

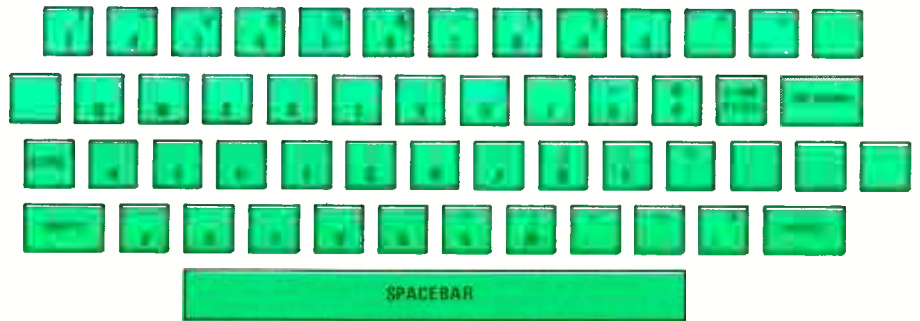
Keyboards: how to

Input/output. Ports. Video display terminals. Keyboards. What are these home-computer accessories? Each month, *Modern Electronics* explains in detail how one piece of hardware works and what it could do for your microprocessor system. This month: a close look at one way you can talk to your home computer.

by Peter A. Stark
Contributing Editor

The keyboard may well be the cheapest part of a small computer system. There isn't much that a keyboard can do all by itself, and yet it is an almost indispensable part of many general purpose computer systems.

The keyboard is simply a group of keys or pushbuttons, through which one enters numbers, letters, or punctuation marks into a computer. In a simple system where the computer is used only for numerical calculations, the keyboard might only consist of eight or ten keys used to enter the numbers. Where the computer is also used for processing words or names, the keyboard might include enough keys for all the letters of the alphabet, as well as all the digits and punctuation marks, and some special purpose keys as well. Keyboards having just number keys are *numeric* keyboards, while keyboards containing numbers, letters, and punctuation are called *alphanumeric*. The alphanumeric keyboard



Key layout in a typical computer keyboard. Some of the letter keys have additional symbols, as do all the number keys. Pressing the SHIFT key moves the command from the lower character to the upper one, as on a typewriter. The CTRL key enables the keyboard to generate special control codes.

is by far the more useful and interesting, so let us see what it consists of.

Most alphanumeric keyboards look very much like a typewriter keyboard. They have four rows of keys, with from 12 to 15 or more keys in each row. As in a standard typewriter, the top row of keys has the numbers, the second row starts

with QWERTY... in the traditional order, and so on. In addition, the keyboard may have special symbols above a few of the keys, and perhaps additional keys at both the left and right ends.

The purpose of an alphanumeric keyboard is to generate a unique binary number for every press of a key. This binary number is fed from the keyboard to a computer or other digital device over a set of six to eight wires, depending on the *code* used, with all of the bits available at the same time. This is called *parallel data transfer*.

ASCII Code

Although several data codes are used by various manufacturers, and may be available on inexpensive keyboards if they are used or surplus, the most common and most desirable code is called ASCII, which stands for the American Standard Code for Information Interchange. This is the code which is used by most of the larger computer systems, as well as almost all of the small ones.

ASCII is a seven-bit code where each of the seven bits can be either a binary 0 or a binary 1. With seven bits, there can be a total of 128 different combinations of bits, so ASCII has enough codes for 128 different characters, many more than there are on the typical keyboard. As a result, most keyboards only generate about 60 to 64 of the different codes;

The standard ASCII code

0	0	0	0	0	1	1	1	1
0	0	1	1	0	0	1	0	1
0	1	0	1	0	1	0	1	0

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	0	1	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	0	1	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	0	1	0	0
1	1	1	0	0	0	0
1	1	1	1	0	0	0
1	1	1	1	1	0	0

NUL	DLE	SP	0	@	P	^	p
SOH	DC1	!	1	A	Q	a	q
STX	DC2	"	2	B	R	b	r
ETX	DC3	#	3	C	S	c	s
EDT	DC4	\$	4	D	T	d	t
END	NAK	%	5	E	U	e	u
ACK	SYN	&	6	F	V	f	v
BEL	ETB	'	7	G	W	g	w
BS	CAN	(8	H	X	h	x
HT	EM)	9	I	Y	i	y
LF	SUB	*	:	J	Z	j	z
VT	ESC	+	;	K	[k	{
FF	FS	,	<	L	\	l	
CR	GS	-	=	M]	m	}
SO	RS	.	>	N	^	n	~
SI	US	/	?	O	_	o	DEL

Special control codes

Most common codes

Lower case letters

talk to computers

most of the others are used for special purposes in communications between computers and other digital terminals.

