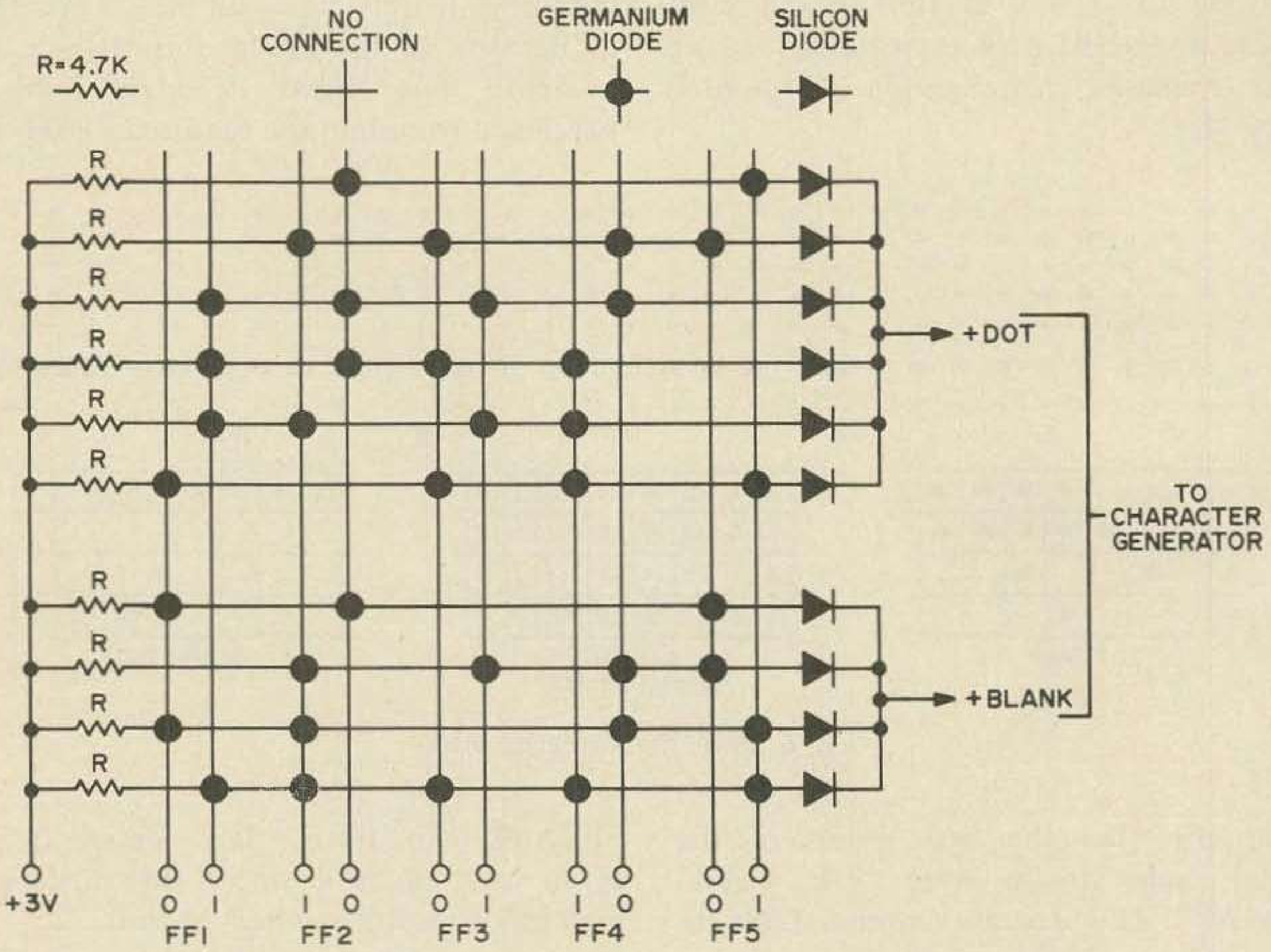


Fig. 7. Diode read-only memory matrix for "DE W7DBF."

