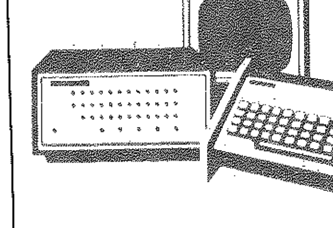


Computer Bits

By Hal Chamberlin

KEYBOARDS

WHAT IS the most important part of a microcomputer system? Is it the microprocessor chip, memory boards, mass storage system, or display interface? If it is the part that is most used by the owner, then the answer is probably the keyboard. As necessary as these other parts are to the functioning of the system, most users spend far more time with the keyboard than any other peripheral. While many informed sources foresee the day when the primary man-machine interface link will be through speech, keyboards will be with us for quite a while. It is amazing how many manufacturers overlook the importance of the keyboard to a successful product and how many prospective users fail to intelligently evaluate a system's keyboard.

Human Factors. When speaking of keyboards in conjunction with microcomputers, we mean so-called *alphanumeric* keyboards; that is, those that have 10 digits, 26 English letters, and an adequate complement of punctuation marks. Today, the feature that most distinguishes a microcomputer from the multitude of less expensive programmable calculators is the ability to handle *textual* information as easily as numerical information. Any system that offers only a numeric keypad is definitely limited in its potential beyond learning machine language and experimenting with short programs.

Assuming an alphanumeric keyboard is needed, how many and what kind of characters should it have? Since all personal computers use the ASCII character code which in its smallest form defines 10 digits, 26 letters, and 27 symbols, it follows that a minimum keyboard should have all of these. A more comprehensive keyboard would have an additional 26 lower-case letters and 5 symbols and thus could be called a *full ASCII* keyboard. While many of the earlier packaged computers had just a minimum keyboard, the availability of lower-case letters is required for the computer to realize its text editing and processing potential—a major application area. In fact, many people have either sold their older systems or modified them to handle lower-case letters. Fortunately, manufacturers are starting to get the message so most personal computer keyboards and displays in the future will be capable of handling the full 94-character ASCII set.

Another keyboard variable is the arrangement of the character keys. There is little question that the 26 letters should be arranged in the traditional typewriter style. While exhaustive human-factor tests have shown this layout to be less than optimum with respect to typing speed and accuracy (the relatively obscure "Dvorak" layout, which is also scrambled, holds that distinction), it is practically universal on today's typewriters and is taught in typing classes. Any system with a sequential "ABCDEFG" keyboard is severely limited in utility when the user passes the "get acquainted" stage. Even if the user cannot touch-type, it is valuable to have a keyboard that uses a standard layout because then it will be easier to adapt if he or she should ever "trade up" to a more powerful system. There are actually touch-typing training programs on the market in which the computer itself generates the drill text (on the screen) and measures typing speed and accuracy automatically.

There is very little agreement however on where the digits and symbols should be placed on the keyboard. Digits are usually placed in a horizontal row above the letter keys (as on a conventional typewriter) but this is not very efficient if a lot of numerical data is to be typed in. A method used by many data processing machines is to put the digits in a 3-by-3 matrix as the *shifted* version of the characters U I O J K L M , . (with zero in its normal location), which can then be typed by skilled adding machine operators. Fortunately no personal computer uses this approach, but many have a "numeric keypad" with the 3-by-3 digit arrangement to the right and separate from the main keyboard. The ideal keyboard would probably have both with provisions for the program to determine which group of digits were to be used. This would allow nonbusiness programs to use the extra keys for functions such as cursor movement or text editing commands.

Symbol placement generally conforms either to the IBM or ASCII arrangement. While the latter is preferred because of its association with the standard character set used by computers, many systems aimed primarily at text-editing applications use the IBM arrangement because of its greater similarity to typewriter keyboards. The two arrangements are easily distinguished. IBM has the parentheses on the 9 and 0 keys while ASCII has them on the 8 and 9 keys.

