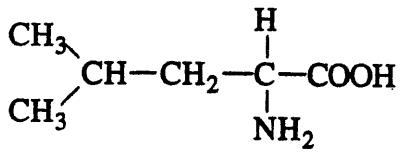
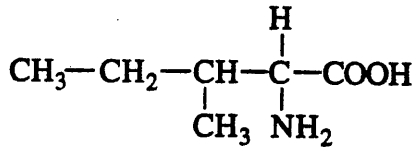
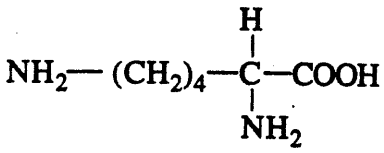
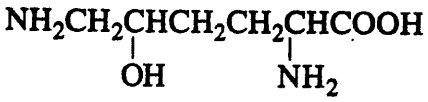
Computers in Chemical Education Newsletter

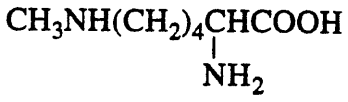
Spring 1995

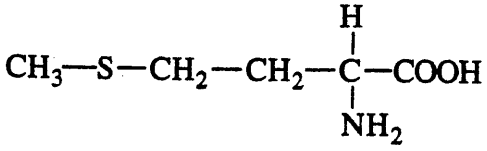












Published by the ACS Division of Chemical Education
Committee On Computers in Chemical Education
Don Rosenthal Chair

Editor Brian Pankuch, Department of Chemistry, Union County College, Cranford, NJ 07016
pankuch@hawk.ucc.edu.

Submissions: General articles should be sent to editor Brian Pankuch at the above address. We would appreciate both 1) printed copy (hardcopy) and 2) a readable file on a Macintosh or IBM compatible 3 1/2" diskette. We have fewer problems with 3 1/2 " diskettes.
Submission deadlines: Fall issue - Sept. 25; Spring issue - March 15.

ALL NEW AND RENEWAL SUBSCRIPTIONS : PLEASE SEND REMITTANCE TO M. Lynn James, Dept. of Chemistry, University of Northern Colorado, Greely, CO 80639.

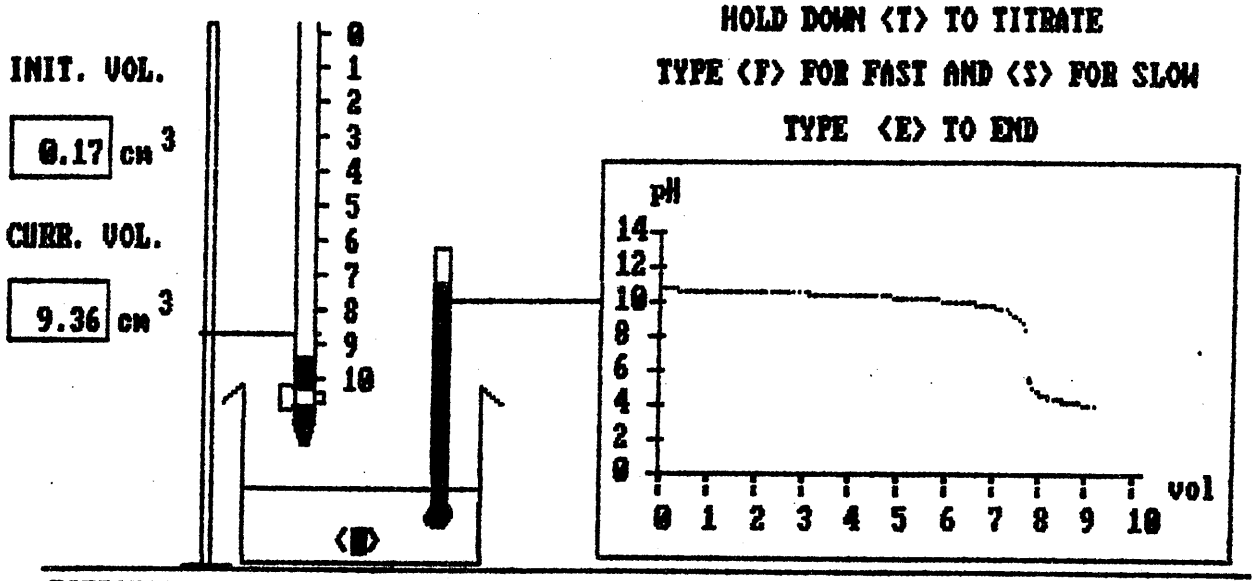
RATES: USA 1 year $2.50, two years $4.50: Other countries 1 yr $5, two yr $9. Please make a check or money order payable in US funds to Computers in Chemical Education Newsletter.
One to three issues are published per year.

