John W. Cartinhour, Jr. N5DY
36 Nottingham Lane
Pine Bluff AR 71603

Keyer Magic

—unravel the mystery of TTL circuits

In the fall of 1979, I took a course in digital electronics at the University of Arkansas at Little Rock, where I am studying for a degree in engineering technology. During the month-long Christmas break that followed, I decided to see if I could apply what I had learned. An iambic keyer seemed like an ideal project, since the circuit would be sufficiently complicated to be a genuine challenge. Besides, it so happened that my station did not already include a keyer.

You may wonder why anyone would go to the trouble of designing a keyer circuit from scratch, when it would be far easier to build a keyer based on a single IC such as the Curtis 8044. The answer is that designing a circuit based on common, general-purpose

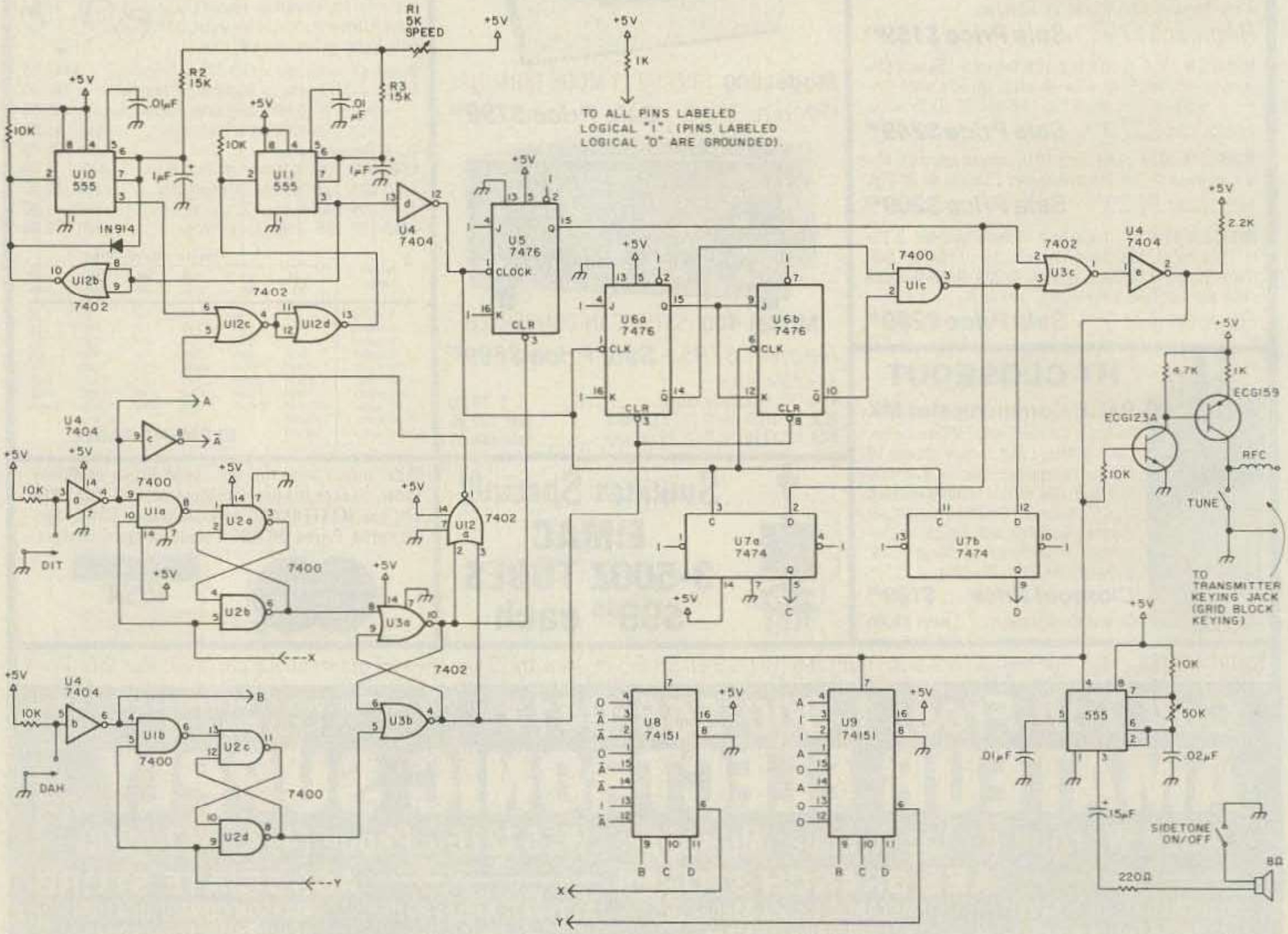

Fig. 1. Complete schematic diagram of the N5DY keyer. Pin numbers not shown are left unconnected.

