John F. Huntington

Selecting microcomputer software

The array of computer hardware and software available in the market is bewildering both in the complexity of its inter-relationships and in the speed with which those relationships (and functions) constantly alter. Huntington brings to bear the practical experience of an old hand who is uncommitted (unlike some) to any particular line of machines (many computer users have the same blind loyalties as car owners), and gives careful practical advice on such matters as selecting the basic collection, the requirements for memory, the purchase of peripherals, and the after-sales support to be looked for from vendors. He talks the language of the craft, but he knows what you want.

Microcomputer users can write their own software. During the early years of microcomputer production many of us wrote virtually all the software we used, out of necessity. There wasn't much available to do what we wanted done. Some adapted programs from larger systems, some used ideas and programming sequences from software consortiums, and so on. The time required to complete a program that ran smoothly and that had adequate documentation was extensive. It wasn't unusual to spend hundreds or thousands of hours writing, testing, debugging, and documenting. Fortunately, as the use of microcomputers expanded, collectives of software producers have been formed through association among various kinds of microcomputer users' groups.

Manufacturers then made corporate decisions about software support management that have had far-reaching implications for the success of their products. The most successful approach was to support only a very limited amount of software and to publish system configurations, so that the ever-growing numbers of users could enter the software

marketplace. The impact of this microcomputer support philosophy was tremendous, and the manufacturers benefited greatly. This is a rare example of consumer-oriented decisions acting to enhance greatly the desirability of particular products; it is unusual when consumers construct a product feedback system that materially affects the quality of these products.

This collective software production means that if you contemplate writing your own software, you should be aware that somewhere someone may have already invested the considerable amounts of time, psyche, and intellect to create a package that may suit your needs.

Seeking advice

It is not infrequent to hear the opinion that one microcomputer system is better than another, and that if you purchase a particular system much more software will be available for it. Another phenomenon I have noticed is that some users become ego-bound with their system, much as one may become involved with a particular automobile. If you are seeking advice, it is particularly important to know if the advice you are getting is tempered by the giver's experience with several kinds of microcomputers and related software. Thus advisors who use a 6502 microprocessor-based system might state that what they are doing on their system isn't possible with a Z-80 based system, when they haven't used the latter and consequently their judgment is seriously salted with misinformation and/or ignorance. The operating methods of these two systems are slightly different, but both are universally capable and some microcomputers employ both. The newer microprocessors are used in systems for which salespersons tout greater processing speed, more bits processed at a time, and more. In practice, most of the differences between microprocessors become transparent, and what differences we perceive are the differences as expressed by the designs of their operating methods.

The best judge of the integration of software and hardware to meet your needs is an individual who has knowledge of your specific needs and who also has knowledge of several systems from which to choose. Searching for the right advisors may be one of your most important steps in deciding which software is most useful.

The basic collection: an integrated system?

Microcomputers are most often used for word processing, data filing, data manipulations, game playing, and instruction. I have found that integrated systems are much easier to use than a collection of independently produced software. The common commands and structures represented by the software produced

by one vendor are much easier to remember and transfer from one use to another. It is not always possible, however, to find a package of integrated software that offers the best available features in each of its components. Thus if you are considering a spreadsheet analysis program and select Visicalc, you will find that the commands used in Visicalc are not the same as those used in other programs for word processing. (There are many versions of Visicalc - like spreadsheet analysis programs - that perform more functions than most versions of Visicalc, and that use a common structure and set of commands with a companion word processing program.)

The ArtSci Magicwindow / Magic-Calc / Magic-Memory series is an example of an attempt to integrate microcomputer programs so as to avoid the need to reorganize one's thinking patterns in order to operate the different systems. Some of the commercial/professional utilities use this approach. If, though, one purchases Scripsit or Applewriter along with Visicalc, and adds PFS or some other filing system, it is necsessary to remember further different commands and structures. This causes interference for many, and reduces the efficiency of use of these programs, especially when one uses the microcomputer infrequently. If graphics-generating programs and other utilities are added, the information overload produced by requiring the recall of so many commands can be overwhelming and discouraging. This phenomenon is not uncommon for those of us who operate more than one microcomputer system.

It is recommended that you carefully assess your abilities to recall and adapt, your need for consistent structure, and your ability to organize a documenting and posting of procedural steps which might be difficult for you to remember. Only if you possess these characteristics might it be advisable to search for utilities that individually provide the most use for your needs, but that are not highly integrated.