It is possible to list the specific ASCII code for each letter, number, and punctuation mark in a table such as

A	1000001
B	1000010
C	1000011
D	1000100

but this takes a lot of paper and is very inefficient. Instead, ASCII codes are usually shown in a square table as in Table 1. Each of the seven bits in the ASCII code is labelled, with the leftmost bit being b7 and the rightmost bit being b1. The left three bits are read from the top of the table, and the right four bits are read from the left side of the table. For example, the letter A is in the column labelled 100, so the first three bits of its code are 100. Going left from the A, we see that the right four bits are 0001, so the complete ASCII code for an A is 1000001.

The most important characters in this table are in the center four columns; those whose first three bits are 010, 011, 100, or 101. These four columns include all the upper case letters (the capital letters), the numbers from 0 through 9, and the common punctuation marks. Almost every keyboard will generate the codes for these four columns, with the possible exception of the punctuation marks just under the letter Z.

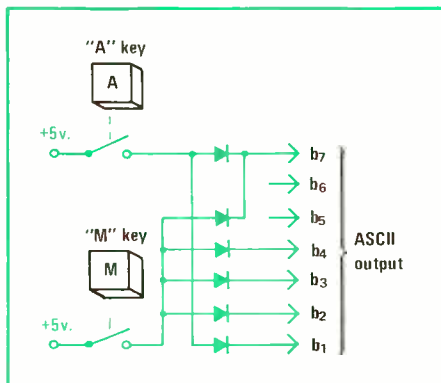
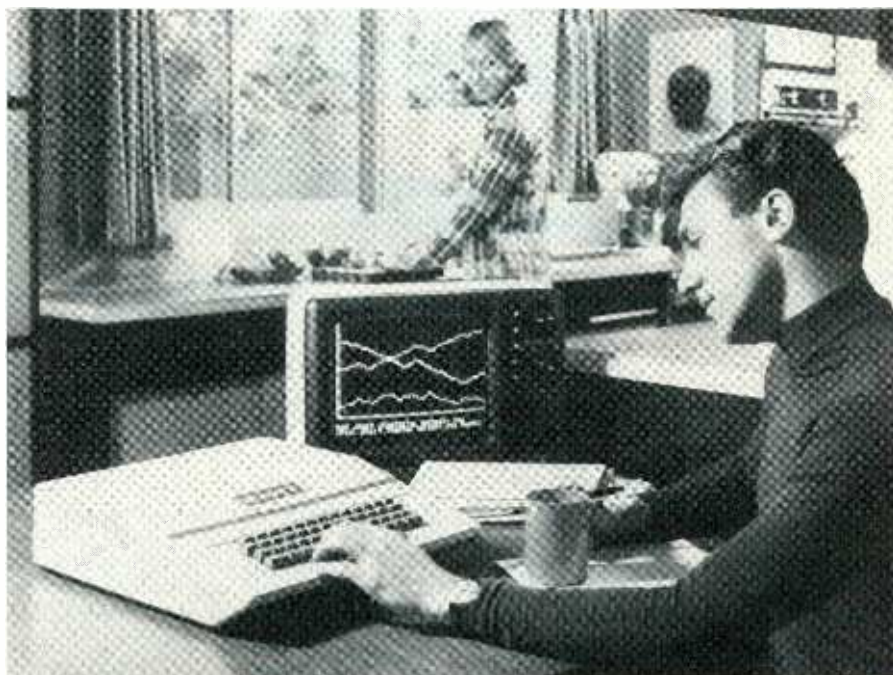


Figure 1: simplified diagram of a keyboard



Apple II is a complete home computer center for the hobbyist. Its capabilities go beyond calculators and video games and it uses the family color television set for startling graphic displays. You talk with the computer via its keyboard.

The right two columns, starting with 110 and 111, contain mostly the lower case letters a through z, as well as a few special punctuation marks. Most keyboards generate only the upper case letters and cannot generate any of the codes in these two columns; special upper/lower case keyboards are required if the codes for the lower case letters are needed, but sometimes these keyboards are difficult to use with some computer systems.