digital ICs (TTL in this case) is a valuable educational experience for those who are just starting to learn something about digital electronics. I certainly don't claim that my circuit is the ultimate state-of-the-art design, but it works well. Readers who like to experiment with digital circuits may find it interesting.

Design Considerations

Let's begin by defining iambic operation in terms of what the circuit must do. At the end of each code character, a control circuit, or "decision" circuit, as I have chosen to call it, must check several circuit conditions and issue a command as follows:

1) If the dit paddle is on and the dah paddle is off, send a dit.

2) If the dit paddle is off and the dah paddle is on, send a dah.

3) If neither paddle is on, stop sending and reset parts of the circuit where necessary.

4) If both paddles are on, send a character opposite to the one just sent. For example, if the previous character sent was a dit, the keyer would then send a dah.

It is this fourth requirement, of course, that makes iambic keyer circuitry more complex than the circuitry required for a single-paddle keyer. The iambic keyer needs a memory circuit to record the last character sent, as well as control circuitry which can take command of the paddle at certain times.

In addition to the above requirements, there should be no initial startup delay between the time a paddle is depressed and the time a code character is started (other than the extremely small propagation delays in the ICs themselves), and code characters should be self-completing.

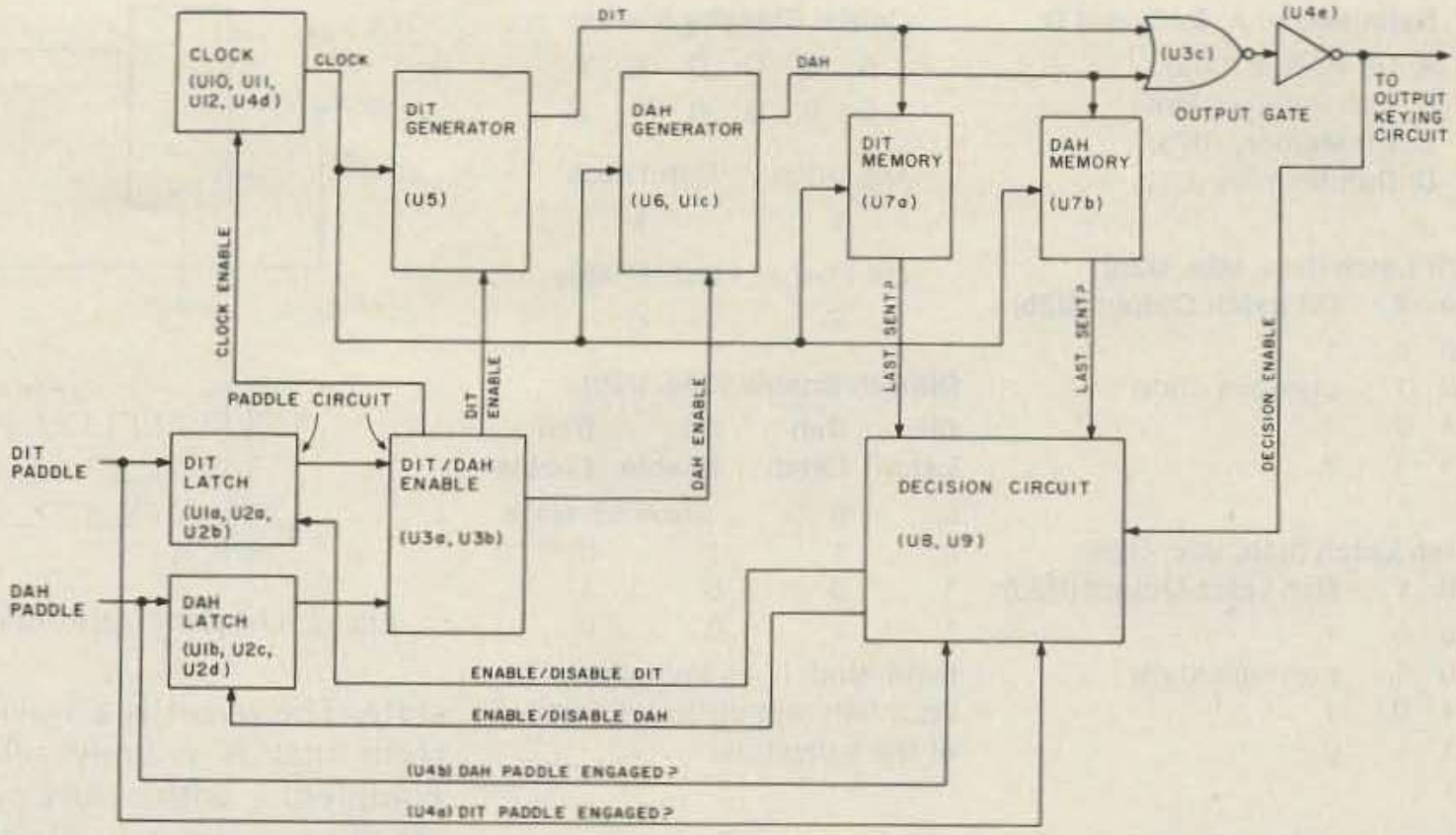

Fig. 2. Keyer block diagram.

The complete schematic diagram of the keyer is shown in Fig. 1. Fig. 2 shows the keyer in simplified block diagram form. (IC pins in the schematic diagram shown connected to "1" are actually connected to +5 volts, and pins shown connected to "0" are connected to ground. The labels "1" and "0" signify that these are logical "high" and "low" connections instead of ordinary power-supply connections. Some texts suggest that these logical "1s" should be connected to +5 V through a 1k resistor, with as many as 25 such connections sharing the same resistor; this is the procedure I have followed.)

Character Generation and Memory