Should programs be menu-driven?

New users usually prefer programs that present a menu of choices and subchoices when a program begins. Menus generally increase user-friendliness. But once users have become familiar with the commands and options provided by the program, some may then wish that the menu could be suspended so that the time required to pass through the steps of the menu can be reduced.

Menu-driven programs are especially useful for users who are unfamiliar with or who have forgotten the command structure of the program. Moreover, errors of use are generally less likely when menu-driven programs are used. It is important also to consider whether a collection of programs might not include menu structures which may interfere with one another.

Memory requirements

Some programs run much more efficiently when larger memories are available for real-time processing. Larger memory permits more data to be analyzed at one time, and if some data are needed at any given time the larger memory might permit easy and quick access. Many microcomputers are equipped with 48K of memory, available as random access memory (RAM). Increasingly, the standard memory configuration is progressing to 64K of RAM. This standardization of memory is dependent on memory chip manufacturing. The chips have been available generally as 4K, 16K, 48K, each representing state of the art production techniques. Chips with larger RAM's are now being manufactured, and their availability will filter down to the appliance grade microcomputers which many call personal microcomputers. Programs selected currently should be written for at least 48K microcomputers.

Program selection should include considerations of available memory size. Different versions of some programs require differently sized RAM's. Visicalc, for example, is available in 48K and larger RAM configurations. The processing speed of the larger RAM versions is much faster. (However, an instructional programming language called Pilot is available in 48K (Pilot) and 64K (Super-Pilot), which means that the lessons prepared using the 64K version cannot be run on 48K systems.)

Programs which use lots of memory to store many files are quicker and usually more error-free than programs that use smaller dynamic memory and that must swap data files to and from external storage so that there may be enough space to manipulate the data in RAM. The trade-off in considerations of memory is cost versus storage capacity. Note that adding memory to existing systems may however result in diminished system integration.

Software and peripherals

Each introduction of enhanced models usually results in some incompatibility with the software designed for previous models. This occurred for the changes from TRS-80 Models I to IV, and from Apple II's to II+'s to IIE's. Programs give access to peripherals such as printers and modems, and these devices are addressed through slots located in the microcomputer. A slot may be "explicit", as a physical location for a plug, or "implicit", as an "address" in the memory. In the Apple II+ for example, the printer slot was usually designated as 1, and the 80-column board was usually designated as slot 3. The Apple IIE however will search slot 3 and will emulate this slot even though the 80-column board is located in its own special slot. If you have older programs which address slot 3, and you have plugged another device into this slot in the Apple IIE, the result might not be desirable.

When considering purchase of a program it is therefore important to judge a program's adaptability to newer versions of microcomputers. Using Applewriter II on an Apple IIE will not result in as smooth a working system as purchasing the newer version of Applewriter. Scripsit when used on TRS-80's will vary by versions used on Models I, I or IV. Apple Pilot or Super-Pilot is not easy to use with printers less intelligent than the Trendcom or Silenttype printers and interfacing boards.

The availability of such a variety of peripheral devices and interfacing cards for a system like an Apple microcomputer emphasizes the importance of making certain that the software takes advantage of configurations and fully utilizes the capabilities of this added equipment, as well as being adaptable to any newer devices which become available. Some software includes a driver diskette which permits you to configure and reconfigure the program so as to use various microcomputer systems fully. In some cases configuration routines are written on the master diskette containing the program. Even though these routines may seem less convenient, on a frequent-use basis they are very useful if you change or add to your microcomputer system.

Anyone contemplating the purchase of a microcomputer system should pay particular attention to the degree of integration of the various peripheral devices with software. The disadvantages of word processing using 40 columns when you could use 80 columns, or of not being able to view prints before they occur, are a result of the limitations imposed by a poor integration of software with the peripherals.

Program adaptability

In addition to adapting programs to take full advantage of various peripheral devices, it is possible for some users with knowledge of programming to enter a program and then revise it to fit local needs. (It is also possible that the software vendor may offer reprogramming services in order to tailor special programs for unusually configured microcomputer systems.) The language used to create software is important to know if you intend to revise the coding of programs. Those familiar with Basic may want to acquire programs written in that language, so that they may easily modify the programs. Programs written in machine language, to make more efficient use of the microprocessor, require users with more skill.