The left two columns do not represent letters or numbers at all, but instead are used for *control* characters which are mostly used for special high-speed communications between computers. The only two exceptions here are the code 0001010 for LF or *line feed*, and the code 0001101 for CR or *carriage return*. These two codes are virtually essential for every computer system and a key-

board which does not have them can be quite difficult to use.

Actually, the only difference between these left two columns and the fifth and sixth columns is that bit b7 is a 0 instead of a 1. Many keyboards have a *control* or CTRL key which reverses bit b7 whenever it is pushed. Hence pushing the CTRL at the same time as an M, for instance, would generate the code 0001101 instead of the normal M code of 1001101; this is the same as the carriage return. This means that such a keyboard can generate any of the control codes by a simple combination of two keys. This is useful for those codes which are seldom used, but using CTRL M at the end of each line would be very inconvenient; separate CR and LF keys are still almost a necessity to avoid errors.

Although the basic ASCII code is a 7-bit code, there are 6 and 8-bit varia-



Heathkit's model H9 cathode-ray tube (CRT) terminal includes a keyboard so you can send commands and requests for action into the computer and a video-display tube so it can send out its answers.

tions which are often used. As mentioned earlier, the center four columns of Table 1 are the most important; if you study the codes carefully, you will see that bits b7 and b6 are opposite for all of these columns. Hence, if we want to store the codes in a computer memory and need to save space, it is possible to discard bit b7 and only save b6; if we ever need b7 we need only invert b6. The resulting 6-bit code is often called *stripped ASCII*.

In many applications a bit b8 is added to the left of b7. This bit is used for error

checking and is called the *parity* bit. Depending on the remaining seven bits, the parity bit may be either a 0 or 1. Each different code has a different parity bit, and if a character is sent from one place to another and an error is made in one of the bits, the receiving system may detect an error if the parity bit is different from what it should be. The result is then an 8-bit code. (In some systems the parity bit may not be used, or it may be generated by the keyboard but not checked by the rest of the system. In that case it might always be 0 or 1, since it is not used.) The parity bit is especially useful when digital data must travel over great distances or when it is recorded on tape or disk and likely to contain errors.

A Simple Keyboard

The function of every key on the keyboard is to generate the correct ASCII code each time it is pressed. Figure 1 shows the basic principle used in some simple keyboards to provide the correct digital output, to keep the diagram simple, only the A and M keys are shown. In reality, the diagram might have as many as 50 or 60 keys.

As shown, the ASCII output is on the right side on seven wires. Each of the keys is connected to a unique combination of wires, so that when the key is pressed a positive voltage, usually +5 volts, is connected to generate the correct code. In the case of the A key, when the key is pressed +5 volts is connected

to the b7 and b1 outputs, while no connection is made to the other outputs. This results in an output of 1000001, assuming that +5 volts means a binary 1 and no voltage means a binary 0. In the same way, pressing the M key generates a code of 1001110. Diodes must be added as shown to isolate the keys from each other; if the diodes were missing sneak current paths would exist so that whenever any key was pressed, *all* the outputs would go to +5 volts.

On the average, each key requires about 4 diodes. Thus a 50-key keyboard needs about 200 diodes. This number of diodes is difficult to clearly show on a diagram, and it also creates problems in physical placement and connection within the keyboard itself. As a result, an arrangement called a *diode matrix* is often used, both on diagrams and in the actual physical assembly itself. Figure 2 shows the revised diagram in the form of a diode matrix.

Each of the key switches is connected to a vertical wire, while each of the outputs is connected to a horizontal wire. In an actual keyboard, a double-sided printed circuit board is often used, with the vertical wires on one side of the board and the horizontal wires on the other side. There is no connection between the two sets of wires, except that the diodes are installed at the crossovers between the key switch wires and those outputs which are supposed to be a 1 for the ASCII code. Keyboards using this principle are often available as surplus, and are popular with computer hobbyists since it is simple to move diodes