There are many possible ways of generating dits and dahs. An article in the November, 1979, issue of *73* ("Son of Keycoder" by W4RNL) prompted me to start thinking about ways it could be done. In the end I chose a method that is perhaps the easiest to understand.

Let's first consider the dit generator and memory. Dits are formed by feeding a clock signal to a J-K flip-flop wired as a simple divide-by-two counter. When wired as shown in Fig. 3, the J-K flip-flop toggles whenever the clock signal goes from high to low. The output of the dit generator is monitored by a D flip-flop, which is activated by a positive-going clock pulse. Fig. 3 shows the circuit and the resulting waveforms.

Notice that the dit memory persists for half a clock cycle after each dit. The same relationship holds for a dah and the dah memory. The time between the end of a character and the subsequent clearing of its memory is the decision time, labeled t_d in Fig. 3. As will be explained in more detail later, the decision circuit will be activated whenever the keyer output goes low, such as at the end of one of the dits in Fig. 3. However, the final decision (that is, whether to send another character or stop sending) will be based on the conditions that prevail immediately before the character memory is cleared, or in other words, at the end of the decision time. The significance of this is that it gives the operator an extra half-space to get off the paddle before the keyer is committed to sending another character.

The dah generator is based on a synchronous divide-by-four counter. Two of the resulting waveforms are combined with a NAND gate to form the dah waveform, as shown in Fig. 4. (It should be noted here that in the original version of this circuit, the two J-K flip-flops were connected as a ripple counter instead of a synchronous counter, with the result that the keyer wouldn't work properly because a glitch in the dah waveform was activating the decision circuit at the wrong time. I don't intend to go into the details, but it is something that should be pointed out to those who may want to experiment with this circuit.)

The enable lines in Figs. 3 and 4 will enable the generator when high and disable it when low. The paddle circuit controls these enable lines. Whenever one of these enable lines goes high, the clock will also be enabled.

Referring to the complete schematic diagram, note that the outputs of the two generators are combined through U3c and U4e, which act as a simple OR gate.