Successive pairing, then pairing pairs, etc. allows a reduction from the original 144 diodes to a total of 48 diodes arranged as shown in Fig. 7. Note that in Fig. 7, the **0/1** flip-flop lines are reversed for FF2 and FF4. This reversal materially simplifies printed circuit construction.

FF5	o o o	o o o	o o o
FF4	o +	o +	
FF3	o o = o	+ + = +	o + =
FF2	o o o	o o o	o o o
FF1	o o o	o o o	o o o
	0 8	4 12	Pairing A and B
	(A)	(B)	(C)

Fig. 8. Actual reduction procedure.

As an example of pairing, consider the blanks in positions 0, 4, 8, and 12 of the BLANK map of Fig. 6. If the map is folded about the second vertical line (the one separating positions 4 and 12), then blank 4 pairs with blank 12, and blank 0 pairs with blank 8. If we fold again, then all four blanks coincide is illustrated in Fig. 8. In part A, the program counter states for positions 0 and 8 are compared. They differ in exactly one FF position (FF4) as all pairs must. A single diode decoder of the type shown in Fig. 5 could get both blanks simply by neglecting to connect to FF4. It even saves one *and* diode.

Part B of Fig. 8 presents the same comparison for blank 4 and blank 12, and again they differ only in FF4. In part C, a comparison of the 0/8 with the 4/12 pairs shows that these differ only in FF3. Thus, if a three-input diode decoder of the type presented in Fig. 5 were connected to FF1 (**0**), FF2 (**0**), and FF5 (**0**), it would give a +BLANK for all four desired program counter states (0, 4, 8, and 12). Without this reduction, 24 diodes (six for each blank) would have been required rather than the four actually required. This diode decoder may be found in Fig. 7 as the first line in the +BLANK group.

The foregoing example illustrates the use of a Karnaugh map. The states which map, and then the actual reduction is performed as in Fig. 8. It should be noted that some states will not pair at all, such as blank 27 in Fig. 6. To pick up this blank, a full five-input diode decoder is required. From Fig. 6, we see that for blank 27, the program counter is in state "++0++" and the resulting diode decoder can be found in Fig. 7 as the bottom diode line.

The rule when pairing states is that the two program counter states can differ in exactly one flip-flop position. All other positions must agree, including any omitted connections as in Fig. 8, part C. If there is disagreement in more than one FF position, then these two states do not pair.