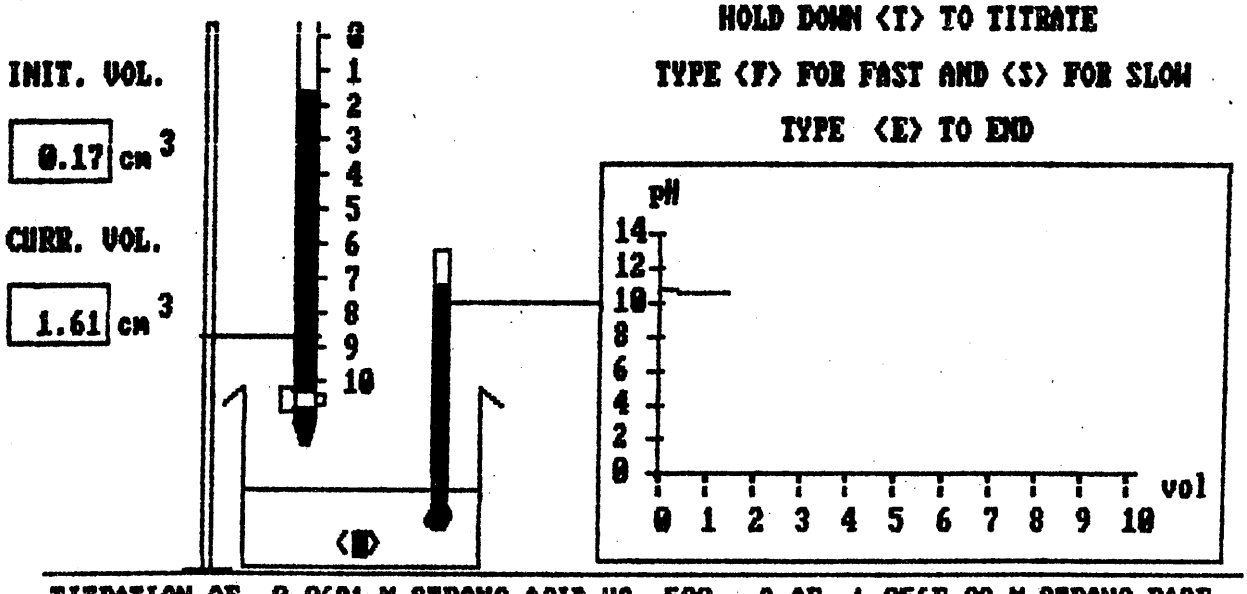
Cover: Stuff

The newsletter is done using Aldus PageMaker 4.2, and is printed using a LaserWriterII.

Computers in Chemical Education Newsletter

Spring 1989





Published by the ACS Division of Chemical Education
Committee On Computers in Chemical Education
M.Lynn James, Chairman, Dept. of Chemistry, University of Northern Colorado, Greeley, CO 80639

Editor Brian J Pankuch, Department of Chemistry, Union County College, Cranford, NJ 07016

Submissions: General articles should be sent to editor Brian Pankuch at the above address. We need both 1) printed copy (hardcopy) and 2) a readable file on an IBM compatible diskette , 5 1/4 or 3 1/2. We have fewer problems with 3 1/2.
Submission deadlines: Fall issue - Sept. 25; Winter issue - Jan. 25; Spring issue - March 25.

Some Supported Word Processors	File Format
DCA files	.DCA
Microsoft Windows Write	.WRI
Microsoft Word	.DOC
MultiMate	.DOC
WordPerfect	.WP
WordStar 3.3	.WS
XyWrite III	.XYW

For example if you use MultiMate to make a file named foo send it as foo.DOC. FOR OTHER WORD PROCESSORS SEE Aldus PageMaker 3.0. PLEASE DO NOT SEND ASCII FILES.
Please see the backcover for more complete list of readable wordprocessors, we can only handle IBM compatible applications.

ALL NEW AND RENEWAL SUBSCRIPTIONS : PLEASE SEND REMITTANCE TO LYNN JAMES USING THE SUBSCRIPTION FORM IN BACK.
RATES: USA 1 year $2.50, two years $4.50: **Other countries** 1 yr $5, two yr $9. Please make a check or money order payable in US funds to Computers in Chemical Education Newsletter.

COVER : Graph is from program PhTitration by R.J. Lancashire, Dept of Chemistry UWI, Jamaica.
The newsletter is produced using Aldus PageMaker 3.0, and is printed using a AST turboLaser/PS, with the help of the Union County College Resource Center.

MESSAGE FROM THE CHAIRMAN

A valuable resource has developed to encourage the development and use of high-quality, educationally sound software in undergraduate liberal arts subjects. Two years ago EDUCOM and the National Center for Research to Improve Postsecondary Teaching and Learning (NCRIPTAL) jointly established an annual awards competition for this purpose. This effort provides a valuable resource for educators who desire to incorporate quality software into their programs but find it difficult, for a variety of reasons, to locate and adequately evaluate the software that is available. In the 1988 competition, the CCCE Newsletter Consulting Editor, Don Rosenthal, and I had the opportunity to be members of the National Review Panel. The results of the competition in chemistry are reported below.