An assessment of programming skills is necessary to determine if attempts to modify programs are going to be worthwhile. In many cases the documentation of programming only includes general descriptions of coding sequences. In order to understand how the programming works in such a case, it is necessary to understand the implicit structuring of the program. Simple changes such as converting output to disk drives from tape output, or sending data to a printer as well as to a CRT,

tape output, or sending data to a printer as well as to a CRT, are not difficult for users with moderate programming skills. But most users I've known have not altered programs written in machine languages.

Quality of vendor support

When you buy an automobile you expect to get service and repairs. Too many microcomputer purchasers do not take this into account, or neglect it, in order to purchase equipment and software at discounted prices. There are many programs available in CP/M (Computer Programs for Microcomputers) that are of professional quality. They require considerable work to configure the system so that all peripherals work as planned. Many uninitiated microcomputer users will be overwhelmed by the Init and Config requirements of adapting a CP/M system to their upgraded system. A vendor should provide services which include setting up your CP/M system so that it will run the software without problems.

I would not recommend purchase of software without support services, unless you have access to a specialist who can provide these services. Vendors should also provide the necessary information about configuring special software so as to fit the specifications of your system. Lots of software isn't used because it requires more adapting than the users are able or are willing to provide. You would therefore be prudent to be sure what services are provided by the dealer when you purchase software.

It should also be noted that when a microcomputer system is upgraded, additional reconfigurations may be required. The inclusion of an undertaking to reconfigure the software when the system is changed would be a useful bonus.

Intersystem compatibility

Software is written for specific microcomputers, that use microprocessors in special ways. Hence a program written for a Z-80-based Heath/Zenith will not operate on a Z-80-based TRS-80. Many programs written for microcomputers use a version of microsoft Basic, which is a standard for the industry. Though programs written in microsoft (such as Applesoft) are similar, they are not exactly the same, because the memory locations used to store or retrieve data have different assignments from one system to another. In addition, the storage of the program on tape or disk is in a format unique to the operating system of each microcomputer system. While it may be possible to read a program into a microcomputer as a text file through telecommunications procedures, it will be necessary to adapt parts of that program so as to operate successfully on a different kind of microcomputer.

Program compatibility is provided for in the CP/M systems. In effect, each microcomputer is converted to a Z-80(or other)-based CP/M system by adding a board to the microcomputer. This is a piggyback addition of a microprocessor, which uses much of the microcomputer's operating system to enable use of the CP/M system. It is possible to use the same CP/M-based programs on a Model IV TRS-80, an Apple IIE, Commodore 64, Atari 800, Heath/Zenith, or IBM-PC, when these microcomputers have been configured to CP/M standards. The additional cost of CP/M configurations and the increased cost of CP/M-based software may deter some from considering this system. It is not an easy system to use, especially for beginners and casual users. It is, however, at present an important choice to permit software compatibility among different brands of microcomputers.

Another method of achieving software compatibility is through the use of system-emulating boards. These boards are really like limited versions of the microcomputer they are emulating. The extent of their abilities to read all of the software written for the emulated system remains unknown. But some permit reading and also writing operations: a board called Quadlink is available for IBM-PC's to permit this microcomputer to read and write programs designed for the Apple II. However, a board for the Commodore 64, which permits you to use programs written for the Apple II, limits you to executing the programs, and does not permit your writing a program to disk.

Recommended software collection

The collection you create should be tailored to your needs and interests. Most operations can be performed by acquiring word processing, spreadsheet analysis, and filing system programs. I use these programs for most of my microcomputer applications, with the exception of instructional programming. If your interests include music writing, speech synthesis, graphics production, stock market analysis, analog-controlled devices (like alarms and thermostats), games, communications networking, robotics, or other less common applications, then the choices are more specialized. It is important to estimate accurately how much use the program is going to be or how important will be its application. There are lots and lots of rarely used, expensive programs residing passively in diskette storage boxes. Of the programs I've collected over the years most are outdated and archaic, and they represent an investment that had to be amortized during a relatively short time period. Consider this aspect when you review software. A premier word processing system like Wordstar, which is very expensive, is being challenged seriously by less capable but more friendly systems represented by the Bank Street Writer. Visicasle, as mentioned previously, is now competing with newer

systems that are more flexible and better integrated with common vendor word processing systems. If you seek social support for your microcomputer activities, then choosing software that is used by your associates is an important consideration. The best advice one can give is, try it before you buy it.

John Huntington has studied CAI languages since the 1960's. He is a Professor and Assistant Chairman in the Department of Educational Psychology at Maimi University in Ohio, and in addition is a columnist, author, consulting editor, and presenter.