The Clock Circuit

The clock circuit consists of two 555 monostable mul-

Definition of A, B, C, and D:

A: Dit Paddle (U4a)
B: Dah Paddle (U4b)
C: Dit Memory (U7a)
D: Dah Memory (U7b)

Initial Standby States:

A	B	C	D	X	Y
0	0	0	0	1	1

Dit Latch	Dah Latch
1	1

Dit Enable	Dah Enable
0	0

Dit Latch (U1a, U2a, U2b)

A	X	Dit Latch Output (U2b)
0	0	1
0	1	previous state
1	0	1
1	1	0

Dah Latch (U1b, U2c, U2d)

B	Y	Dah Latch Output (U2d)
0	0	1
0	1	previous state
1	0	1
1	1	0

Dit/Dah Enable (U3a, U3b)

Dit Latch	Dah Latch	Dit Enable	Dah Enable
0	0	previous state	
0	1	1	0
1	0	0	1
1	1	0	0

Note that it is impossible for both dah and dit to be enabled at the same time.

Decision Circuit (U8, U9)

Keyer Output	A	B	C	D	X	Y	BCD Address Corresponds to Input on Pin Number:
0	0	0	0	0	1	1	4
0	0	0	0	1	0	0	3
0	0	0	1	0	0	0	2
0	0	0	1	1	1	1	1
0	0	1	0	0	0	1	15
0	0	1	0	1	0	1	14
0	0	1	1	0	0	1	13
0	0	1	1	1	0	1	12
0	1	0	0	0	1	0	4
0	1	0	0	1	1	0	3
0	1	0	1	0	1	0	2
0	1	0	1	1	1	0	1
0	1	1	0	0	1	1	15
0	1	1	0	1	1	0	14
0	1	1	1	0	0	1	13
0	1	1	1	1	1	1	12
1	X	X	X	X	1	1	X

Note: X = don't care. Outputs from pin 6 are inverted data inputs.

Table 1. Truth tables.

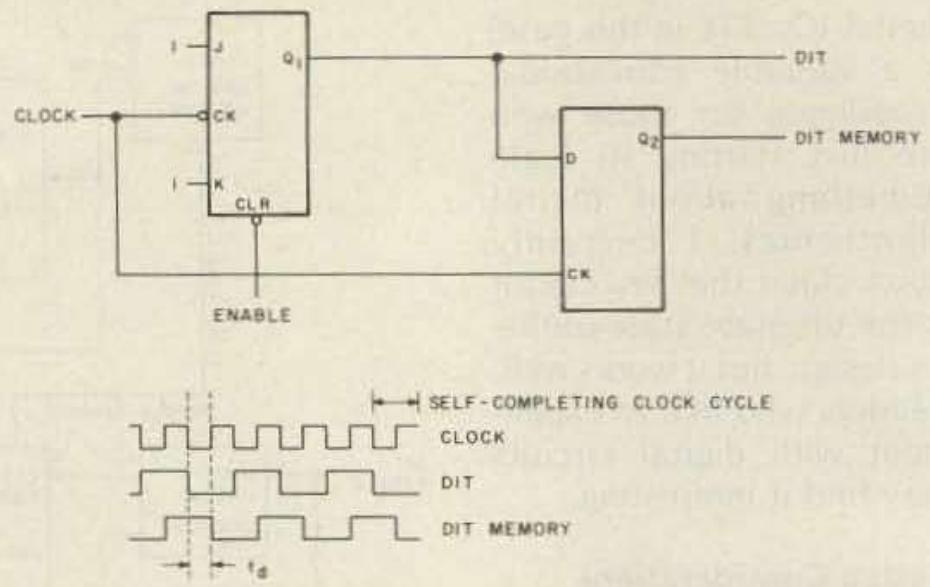

Fig. 3. Dit generator/memory circuit and waveforms.

tivibrators, wired back-to-back. It is basically an adaptation of a single-paddle, 555-timer-based keyer circuit described in *Solid State Design for the Radio Amateur* by Hayward and DeMaw, published by the ARRL. I experimented with several versions of a single 555 astable multivibrator clock circuit, but was not satisfied with any of them. With a single 555 clock circuit, the first clock pulse of a series of pulses will be longer than the pulses that follow. This is an unacceptable characteristic for my keyer circuit, since the clock is enabled and disabled along with the character generators. The twin 555 circuit described here eliminates the problem.

The complete clock circuit consists of U10, U11, U12, and U4d (see Fig. 1). Notice that the dit and dah enable lines are inputs to gate U12a. Initially, both 555 outputs (pin 3) are low, and both timing capacitors are discharged. A high signal on either enable line will trigger U11, which in turn triggers U10 through gate U12b. However, the diode in the circuit prevents the timing capacitor of U10 from starting to charge until after U11 has finished timing out. The clock circuit cannot be retriggered until after the output from U10 has returned to the low state. The result is a waveform that is entirely self-completing, with a duty cycle of approximately 50 percent with the component values shown in Fig. 1. At the end of one complete cycle, another cycle will be generated if an enable line is still high.

