Using Microcomputers in the Laboratory

JAMES P. AVERY
Department of Electrical and Computer Engineering, University of Colorado, Boulder, Colorado 80302

Received December 28, 1982

This series of papers is intended to aid the chemist in determining what type of microcomputer equipment is needed in developing computer-aided instrumentation systems. This first article will try to define the types of instrumentation systems that are commonly connected to computer systems and the magnitude of the computer systems needed to effectively improve the operation of the instrument over the level that existed prior to computerization. Program development tools and the computer "environment" needed for their support will be discussed. The attitude that will prevail throughout is one of healthy skepticism, with the understanding that many chemists would not, of their own volition, allow the computer into the laboratory if modern experiments did not require it. In other words, these articles will attempt to define the minimum level of computer science and electrical engineering expertise needed to give the chemist a fighting chance at making the system work.

For all of our specialization and advanced technology, all of our instrumental systems are fundamentally similar. Mass spectrometers, UV–visible spectrophotometers, electrochemical instruments, chromatographs are all kicked by some physical event or series of events at one end, and numbers fall out the other. Sometimes the numbers are voltages and other times currents, counts, or weights, but the principle is the same. The primary function of the computer in such a system is to determine from the operator how and when to kick the system and what to do with the numbers when they appear. The type and complexity of the computer needed to control the instrument are determined primarily by two considerations: first by the speed and complexity of the "kick" and second by the number of numbers that result and the speed with which they show up. For an amazing number of systems, the kick is simple and direct, and the numbers are few and slow to appear. This leads many to the conclusion that a simple slow computer would suffice for these applications. This is a mistake.

"IT IS A SIMPLE MATTER OF SOFTWARE"

The reason for the error is a fundamental one in the use of computers. The computer that controls the instrument can, in fact, be slow and simple. However, someone must *program* that computer to do the slow simple job. Unfortunately, even simple programs are time consuming to write, even under the best of conditions. Slow, inexpensive, simple computers usually do not provide the type of program development tools (software) needed to produce good, efficient programs easily. The result is predictable: either much more work is done to make the programs good or, far more likely, the programs are not good. Programs written without good software development tools tend to be convoluted, error filled, and hard to use.

They are convoluted because making changes is hard, so instead of fixing a problem properly, it is "patched"—written around in some complicated way. A program with more than a few patches becomes unintelligible to all but its author. A program with a large number of patches will defy even the author's attempts to explain or alter its function.

The error-prone aspect results more commonly from the use of assembler language routines to perform the control operation. Assembler is never a joy to use, because it deals with the machine at the machine's own level. This puts the burden on the programmer (i.e., the chemist), who is not used to dealing with the machines. Some of the simpler machines actually require the programmer to translate the assembler code into machine language by hand, a nasty task even for short programs, as error prone as doing multiplication with logarithm tables and about as up-to-date.

The programs are hard to use because easy to use programs are hard to write. The magic word for a good program is "user friendly", which many programs are supposed to be and few are. To be friendly is to be forgiving: to allow the user to recover from mistyped statements gracefully, without reentering whole sections of command. To be friendly is to speak the user's language: not to ask for wavelengths in centimeters, potentials in decivolts, or time in hexadecimal centiseconds. To be friendly is also to ask for inputs that have some relationship to the experiment as the experimenter remembers it before the computer took over. Many of the parameters that the computer uses to control the "kick" or take the data are derived from the "physical" parameters with which the chemist is familiar. It is silly to have to use a calculator to calculate the value of a parameter prior to feeding it into a computer. But the difficulties faced by the chemist/programer in developing the program on a simple system often make this the simpler path. After all, someone who knew what was going on has already programed the calculator.

HOW NOT TO BE SUCCESSFUL

The primary idea to remember is that the computerization of the experiment is supposed to help things: improve reliability, increase throughput, reduce tedious calculations, present the results in a more useful form. The difficulty in the current environment is that the hardware to do these things is cheap. People remember when the software costs were about half the price of the total package (as they were a very few years ago) and think that the same is still true. It is not. Mass production is driving the cost of computers down; no similar force is yet active in the custom software market. And custom software is what the chemist needs, because for all the similarities between instrumental systems, each one is still a little different. Software is still a labor-intensive commodity. Most of the disappointment with computers, whether with personal computers in the home or laboratory computers in instruments, stems from failing to recognize this fact: programing is hard work. One manufacturer used to say that all you needed to use his computer in the laboratory was a screwdriver, which he supplied. It ain't true.

There are two types of groups whose work seems to belie these statements. The first is the seasoned veterans of the

0095-2338/83/1623-0001$01.50/0

computer revolutions. These folk have lived and breathed computers for 10 years, are fluent in six computer languages, know the architecture of 12 microprocessors intimately, and can interface three impossible things before breakfast. The newcomer looks to these people for guidance and finds they have moved off in pursuit of more challenging game: self-modifying instruments, laboratory networks, multidimensional instrumental systems. These are important areas for the future and must be explored. But to follow in these footsteps, one needs a very long time for learning—time most chemists would rather spend with chemistry.