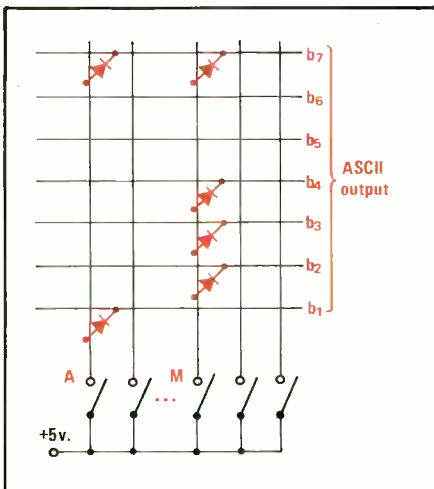


Figure 2: diode matrix from a keyboard

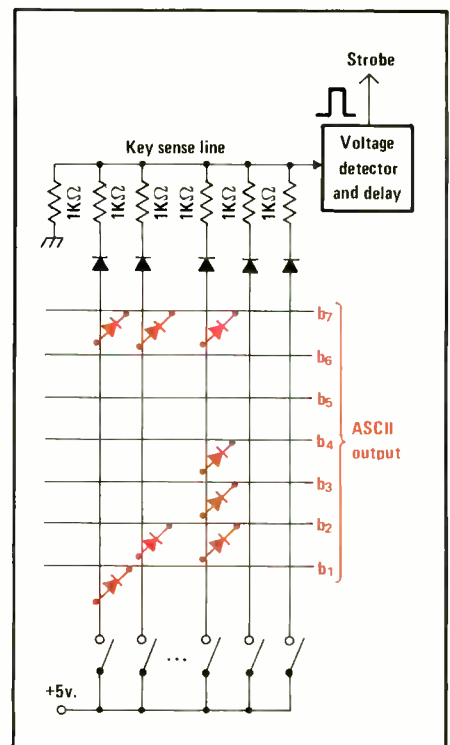


Figure 3: rollover and strobe circuit added to diode matrix

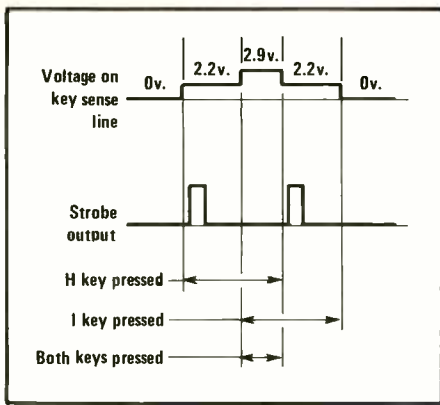


Figure 4: operation of the rollover and strobe circuit

from place to place to adapt the keyboard to whatever code they desire.

Although Figure 2 could be used to wire a keyboard with simple switches and inexpensive diodes, it is far from complete. We must add more circuitry to make it useful. First, we must make provision for a SHIFT key which would select the second symbol present on some of the keytops. For instance, the 1 key also has the ! symbol; when the SHIFT key is pressed it should generate the ! code of 0100001, while if the SHIFT key is not pressed it should generate the 1 code of 0110001. Clearly, the SHIFT key must change bit b5, in this case from a 1 to a 0. But other times, it must change b5 from 0 to a 1, since the N key must generate the 'up-arrow' symbol when SHIFT is pressed. So the SHIFT key must invert the b5 bit whenever it is pressed.

Next, we must add a CTRL key which will change bit b7. This circuitry could be similar to that used with the SHIFT key. If the keyboard is to generate both upper case and lower case letters, then still other changes are needed.

Still another circuit which should be added is one to generate a *strobe* or *key pressed* signal. This is a signal which informs the computer or terminal to which the keyboard is connected that a key is pressed. This is needed since otherwise the system would have no way of knowing whether a key is being pressed or not, except by continuously monitoring the ASCII code output to see whether it has changed or not. The strobe is usually a fast positive pulse (a voltage which goes positive and quickly returns to zero).

The strobe is closely related to another desirable function, that of *debouncing*. Since the key switches used in the keyboard are mechanical and involve sliding metal contacts, they do not open and close instantaneously, especially when the typist presses and releases the keys slowly. As the switches open and close, they slide or bounce briefly, so that the connection is intermittent both at the beginning of the character and at the end. Computers are generally so fast,

that during the short time that the contacts seem to open-close-open-close rapidly, it appears as though the key has been pressed several times rather than just once. As a result, a word may appear like tthhhiissss to the computer. The strobe circuit is designed so that it will wait a few thousandths of a second to give the switch contact time to settle before generating the strobe pulse.

Since the strobe circuitry has to detect the closing of a switch (*any* switch) and provide a delay, it is often improved so that it will respond only when *one* switch is closed, not more than one. This provides a function called *2-key rollover* which is extremely important to fast typists.