Mechanics. Just having an acceptable arrangement of keys doesn't necessarily make an acceptable keyboard; mechanics are important too. Probably the most important mechanical factor is the size and spacing of the keys. In this respect, the standards are quite clear: the keys are to be placed on $\frac{3}{4}$-inch centers both horizontally and vertically and are to be staggered by $\frac{3}{8}$ inch for the top two and bottom two rows and $\frac{3}{16}$ inch for the middle two rows. Some manufacturers have attempted to ignore this standard and provide a keyboard with smaller-than-normal keys packed together with as little as $\frac{1}{2}$-inch spacing. The results have been tens of thousands of lost sales and, for at least one vendor, a new line of machines with standard-size keyboards.

Another area of considerable variation among available systems is keybutton travel. On standard typewriters, this varies from nearly an inch on manual to about $\frac{1}{8}$-inch on electric machines. The electronic keyboard industry, however, is in general agreement the optimum travel should be about 0.15 inch for the feel of good positive action. Recently, there have been some systems with keys that travel little if any. In this case there are really no keys at all; the letters and symbols are merely printed on a plastic sheet stretched over a supporting frame with pressure-sensitive contacts molded in. When typing, the effect is like drumming your fingers on a tabletop. While this makes little difference for the occasional hunt-and-peck typing needed for preprogrammed game machines, it would probably drive a touch-typist crazy. Although such keyboards are very inexpensive and somewhat more immune to coffee spills, a standard travel keyboard is certainly worth the difference when many hours are to be spent using the system.

An additional mechanical consideration is the degree of feedback the keyboard gives the operator when a key is struck. As previously mentioned, full-travel keys give some feedback merely by their substantial displacement when pressed. There have been numerous attempts on professional keyboards to provide tactile feel with the key itself making a click and then suddenly releasing backpressure at the instant the computer gets the character code. While the basic idea is sound, very few keyboard manufacturers have implemented it correctly (there is a confusing secondary click and pressure increase when the key is released). As a result, acceptance has been minimal. On personal computers it would be a simple matter to provide an audible click or short beep from a built-in speaker when a keystroke is entered—but very few systems actually do. Audible feedback not only gives confidence that the system is responding (or an instant indication that it is not), it also improves typing rhythm and thus speed and accuracy. If audible feedback is provided, however, there should be a way to turn it off when silent operation is desired.

One final human-factor feature that is nice to have on a keyboard is automatic repeat. It is often necessary to enter the same character continuously—as when moving the cursor for editing. Pounding the same key repeatedly not only shortens its life, but is annoying and slow. With automatic repeat, a key continuously held down repeats its code in rapid succession until the key is released. There is usually a time delay of about $\frac{1}{2}$ second before repetition starts to avoid unintended repeats, and the repeat rate is about 15 per second. Although this feature is very easy to implement (and nearly free) when the keyboard is scanned by software, few if any packaged systems offer it at present.

Interfaces. Several general methods of keyboard interfacing are in common use, even on packaged systems in which the keyboard is integral to the system. Modular computers, such as S-100 machines, often use the keyboard of an external display terminal. From the computer's point of view, characters come in at random times on a serial interface encoded and ready to go. A large number of modular machines use a video display interface plugged directly into the bus and a separate keyboard interfaced to a parallel port. This approach saves the expense of a separate terminal and gives added flexibility.

In either case, the computer program must be "looking" at the keyboard when the keys are struck, otherwise the input characters may be lost. A system with this difficulty can be greatly helped by having audible feedback to tell the operator instantly when a keystroke is or is not accepted. A better way to solve the problem is to use interrupts with the keyboard. With interrupts, every keystroke interrupts the running program and informs it that a key has been struck. If the program is not yet ready for an input, the character is read and placed in a memory buffer. When the program is ready, the buffer is examined first and characters are taken from it until the buffer is empty. Then the program waits for a keyboard input. This is a very valuable feature for interactive programs written in BASIC, in which it is easy for the operator to get ahead of the program.