The second group is even more insidious. The are the academics who, armed with a $279 computer, a $50 purchase order made out to Radio Shack, and three bright, agressive juniors, have interfaced an instrument five times more complex than yours. What nobody bothers to count is the man-hours involved in the operation. What research would have been done with those same hours if they had not been spent on the computer? Will the improvement in the instrument ever pay off? It can be done. The question is whether it is worth the cost.

SOLUTIONS

The preceding bad news was designed to bring into focus the area which will cause the most difficulty in adding a computer to an instrument: the design and implementation of the software. As subsequent articles will discuss, the hardware really is not that much of a problem in many cases. How then can the software problems be minimized?

The first step is to admit that there are software problems. Throw people who say "It's only a software problem" unceremoniously out of your office. Recognition is the first step to cure.

Second, ask upon what you should be spending your time. If you enjoy translating arcane code into hexadecimal numbers, fine. But if you still tell your offspring that you are a research chemist, spend the money you need to get help. This means buying systems that meet at least the following specifications: (1) a good BASIC interpreter; (2) a good, usable keyboard, and a clear display; (3) a reliable printer; (4) access to some other high level language such as PL/M, Pascal, and even FORTRAN (not, however, COBOL); (5) at least two floppy disk drives; (6) at least 64K of memory; (7) instruction manuals in English; (8) at least one other chemist owner; (9) access to a decent assembler/debugger package. These will be discussed in order.

BASIC is much maligned. Improperly used, it has the capability of turning clear algorithms into code so opaque not even the author knows what is happening. It does not have to be used improperly, however. The magic word in this area is "structured" programing, which is primarily a codification of the common sense idea of divide and conquer. The programing task is divided into subsections which communicate with each other in carefully controlled ways. The advantages of a good BASIC interpreter are that program development time is short, that errors are relatively easy to fix without time consuming edit/compile/load cycles, and that the language is easy. If you type

```
PRINT "START EXPERIMENT"
```

"START EXPERIMENT" is printed. The language is close to English; it can be learned by mortals in a few hours or days. The best thing about BASIC is that it is easy to write programs that ask the instrument's user what the instrument is supposed to do, in English, using parameters and values that mean something to the user.

The disadvantages of BASIC are that it is slow and that it is limited in its ability to handle some types of interfaces. There are academic reasons for disliking BASIC, especially its awkward way of handling subroutines, and other features that make structured programing difficult, but the slowness is the main problem. There are two routes to circumvent this: assembler routines linked to BASIC programs or writing in a compiled language. The assembler route produces the fastest code but the greatest agony (see below). No BASIC that cannot call assembler language routines should be considered for any but the simplest of projects.

The keyboard and printer are essential to the preservation of the programer's sanity. A keyboard that repeats keys, or does not type some at all, is a considerable pain. The lack of a printer will make programing much harder than it otherwise would be. The speed of the printer has an interesting effect upon the length of programs that people write. Most people will wait about 10 min for a listing. The number of lines that the printer can list in 10 min is a good estimate of the convenient length of a program.

Some sort of compiled language capability may save you from having to go to assembler. Compiled languages usually, but not always, run faster than interpreted languages. This is because the translation to machine code is done before the program starts, rather than on the fly as in an interpreter. The price for this is a longer turn-around time between fixing an error and trying the program and the requirement in nearly all systems that there be some mass storage around to load the source program, the intermediate output, and the machine code. Cassette interfaces will not reliabliy handle these storage requirements.

Two floppy disks are a minimum for any program development system. Currently, this will provide storage "on-line" of about 128–300K bytes per disk; disks are cheap, reliable, fairly fast, and easy to maintain. A single disk drive is OK on a second or third system, but in order to copy disks for the purpose of backing up valuable information, a two-drive system is essential.

It is not possible to get too much memory. Memory is cheap, and prices are continuing to fall. It is very difficult to cram extra code into a very small region of memory. Be advised also that the first estimate of memory need is often off by a factor of four or five, never on the high side.

Manuals are the sore point of the industry. It used to be rumored that the manuals of a major minicomputer supplier were written by their janitors, who pasted up random bits of paper found on desks and wastepaper baskets. Manuals have improved since then, but others carry on the tradition. It is usual even for veterans to read an entire manual before attempting to read it the first time for information. See if there are manuals at your current level. But beware if only introductory manuals are available, because eventually you will try something that is not novice-class stuff.

A friend with the same computer using it the same way is worth 40% off of the list price of the computer. If that is not obvious, do not buy any computer at all.

Eventually you will need to do something in assembler. It is a rite of passage. Also, as your use of the computer becomes more sophisticated, you will eventually run into a program that cannot done without the speed that only assembler gives. To do this, a good assembler and a well thought out debugging system are needed. This is last on the list to discourage its use.