In the competition, awards are given in two divisions, the Product Division, for new software, and the Curriculum Innovation Division, for innovative uses of software in the classroom. Overall, the 1988 program attracted 182 qualified submissions in all areas. From these were chosen seven "Best" products and seven "Distinguished" ones along with three "Best" and two "Distinguished" curriculum innovations. The details of the competition are given, among other places, in the article "The 1988 EDUCOM/NCRIPTAL Competition" by Jerome Johnston and Robert B. Kozma in the November 1988 issue of Academic Computing. The purpose of this article is to give a brief summary of the program and to alert you to those packages in chemistry which were judged outstanding.

NCRIPTAL, which is located at the University of Michigan at Ann Arbor, runs the software competition. EDUCOM, which is a consortium of more than 550 colleges and universities concerned about computer use in the campus environment, develops support and publicity for the program. Financial support comes from a variety of the leading computer hardware and software companies. Of particular interest in 1988 was the emergence of winners in the Curriculum Innovation Division which emphasizes the use of computers to solve important local educational problems or needs. Rather than being concerned with the writing of software as in the Product Division, this Division is interested in such solutions if they can be generalized to other situations and focuses on the overall design of instruction and how the technology supports and improves it.

The Review Panel for any given subject area typically consisted of two content oriented and two pedagogically oriented individuals. They were all asked to evaluate the design including its appropriateness and validity.

The winners in chemistry for 1988 were both in the Product Division and include the following: Best Chemistry and Best Simulation Software: "Simulated Qualitative Organic Analysis (SQUALOR)" by Donald Pavia, Western Washington University. The program is available for the IBM PC, PC/XT, PC/AT, PS/2 series and compatibles. Receiving the Distinguished Software Award was the package, "Concepts in General Chemistry" by John Weyh and Joseph Cook, also of Western Washington University and Les Hauge of Spokane, WA. This software is also designed for use on the IBM PC and compatibles.

The awards program will continue as an annual competition. Although this article will appear too late for any of our numbers to apply for the 1989 awards program it is hoped that those of us who are involved in this type of work will keep this in mind for future years. Further information can be obtained by contacting the Software Awards Program, NCRIPTAL, 2400 School of Education Building, The University of Michigan, Ann Arbor, MI 48109-1259; 313/936-2741.

Editor

Son of a gun we've made it through an academic year. My thanks to all our contributors, subscribers, Lynn James, and our editorial and secretarial staff.

For the next academic year Don Rosenthal would like someone else to take over as Coming Events editor. We need someone to accept notices for meetings, courses, etc., and send them to me in chronological order. We continue to welcome contributions from all of you, it is your newsletter. So if you're out enjoying the garden, fishing, windsurfing (my favorite), hiking in the mountains or whatever and you get a beauty of an idea for you or your students using your computer - share it with us . Send us a letter or larger piece. The general interests of our subscribers were published in the Winter issue. General guidelines for submissions are given in the inside of the front and backcovers. If you can send material under our guidelines great, if not send it anyway and we'll get it into our system somehow. Please always send hardcopy- I've had more damaged diskettes in the last 6 months than in the previous 10 years.

Most of all enjoy! Hope to hear from you by September. What follows is a review of an excellent program I've been using with my Pascal students for 3 years. Readers response to our last survey showed most of you program in BASIC, but a higher percent of Pascal programmers were passionate about it. If you want to learn more about Pascal programming, and perhaps add more passion to your life, get a good text and this programming tool and you'll learn a lot.

Dr. Pascal $89.00
$35 for students.
From Visible Software
P. O. Box 7788
Princeton, New Jersey 08543
(609) 683-4386
Reviewed by: Brian J. Pankuch

Dr. Pascal is an outstanding set of software tools and more. You get a compiler, a screen editor, a procedure editor, and a general set of tools that the manual characterizes as "visibility" or visible options. The system allows fantastic insight into how a program actually works. It helps you to improve writing, debugging, and making changes in your programs.Dr. Pascal will help whether you are a novice or an experienced Pascal programmer.

Let me take a moment to explain what I see as some of the uses for this package.

1. If you're learning Pascal, sit down with your textbook and use the Dr. Pascal editor. Type in the example programs given in the text. The compiler will be quite helpful in finding typing and syntax errors. While running the program you can turn on various aspects of visible options. You can watch the computer go through the program at a wide range of speeds. The value of variables you have in the program are automatically shown on the screen as they change. You can step through a line at a time - the line the cursor is at is always highlighted whether you or the computer is stepping through. The screen is automatically broken up into windows. In one window you see about 20 lines of your program with the line you are at highlighted. Another window shows all the variables and data structures used in this part of the program. You see how each changes - what value they contain at each line of the program. Talk about insight! Fascinating with arrays and records.