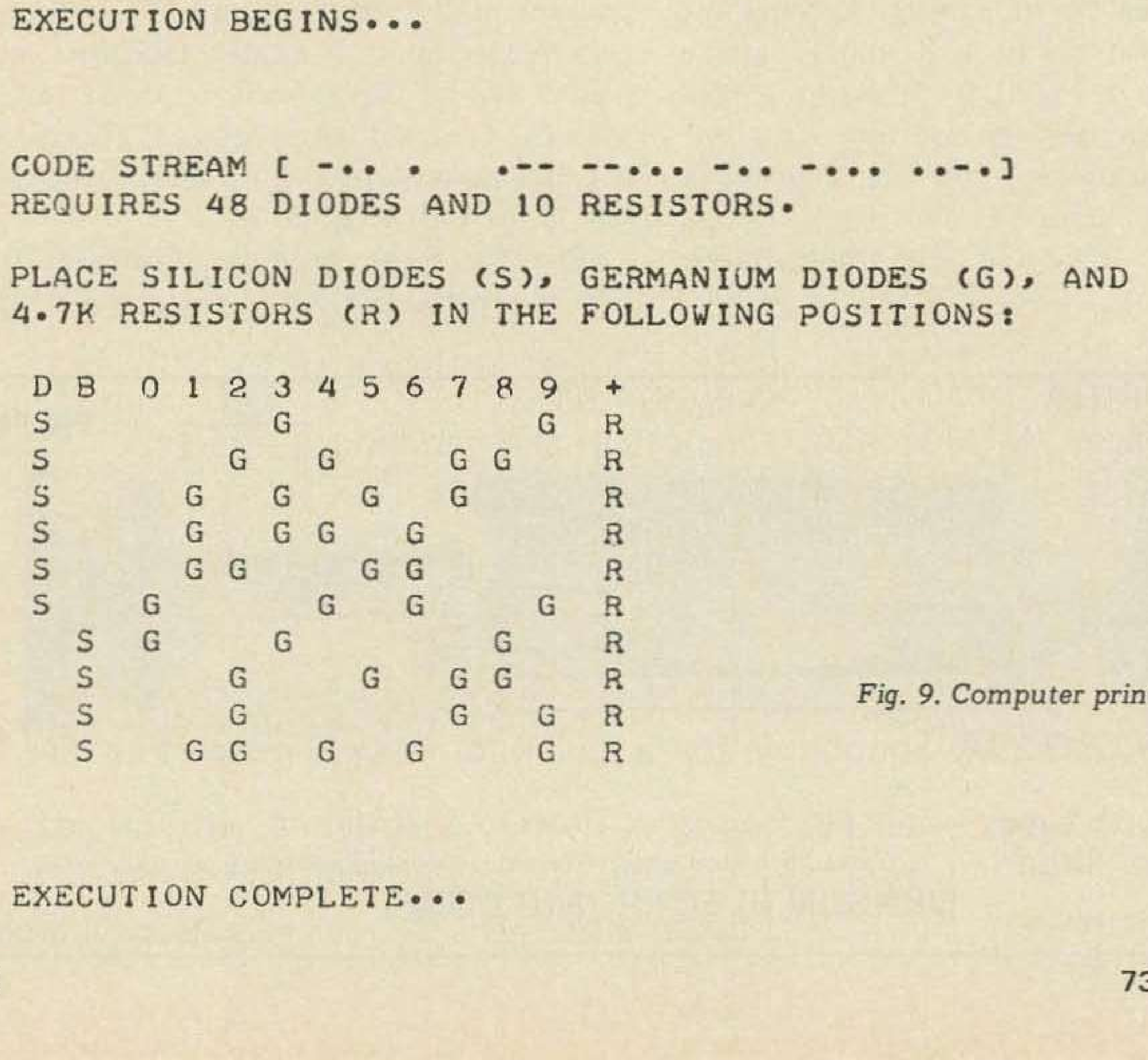
After completing the reduction process, check to make sure that every necessary state has been covered at least once by one of the final decoders; otherwise, you may be surprised at the resulting code stream. This type of calculation, once understood, is not particularly difficult, but it certainly is tedious and has lots of room for errors. Slight changes in the desired code stream (even position) can have a huge impact on the diode count.

For example, the diode count for the code stream in Fig. 6 is 48. If just the *DE* is slid one count to the right, a new code stream is formed which starts with two blanks, and has two blanks between the *DE* and *W*. The diode count for this new stream is 55. If this new stream is shifted two positions to the left, so that the two leading blanks become trailing blanks, then the diode count becomes 85. These effects are unpredictable, and for complete optimization each code-stream version must be reduced separately, and the results compared. This greatly increases the already great tedium of such calculations.

Computer Optimization

In order to minimize the pain of diode memory design, the task was subcontracted to a digital computer. The Seattle repeater group is extremely fortunate in having remote access to the University of Washington Computer Center's Burroughs B5500 computer, one of the nicest hardware/software systems ever put together. The resulting program in extended ALGOL accepts the desired code stream (in dots, dashes and blanks) as an input and performs a complete Boolean reduction for both +DOT and +BLANK diode decoders.