The next paper in this series will look at some of the commercially available offerings and discuss and debunk the great "which is the best microprocessor" debate.

Development of CAOCI and Its Use in ICI Plant Protection Division†

S. BARRIE WALKER

Technical Information Section, ICI Plant Protection Division, Jealott's Hill Research Station, Bracknell, Berkshire, United Kingdom

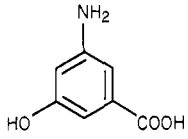
Received December 3, 1981

Research chemists involved in the synthesis of novel organic compounds need chemical intermediates and spend valuable time searching through suppliers' catalogs for useful compounds. The Commercially Available Organic Chemicals Index (CAOCI) data base, an application of WLN, provides the chemist with a means of identifying the commercial availability of a specific organic compound or group of closely related compounds by means of substructure search. The data base has been developed since 1974 when six member companies of the CNA (U.K.) agreed to cooperate in its production.

Chemists involved in the preparation of novel organic compounds for evaluation as agrochemical, pharmaceutical, or other products require starting materials and intermediates in laboratory quantities. It is usually the case that where a chemical can be purchased this is preferable to the chemist spending his valuable time making his own intermediates—in other words, time should be devoted to making speculative compounds for screening rather than the chore of making commercially available compounds.

How does the chemist know whether or not he can buy a specific compound? He undoubtedly will have an assorted collection of chemical suppliers' catalogs that he can look through, but this is a time-consuming and often unrewarding job. Many suppliers' catalogs give only an alphabetic name index. Better ones provide cross-indexing of names, molecular formula indexes, and sometimes chemical-class indexes. Aldrich is the only company to offer a substructure search facility, but of course this covers only their own catalog.

Let us examine the stages that the chemist goes through. Firstly, he has to think up a name for the molecule he requires. This can be a major hurdle.



For instance while the molecule above will have a systematic name, it could be named as an aniline, as a benzoic acid, or as a phenol by the chemist, depending on his use for the chemical. What is the answer to this problem? Well, whatever chemical name you give to a compound, and there are of course examples where many more than three names could be used for one compound, it will have just one Wiswesser Line Notation (WLN):[1]

ZR CQ EVQ

WLN could, then, provide the key to easy access to availability of chemicals.

The idea of using WLN to bring together the same compound from different suppliers' catalogs was discussed within the U.K. Chapter of the Chemical Notation Association—the CNA (U.K.)—at the beginning of the 1970s. Up to that time the main application of WLN had been for the indexing of in-house chemical collections, and using WLN as a tool to bring together information on the same compound was a new departure.

However, by 1972 the production of an index of all commercially available organic compounds was still being discussed and no progress had been made. During the winter of 1972/73 an index of 20000 organic chemicals from 10 small catalogs was produced in ICI Plant Protection Division (PPD). This was done in support of an index of suppliers' information which already existed in ICI, but which was limited to data on just 12 U.K. suppliers. The sorted output was taken to a meeting of interested people in the CNA (U.K.), and this small data base became the starting block for a project which began in 1974.

CNA (U.K.) and U.S. members were invited to contribute some form of resource, e.g., coding effort, card punching, or computer time and programming effort to compiling a larger file on the basis that work would be shared and the product made available to all contributors. A number of people volunteered their company's cooperation in the project, and then the crunch came. The group began to work out just how much effort would be needed in an initial phase of the project. We finally ended up with six participating companies.

Imperial Chemical Industries Ltd.
The Wellcome Foundation Ltd.
The Boots Company Ltd.
Pfizer Ltd.
Glaxo Group Ltd. (then including Allen & Hanburys)
Beecham Pharmaceuticals

With some companies being more involved than others, but in reasonable proportion to their size, ICI took on the role of coordinating the work and producing computer programs linked to parts of CROSSBOW[2] software to establish and maintain the file. Coding and card punching effort was provided by the other contributors, and a file of Aldrich data was acquired (by ICI) and added into the file.

This phase of the work took approximately 18 months, and by mid-1976 the first product was available. By Sept 1976 the members of the group were in possession of hard copy molecular formula and WLN-ordered indexes. At this stage the group decided on the rather unpronounceable name of CAOCI for the project—*C*ommercially *A*vailable *O*rganic *C*hemicals *I*ndex.

Let us now return to our hard-pressed bench chemist frantically thumbing through his jumble of catalogs. Reference to CAOCI will quickly tell him whether the chemical is commercially available or not. Having ascertained that any specific compound is available and from which suppliers, the chemist can then refer to the appropriate catalogs to check pack size and price and order accordingly. If the chemist can see that a compound is available from five suppliers, perhaps one of them is his favourite or has a reputation for supplying quickly.

It did not take long before the molecular formula and WLN-ordered indexes were in the hands of the chemists, who quickly accepted the product and wanted more coverage. At

† Paper presented at the Symposium of the Chemical Information Division of the American Chemical Society, Las Vegas, Aug 1980.

0095-2338/83/1623-0003$01.50/0