A fast typist often presses one key before fully releasing the previous key. For instance, in typing the word HI he might press the I key before releasing the H key. On an ordinary typewriter this

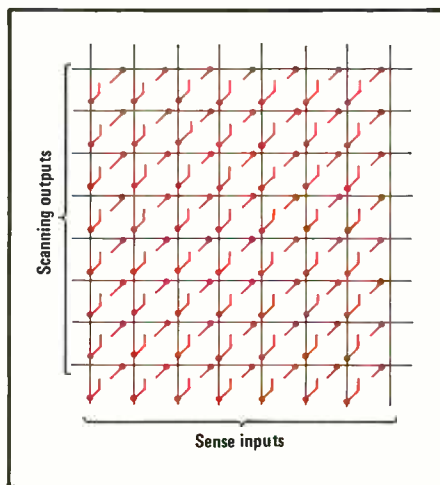


Figure 5: a switch matrix

creates no problems as long as the two keys are not pressed at almost the same time, but with a computer keyboard the wrong code might be generated when both keys are down. It is the function of the strobe output to separate the two codes of the two keys and provide no

strobe pulse when both are down. Figure 3 shows how the strobe circuits detect how many keys are down.

Each key connects, through a diode and resistor, to a key sense line, which is connected to ground through another resistor. When no key is pressed, the voltage on this line is near zero volts. When one key is pressed, the 5 volts on the vertical wire divides across the diode and the 1K resistors so that the voltage on the key sense line goes to about 2.2 volts. But if more than one key is pressed, the voltage on this line goes to 2.9 volts or even higher. A voltage detector and delay circuit is carefully adjusted so that it operates only when the voltage on this line is close to 2.2 volts, indicating that one and only one key is pressed.

Figure 4 shows what happens when the word HI is typed with some overlap of the two keys. Initially, the voltage on the key sense line is 0 volts. When the H key goes down this voltage rises to 2.2 volts and a short time later, after the debouncing interval, the strobe circuit puts out a short pulse. Then, at the time the I key is also pressed, the voltage on the key sense line goes up to about 2.9 volts and remains there until the H key is released. As soon as the strobe circuit sees the voltage back at 2.2 volts, it waits for a short time and puts out another strobe pulse.

The diode matrix keyboard, though simple to design and understand, has some disadvantages when it comes to mass production. Its use of several hundred diodes as well as a variety of other components for rollover and strobing makes it more expensive to manufacture than if it used a more limited number of integrated circuits. The ultimate goal would be to build the keyboard with just several dozen switches and one integrated circuit. This requires a completely different approach.

The typical keyboard has close to 50 or 60 keys. If each of these had to be connected to an integrated circuit, this would require some rather large inte-



Unusual, attractive styling of the computer keyboard, foreground, and video-display terminal are by The Digital Group.

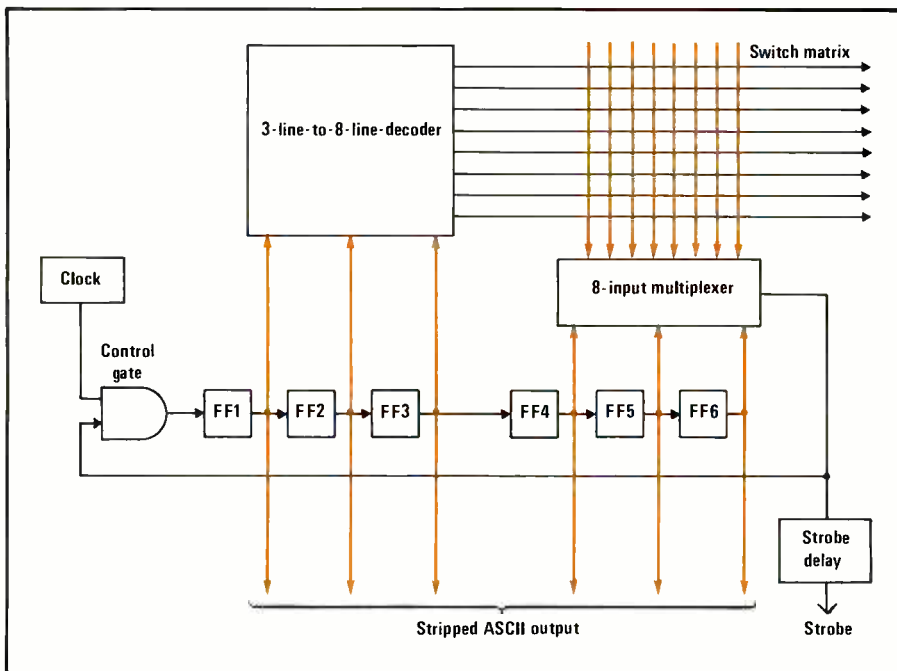


Figure 6: actual circuitry of a simple scanning keyboard, without SHIFT or CTRL keys