Readers who are unfamiliar with the 555 timer should consult *The 555 Timer Applications Sourcebook, with Experiments* by Howard M. Berlin.

The resistor and capacitor values shown will allow for code speed adjustments from approximately 15 to 35 words per minute. Readers who decide to experiment with different values should remember that the timing capacitors must be able to charge to two-thirds of the supply voltage in order to reset the timer. The safest bet would be to make R2 and R3 at least three times larger than the maximum value of R1.

The actual clock output is the inverted waveform of pin 3 of U11, which means that the clock waveform will begin with a high-to-low transition whenever an enable line goes high. This is necessary to meet the design requirement that there should be no initial startup delay between the time a paddle is depressed and the time a code character is started.

The fact that the clock waveform is self-completing means that two code characters cannot be separated by less than one space, regardless of how the paddles are manipulated. (See Figs. 3 and 4.)

Decision Circuit

The decision circuit is based on two 74151 1-of-8 data selectors (U8 and U9 in Fig. 1). Each chip has eight data-input pins and three address pins; three address pins allow for eight possible address combinations (000 to 111). There are two output pins: a data output and an inverted data output. In addition, there is a chip-enable pin (pin 7) which must be held low for normal operation.

One of the eight data inputs is selected to appear as the data output according to the address on the address pins. In Fig. 1, the data inputs to the 74151s are shown on the left side of each IC. They are arranged in order from top to bottom, starting with number 0 (for address 000) and ending with number 7 (for address 111). Note that the IC pin numbers do not correspond to this order. In this keyer circuit, the inverted data output (pin 6) is used. Pin 6 is driven high whenever pin 7 goes high, regardless of the condition of the address pins.

When the output of the keyer goes low, the decision circuit examines the following four circuit conditions, labeled A, B, C, and D:

A: Dit paddle. High if the paddle is on, low if off (from U4a).

B: Dah paddle. High if the paddle is on, low if off (from U4b).

C: Dit memory. High if the last character sent was a dit (from U7a).

D: Dah memory. High if the last character sent was a dah (from U7b).

There are sixteen possible combinations of A, B, C, and D. Four of these conditions (where C and D are both high) will not occur in normal circuit operation, but since they might turn up when power is first applied to the circuit, we must account for them to make sure the circuit doesn't get locked out when the power is applied.

Conditions B, C, and D are used to form the data-select address. Condition A, along with its complement, acts as a variable input to some of the data-input pins. There are two decision-circuit outputs, labeled X and Y, which determine what the keyer will do next. X and Y affect the state of the dit and dah latch circuits, respectively. The complete truth table for the decision circuit is included in Table 1.

The reader should refer to Don Lancaster's *TTL Cookbook* for an excellent discussion of how to design logic circuits with the 74151 IC.

Dit and Dah Latch and Enable Circuits (Paddle Circuit)

The state of the dit and dah paddles is sensed by separate dit and dah latch circuits, which also sense the state of X and Y, respectively. The dit latch consists of U1a, U2a, and U2b, with an output from pin 6 of U2b. The dah latch consists of U1b, U2c, and U2d, with an output from pin 8 of U2d (see Fig. 1). The truth table for these latches is included in Table 1.

The outputs of the dit and dah latches serve as inputs to the dit/dah enable circuit, which consists of U3a and U3b. This circuit controls the dit and dah generator enable lines; the truth table is included in Table 1. The most important feature of the dit/dah enable circuit is that it is impossible for both enable outputs to be high at the same time.

Basically, I designed the paddle circuit to allow the X and Y decision signals to override the paddles and force the dit (or dah) latch high at critical times. When the operator is off the paddle, the X and Y signals will reset the latches to their standby states (see Table 1), which, in turn, forces both dit and dah enable lines low.

An Example of Circuit Operation

Fig. 5 shows the various logic waveforms that result when the dit and dah paddles are manipulated to create the waveforms shown for A and B. The resulting code output is the letter "F". (It should be noted that this is by no means the only way to form the letter "F".) Hopefully, Fig. 5 is a picture worth a thousand words. Table 2 is a step-by-step analysis of the waveforms at various critical times. All of the action described in Fig. 5 and Table 2 is based on the complete truth table selection of Table 1. The notes in Table 3 on the critical times noted in Fig. 5 and Table 2 should be of some help.