If the specified code stream is less than 32 characters (more than 32 characters are not allowed), then the computer assigns the necessary trailing blanks and performs



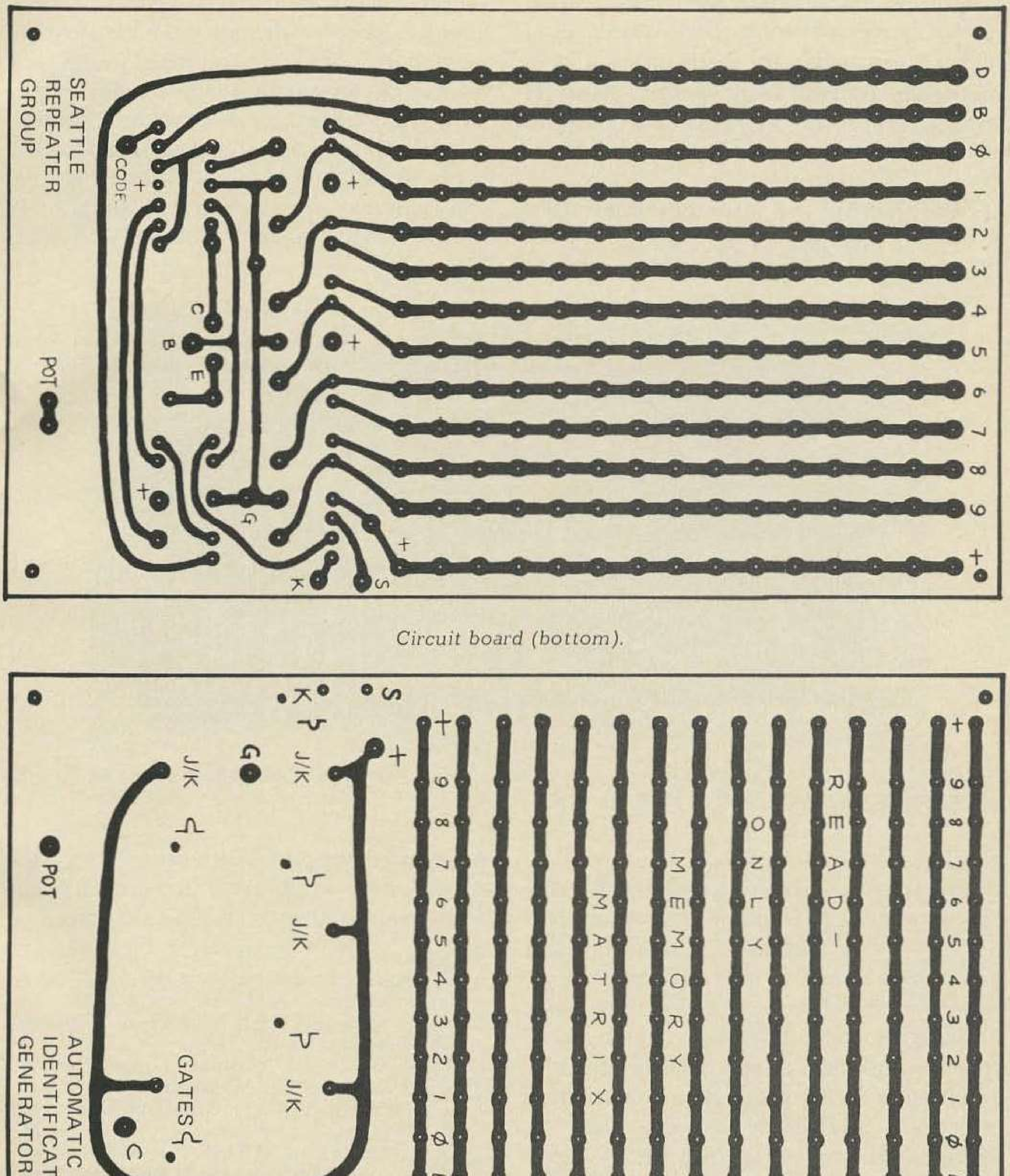
```
EXECUTION BEGINS•••



CODE STREAM [ -•• •   •-- --••• -•• -••• ••-•]
REQUIRES 48 DIODES AND 10 RESISTORS•

PLACE SILICON DIODES (S), GERMANIUM DIODES (G), AND
4•7K RESISTORS (R) IN THE FOLLOWING POSITIONS:

D B  0 1 2 3 4 5 6 7 8 9  +
S          G           G  R
S        G   G     G G    R
S      G   G   G   G      R
S      G   G G   G        R
S      G G     G G        R
S    G       G   G     G  R
  S  G     G         G    R
  S      G     G   G G    R
  S      G         G   G  R
  S    G G   G   G     G  R



EXECUTION COMPLETE•••
```