grated circuit packages with many pins. If the number of connections is to be reduced, a way has to be found so that all the key switches share as few wires as possible. Figure 5 shows how it is possible to connect as many as 64 switches with just 16 wires, by simply building a switch matrix. The 16 wires are divided into 8 vertical and 8 horizontal wires, with a key switch at each intersection. Since each wire is shared by eight switches, a smaller number of wires is needed. (This particular matrix is a square one of 8 by 8 wires, but it could just as well be 8 by 9, or even 8 by 11 if more switches were needed.)

Although the switch wiring is simple, the circuitry to go with it must be somewhat more complex so it can recognize which switch is closed and generate the correct ASCII code simply by monitoring which vertical wire connects to which horizontal wire. If several key switches are closed at the same time, this circuitry must also be able to find which closed first and which opened last so it can provide for rollover. This is done by scanning, the same principle which is also used to monitor the keyboards in handheld calculators.

In short, the scanning circuits put short pulses on the horizontal wires, one after another, in a continuous sequence. At the same time, the vertical wires are connected to a series of sense inputs which continuously look for these same pulses to appear on them. Whenever a pulse is received on the sense inputs, the circuitry takes a quick look to see which of the scanning outputs has a pulse on it at the same instant of time, and determines which switch is closed from that information.

Figure 6 shows how such a scanning keyboard can be wired. A clock oscillator

generates a continuous series of high frequency pulses which pass through a control gate to a series of six flip-flops FF1 through FF6. The control gate acts essentially as an electronic switch which normally lets the clock pulses pass through to the flip-flops.

The six flip-flops are connected in a counter circuit which counts the pulses coming from the clock and generates six outputs; these outputs are binary numbers which indicate how many pulses have been counted:

000000
000001
000010
000011
000100
etc.

Each time a pulse arrives from the clock, the binary output from the six counter flip-flops changes. After 64 pulses, the

counter output reaches the output of 111111, and then returns back to 000000. Hence, the six flip-flops are continuously cycling from 000000 up to the maximum count, and then back again to 000000. Since the ASCII output is taken directly from the counter outputs, it keeps continuously changing. However, since no key is pressed the strobe output is absent, and so the external circuitry should not even be monitoring the keyboard output. Thus the constantly varying codes should not matter.

As all this is happening, the outputs of the flip-flops are also going to a decoder integrated circuit and a multiplexer integrated circuit. The decoder receives the signals from the first three flip-flops, and sends a pulse to one of the horizontal key matrix wires at a time; which one depends on the output from the three flip-flops at that instant. In other words, it continuously scans the eight horizontal wires, sending a pulse to one at a time. These scanning pulses occur very rapidly.

At the same time, the outputs from the last three flip-flops go to a multiplexer. This circuit can be compared to an eight-position switch which has eight inputs and one output. At any given time, one and only one of the inputs is connected to the output; which output depends on the state of the three flip-flops connected to it.

If no key switch is pressed, then the multiplexer will never receive any inputs, and hence will provide no output. Then nothing will happen.

But when a key switch is closed, then it is possible for the multiplexer to detect a signal from the decoder and provide an output . . . but only if all the flip-flops are at the right state so that the decoder is sending a pulse to the closed switch at the exact instant that the multiplexer is looking for it on the right vertical line. Since the flip-flops in the counter are continuously cycling through all possi-

Please turn to page 89



Keyboard is on the front of the Attache, a complete desk-top computer by Pertec Computer

Computer keyboards

continued from page 64

ble states this will happen anywhere up to 64 clock pulses from the time the switch closed. But if the clock pulses are arriving rapidly, this may still be a short time.

In any case, after some delay the flip-flops will reach some unique combination of states, and the multiplexer will suddenly provide an output. This is sent back to the clock control gate, which stops further clock pulses from arriving at the counter. In other words, the counter flip-flops suddenly freeze in whatever state they are in, and the ASCII output stops changing. A short time later the strobe delay circuit provides a strobe pulse.