Additional Circuit Notes

The keyer output circuit shown in Fig. 1 is based on the circuit shown in the *Radio Amateur's Handbook* (1980 edition) for the Accu-Keyer. It is intended for use with transmitters with grid-block keying. Many variations of this circuit have appeared in ham magazines. Take your pick! The choice of transistors was dictated by what was in my junk box, but I checked the voltage rating of the PNP output transistor before I used it. My transmitter has a grid-block voltage of −55 V. The ECG159 transistor is rated for a maximum collector-to-emitter voltage of 80 V, and a maximum collector-to-base voltage of 80 V. Your transmitter may have a grid-block voltage that is too large for this transistor to handle. The safest bet would be to use a 2N5401 or equivalent. The tune switch in the output circuit is actually my old straight key.

The sidetone oscillator is a standard 555 timer circuit which has appeared in ham magazines many times. Once again, component values were dictated by my junk box. Two improvements could be made: (1) Substitute a 1k potentiometer for the 220-Ohm resistor in series with the speaker to provide for volume control, and (2) change the 50k tone control to a 100k potentiometer to provide for a greater range of tone control.

I am using a Radio Shack variable dc power supply to power the keyer, which draws nearly 200 mA of current. If I decide to build a fixed 5-V power supply for the keyer, I will make it at least a 1-Amp supply so it can power additional proj-

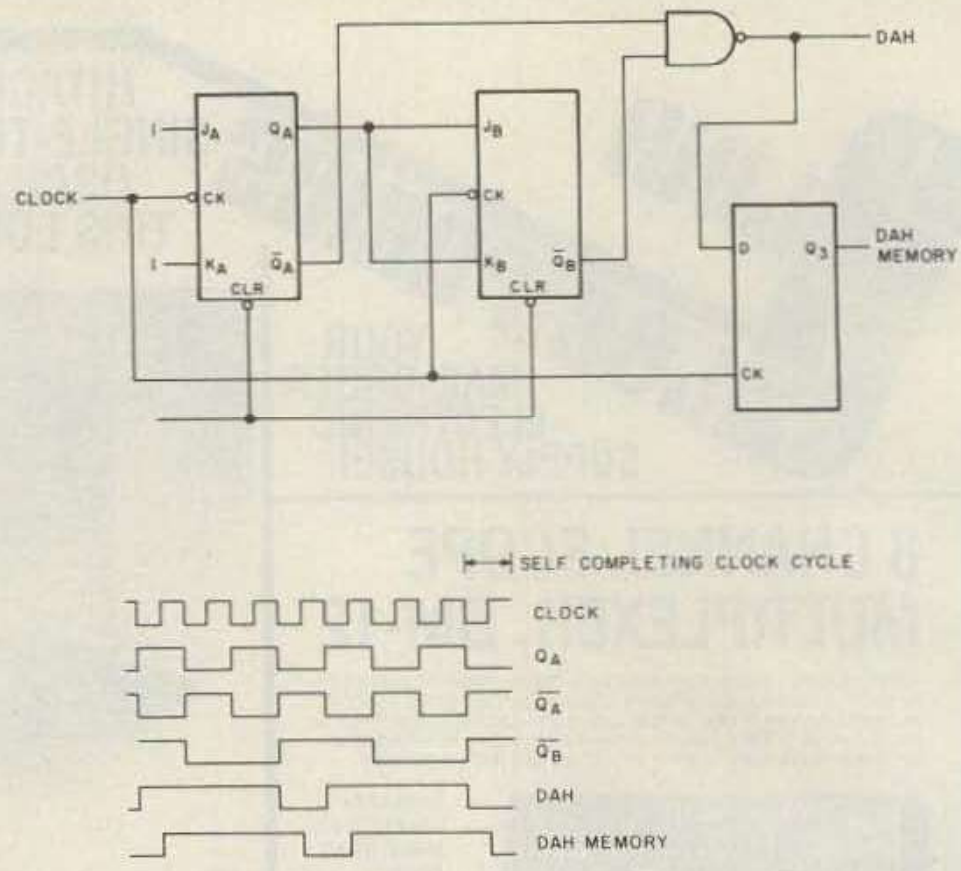

Fig. 4. Dah generator/memory circuit and waveforms.