Fig. 9. Computer printout.

the required reductions, and repeats the process for each shift of the code stream until all the trailing blanks have become leading blanks. The code stream version having the smallest diode count is printed out along with diode and resistor counts and an actual map of the entire diode read-only memory matrix. Examples of the digital computer printout are shown in Fig. 9.

Construction

The four dual JK flip-flops are Motorola MC790P (or HEP 572) and the quad 2-input gate is a Motorola MC724P (or HEP 570). Virtually any PNP silicon tran-



Circuit board (bottom).

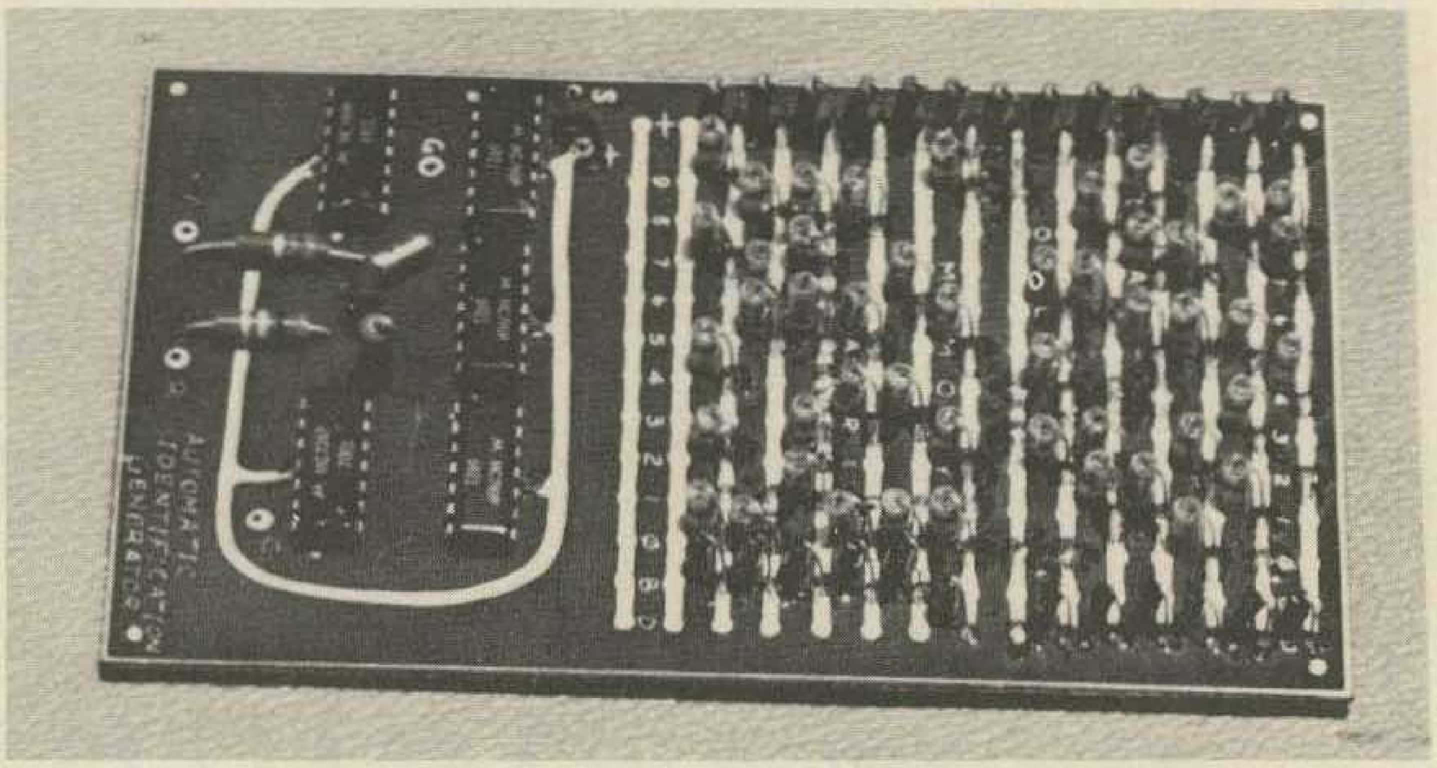
Circuit board (top).