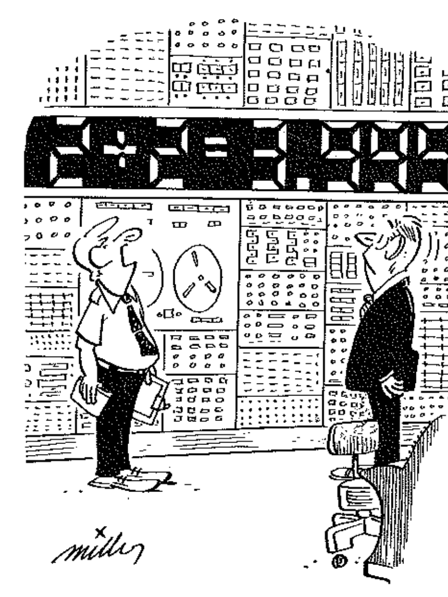
While the keyboard logic does most of the work associated with scanning the keys and forming the character code on modular systems, many packaged systems use the microprocessor itself to do this task at much lower cost. In these systems, the individual keyswitches are wired into a matrix, usually with 8 rows and 8 columns and a switch at each crosspoint. The 8 column wires are then connected to an internal 8-bit output port and the 8 row wires to an 8-bit input port. Whenever a key is depressed, an output is connected to an input with the combination different for each key. With proper construction, it is possible for the program to identify which keyswitch is closed and then look up its code in a memory table for use by the rest of the software.

Typically the keyboard-read program, which is usually part of ROM software, scans the keyboard, which means it "looks" at each key in sequence and registers a character when it locates a depressed key. Additional software tests are required to prevent the same character from being generated each time the software scans and to negate the effects of keyswitch contact bounce.

There are two common ways of solving the former problem, each giving a different typing characteristic to the keyboard. One of these is to stop scanning the keyboard when a depressed key is found, output the code, and then wait until the key is released. At that point, scanning is resumed. This system is called "two-key rollover" because it produces the proper sequence of characters (that is, the same sequence that was typed) for a maximum of two keys being pressed simultaneously. The second method uses continuous scanning and keeps a copy of the state of each key in memory. A keystroke is registered only when a key that was not previously down (as determined by looking at the copy in memory) is seen down for the first time. Thus, the keyboard responds only to up-to-down movement of a key and then completely ignores it. This is called "N-key rollover" because it produces the proper character sequence regardless of the number of keys already down. Having more than one key down simultaneously is a common occurrence, particularly with fast typing, so it is important that the keyboard, whether it be hardware or software scanned, handle the situation in a reasonable manner.

Keyswitch contact bounce is a serious problem. All mechanical contacts fail to make and break instantaneously and may vibrate (produce multi-contacting) for a few milliseconds before a solid contact is assured. With the speed of modern microprocessors, a keyboard scanning program could interpret this bouncing as several keystrokes and produce multiple repetitions of the same character. From a software viewpoint, the problem can be solved by having the scanning software verify the making or breaking of a contact by testing it for about 10 milliseconds and requiring that every test produce the same results before concluding that a make or break actually took place. Fortunately, systems with bounce problems can be cured by using a "keyboard fix" program available from the manufacturer or on the open market.

Interrupt operation of a keyboard that is software-scanned is more difficult than if it is hardware-scanned, but one hardware manufacturer has figured out a way to do it with its line of computers. The trick is to have a 60-Hz line-frequency interrupt cause periodic entry into the keyboard scan routine regardless of program activity. The scan routine in turn, does one scan of the keyswitch array and stores any key that is newly pressed into a character buffer for later use. Thus, it is possible to be several keystrokes ahead of program execution. Since the operating system takes care of all of the details automatically, the programmer doesn't even have to know about it. This same 60-Hz interrupt also updates an elapsed time "clock" in memory, which can be quite useful. ◇



"Remember, I promised you only 99.999 percent accuracy."