	Circuit Conditions						Results			
Time	Keyer Output	A	B	C	D	X	Y	Dit Latch	Dah Latch	Dit Enable	Dah Enable
initial setup	0	0	0	0	0	1	1	1	1	0	0
t_1	0	1	0	0	0	1	0	0	1	1	0
immediately after t_1	1	1	0	0	0	1	1	0	1	1	0
t_2	0	1	0	1	0	1	0	0	1	1	0
t_3	0	1	0	0	0	1	0	0	1	1	0
t_4	1	1	0	0	0	1	1	0	1	1	0
t_5	1	1	1	1	0	1	1	0	0	1	0
t_6	0	1	1	1	0	0	1	1	0	0	1
t_7	0	1	1	0	0	1	1	0	0	0	1
t_8	0	1	1	0	1	1	0	0	1	1	0
t_9	0	1	1	0	0	1	1	0	0	1	0
t_{10}	1	1	0	1	0	1	1	0	0	1	0
t_{11}	0	1	0	1	0	1	0	0	1	1	0
t_{12}	0	0	0	1	0	0	0	1	1	0	0
t_{13}	0	0	0	0	0	1	1	1	1	0	0

Table 2. An aid in interpreting Fig. 5.

t_1: X and Y will always be forced high when the keyer output goes high.
t_2: End of first dit.
t_3: Clear dit memory.
t_4: See note for t_1
t_5: Operator presses dah paddle. B = 1 and Y = 1 will force the dah latch low, but since both latches are low, there will be no change in the enable lines.
t_6: Output goes low; decision circuit enabled. The decision circuits sees A = 1, B = 1, C = 1, and D = 0, resulting in X = 0 and Y = 1. This causes the dit latch to go high and the dah latch to go low. This, in turn, forces the dit enable low and the dah enable high.
t_7: Clear dit memory.
t_8: Analysis similar to that of t_6, except that this time the dit enable goes high and the dah enable goes low.
t_9: Clear dah memory.
t_{10}: Operator releases dah paddle.
t_{11}: Output goes low. Initially, the operator is still on the dit paddle, so the circuit prepares to send another dit. However, the operator has until t_{13} to get off both paddles to stop any further character generation.
t_{12}: Operator gets off the dit paddle in time to stop the keyer. The decision circuit now sees A = 0, B = 0, C = 1, and D = 0, which ends up forcing both enable lines low.
t_{13}: Clear dit memory. The circuit will now be reset to initial standby states.

Table 3. Notes on critical times.