Fig. 10. Full-size reproductions of PC board.

sistors will function in the clock circuit. A HEP 57 is a good choice. Both germanium and silicon diodes are used in the diode read-only memory. Germanium diodes are employed for the *and* function, since their low junction capacitance will not load the JK flip-flops in the program counter. Either silicon or germanium diodes may be used in the *or* function, with silicon signal diodes preferred since they result in a higher noise margin for the memory. Cheap diodes are available from various solid-state supply houses. Poly Paks features 50 silicon or germanium diodes for $1.

Both sides of 3 x 5 in. double-sided PC board are shown in Fig. 10. A one-sided board was used for the first few models, with a second 3 x 3 in. board used to complete the matrix connections. This "cordwood" construction is a pleasure to look at, but a nightmare to wire up. If a diode goes "west" on a cordwood style generator, it is best to throw it away, since unsoldering about 80 diodes and resistors and then getting things back together is even worse than the initial construction effort. A double-sided epoxy–glass PC board is recommended for the ID generator. An operational generator should be enclosed in a metal box with all leads bypassed for rf. Even VHF/UHF fields have the ability to drive RTL logic circuitry absolutely crazy. Before undertaking the construction of this ID generator, the builder should obtain as many of the referenced ID generator articles[1-2-3-4] as can be found and read them over carefully. The additional background material will amply repay the effort involved.

The Seattle repeater group can supply a moderate number of tinned epoxy–glass circuit boards for this ID generator. The board is not drilled, but assembly instructions and a computer optimized diode map for the circuit board is included. Be certain to specify the desired code stream, keeping in mind the absolute limit of 32 characters (dots, dashes, and blanks). Unit cost is $10, and they may be obtained from the Seattle repeater group, 18235 46th Pl. S., Seattle WA 98188.



Here's what the board looks like when the flatpacks and diode matrix are soldered in. The vertical placement of the diodes helps to keep the size down to this 3 x 5 in. circuit board.

Acknowledgement. The author expresses appreciation to K7EVO for art photography, and to K7MWC for the snapshot.

References

1. *A Digital Identification Generator,* Barry Todd K6ZCE, 73 Magazine, March 1967.
2. *IC Repeater Identifier,* Tom Woore WB6BFM, FM Magazine, January 1969.
3. *Automatic CW Identifier* John Connors W6AYZ, Ham Radio Magazine, April 1969.
4. *A Digital Morse Code Generator,* Jerry Hall K1PLP, QST, June 1969.
5. *Digital Logic Devices,* QST, July 1968.
6. *Micro-Logic for Non-logic Users,* Hank Olson W6GXN, 73 Magazine, June 1967.
7. *The Micro-Ultimatic,* Tom Pickering W1CFW, 73 Magazine, June 1966.
8. *Logical Design of Digital Computers,* Montgomery Phister. John Wiley & Sons, 1958.

. . . W7PUG ■