In order to make sure we get the correct ASCII code for the correct key, we must make sure we place the key at the right intersection of the correct two wires. This is easy to do if a printed circuit board is designed to mount all the keys and provide the connections as well.

The delay in the strobe line provides for debouncing, and the circuit itself provides for 2-key rollover. Once one key is depressed and sensed, the clock stops and closing further keys does nothing as long as the first remains closed. Once the first key is released, the circuit automatically resumes scanning and will, in a few clock pulses, find the second key closure. If, however, more

than two keys are closed than the additional keys will be ignored; sensing more than two keys requires a more complex circuit having *N*-key rollover.

The circuit of Figure 6 can be wired with just six or seven inexpensive integrated circuits, although a few more are needed to add the functions of a SHIFT key and a CTRL key. The circuit is sufficiently useful, though, that several large-scale-integration circuits have been designed just for this purpose. One such IC is the 2376 shown in Figure 7.

The 2376 is designed for up to 88 keys, and so it has an 8 by 11 line switch matrix, shown at the top. A simple resistor-capacitor circuit at the left provides the timing for the clock, and another at the right provides the delay for debouncing. Two switches for SHIFT and CTRL connect directly to the IC, and a variety of outputs provide both ASCII codes and strobe pulses. This particular integrated circuit can generate both upper-case-only as well as upper-and-lower-case output codes, depending on the setting of a single switch. It also provides a parity output on pin 7, if desired.

Although computer keyboards are fairly simple in operation and construction, they form the backbone of many computer systems. Any system which operates on any kind of alphabetic information usually starts somewhere with a keyboard which is used to enter either data or programs. 

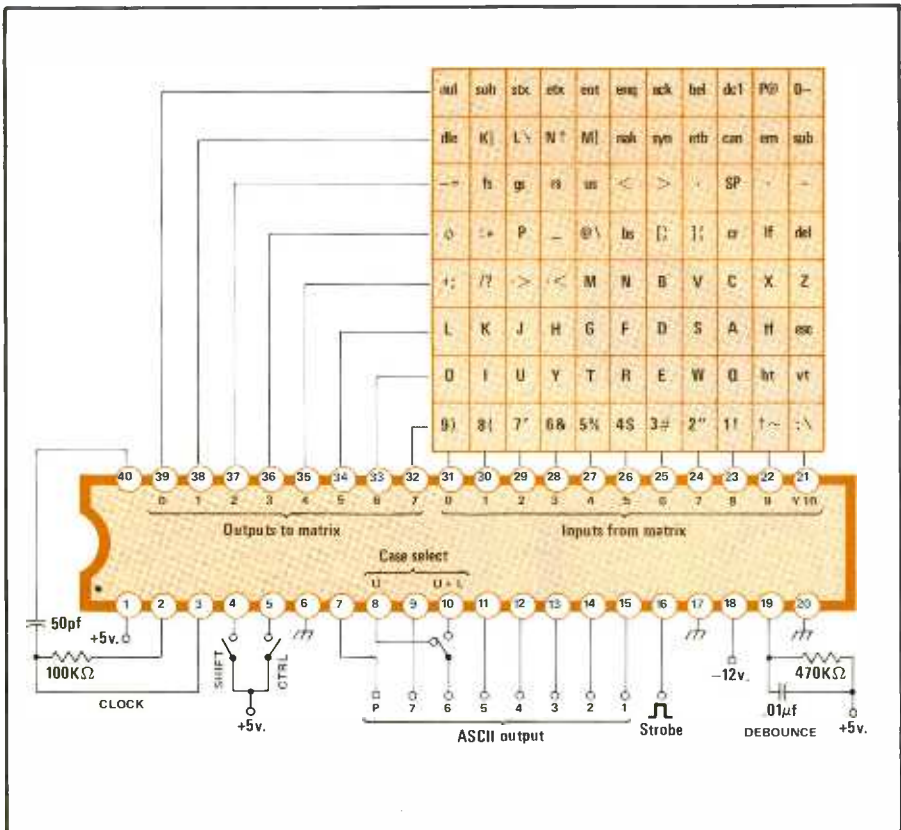


Figure 7: a 2376 integrated circuit provides most of the functions needed by the keyboard, including upper/lower case shifting, CTRL key, debouncing, generation of strobe, and two-key rollover