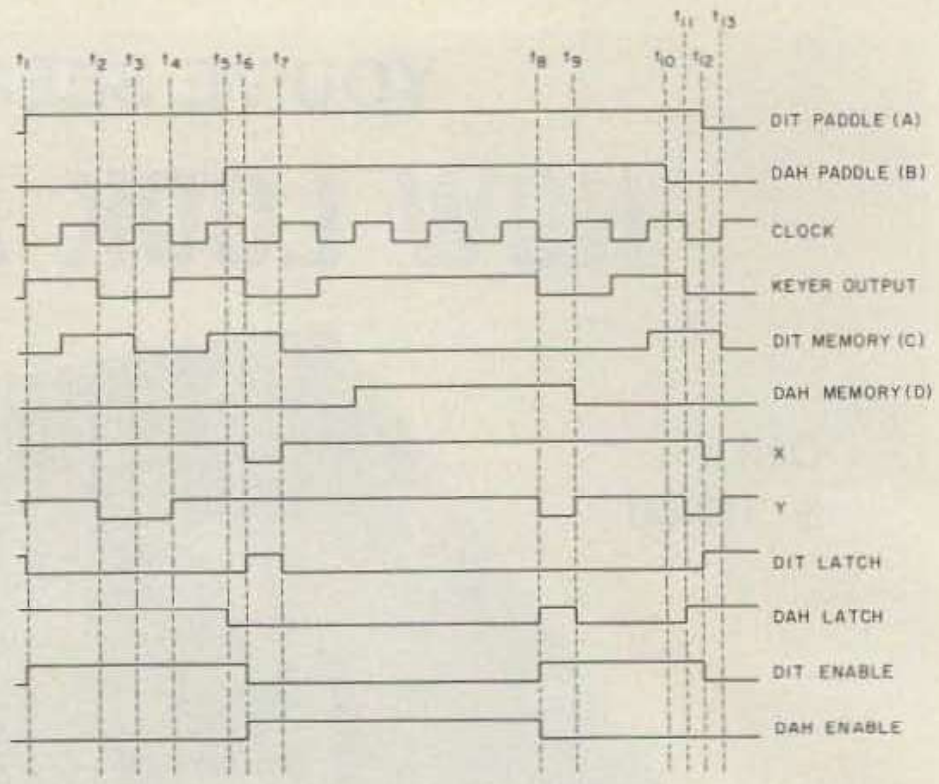

Fig. 5. An example of circuit operation.

ects. The power supply for a TTL circuit must be well-regulated, of course.

This is a good place to mention that the +5-V line in any TTL project should be bypassed to ground with small despiking capacitors distributed uniformly around the circuit board. About five 0.01-uF disc capacitors is about right for this project. Some experts recommend more stringent measures. The *TTL Cookbook* is a good source of information about this aspect of building TTL circuits.

I assembled most of the keyer on a perforated circuit board using wire-wrapping, but I designed a layout for an etched circuit board for the clock circuit.

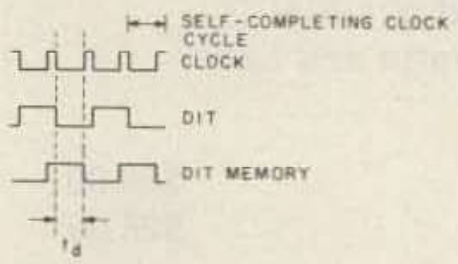

Fig. 6. Decreasing the duty cycle of the clock waveform will increase the decision time. Compare these waveforms to those of Fig. 3.

The best way to build a circuit like this one depends on your personal preference, of course. As of this writing, I have still not decided exactly what type of cabinet the keyer should be housed in. In the meantime, the assembled but exposed circuit has been set up on my operating table and connected to my Bencher paddle. I have used the keyer for many hours of off-the-air practice and am completely satisfied with its operation.

The final test was to connect the keyer to my transmitter to see if it would actually do the job. With the transmitter operating into a dummy load, the keyer worked flawlessly. With the transmitter operating into an antenna, however, the keyer would sometimes act up. Subsequent tests with a dip meter revealed that my rf-in-the-shack problem was worse than I had thought. The dip meter reacted strongly when brought near the coaxial transmission line, the rotator cable, equipment ac line cords, and the cable connecting the keyer to the transmitter.

Part of the solution to this problem was to add a small rf choke in series with the output from the keying transistor, as shown in Fig. 1. I also wrapped the power supply ac line cord around a ferrite rod from an old AM broadcast band receiver. These measures cured the problem as far as the keyer is concerned, and I am now able to use it on the air with no problems. Housing the keyer in a grounded metal cabinet might provide all the necessary shielding. Obviously, I will have to investigate my rf-in-the-shack problem further. The amount of rf-proofing you would need for this keyer circuit—or any other keyer circuit—would of course depend on how much rf is in your shack.

Final Comments

There are two books that you should definitely consider obtaining if you intend to do any experimenting with TTL ICs. The first one is the *TTL Cookbook* by Don Lancaster, which I have referred to in this article. The second one is *The TTL Data Book* by Texas Instruments. It is easy for a mistake to appear in a published circuit despite all efforts to avoid it, so anyone who intends to build such a circuit based on TTL ICs will find it helpful to have a copy of *The TTL Data Book* to check pin numbers and truth tables.

Experimenting with digital circuits such as this keyer can be a lot of fun, in addition to being educational. You ought to give it a try, if you haven't already. If this article encourages someone to get started, it has served its main purpose.

There is one more thing I would like to mention, which could probably come under the heading of second thoughts. A close look at the waveforms in Fig. 3 shows that the decision time could be increased or decreased by adjusting the duty cycle of the clock waveform. For example, Fig. 6 shows how a 25-percent duty cycle would increase t_d from half of a space to three-fourths of a space. The duty cycle can be reduced by increasing the RC time constant of U11, decreasing the RC time constant of U10, or a combination of both. Of course, these changes will also affect the speed range. I have not made any changes in my clock circuit, but since some keyer buffs may feel that a longer decision time is desirable, I thought I should point out how it could be accomplished. ■