2. How about when you are writing your own program? The compiler and the editor work in sync so if the compiler finds an error you can go to the screen editor and make changes for errors found by the compiler. Error messages seem clearer than most I've seen. You can run the program. If you don't get the answer you expect, turn on the "visibility". Step through, see a variable that you forgot to initialize? Freeze the program, put in some temporary code to initialize the variable. Continue stepping through from where you froze the program. Find other errors? Put in some more temporary code and fix the errors. You can opt to have the temporary code run or not. Does the program work now? If it does you can then go to the editor and put the temporary code in permanently. Tools like this really speed up the debugging process. You wind up writing the program faster, and with fewer recompiles. The final program is much more likely to do the things you want. There is less chance of those extra surprises sneaking in.

3. An important point for chemists who are trying to use computers in teaching chemistry - what can we learn from tools like this? Can we develop tools for learning about chemistry that are helpful as this is? I think we can use Dr. Pascal as a model. There is a lot in common between designing a program using top down methods, and mapping out the step by step procedure in solving a chemical problem.

I received this package when I had about 50 minutes between classes, so I ran down to our microcomputer lab to try it. My luck was good and a Leading Edge (an IBM-PC compatible) was available. I sat myself down, opened the 160 page User Manual and began reading. The introduction was interesting but I really wanted to get the system up and see what it could do. I skimmed through looking for directions on how to get started. To make it simpler for you to find - the "getting started" instructions were in the back, in two pages from the back and on off colored paper. I'm probably the only one who would spend 20 minutes looking for these directions, but just in case now you know where to look.

It seemed sensible to put Dr. Pascal up on the hard disk (it can also be used from floppy disks). I had a few problems because I had never worked with any of this hardware before. A few minutes help from a staff member who was familiar with our equipment got Dr. Pascal up and running. The manual proved to be clear and well written. It is an excellent tutorial on the Dr. Pascal system. If you want to learn Pascal, you will need a book on Pascal. Using Dr. Pascal will speed your learning the language.

The tutorial shows you step by step how to use the system with example problems that are already on the disk. I would like to suggest immediately photocopying page VII in "getting started" at the back of the manual since it contains a listing of what each key does. The system itself will always give you a prompt line of what you can do currently from where you are in the system. Since what you really want to do may be two or three steps away, it is helpful to have information on all of the keys right in front of you while you are learning. There is always a help screen available but the sheet is also nice to have.

The second example program accepts numbers as input from the keyboard and then calculates and prints out the mean. The program has a deliberate bug. You are encouraged to turn on some of the "visibility" and step through the program. Sure enough, when you see the values of the variables used in the program change as you step through it is obvious that the program is not behaving as you want. Closer scrutiny shows where the problem is. The tutorial then takes you through the corrected program and everything works fine.

Another example program prints graphs. This one has two procedures in addition to the main program and you get to use Dr. Pascal's procedure editor. It shows the program at procedure level. Each procedure is shown just by its name with indentation representing the nesting of the procedures (how procedures are contained in one another). You can watch the program run line by line as you did before, or you can watch how procedures call each other. You can also change the nesting of the procedures by simply changing the location of the procedure's name. The system will automatically change the scope of all the variables that have been changed.

If you need further proof of the power of being able to look at how procedures call each other, look at

the fourth program example. It is under a page long and finds all the sets of positions where you can put eight queens on a chess board so that no queen can capture another. The problem is solved recursively and if you want to see a computer work! Wow! It's really impressive and quite instructive if you want to understand recursion.

For small programs, a single keypress makes your entire program "visible". The manual shows you how to get the most out of the visible options for large programs. Quite a lot is available. You can look at the smallest detail in a procedure. When the procedure is thoroughly debugged and understood you can make it invisible. Go on to the next procedure and work on it. Check each and the interactions between the procedures. These tools really speed up the process of getting a program running properly.

Some of the other pluses for the system:

1. The controls you have over the program are many, and easy to learn and use. Example - if you don't like the way keys are assigned in the editor you can change them to be the same as your favorite editor.

2. It's great as a learning and exploring tool.

3. You can put in breakpoints for program testing.

4. The screen editor has templates available at a keystroke for the major Pascal structures. If you want a FOR template you just press the appropriate key and it appears at the cursor with the correct syntax. You just add the correct variables or numbers in the slots provided.

5. Since your program is saved as a Standard ASCII file, you can use it with other compilers after it has been written and debugged in Dr. Pascal.

Some of the minuses, all rather minor:

1. When the compiler reformats the program it removes white spaces, i.e., extra spaces unneeded by the compiler. For instance, if you write A = 3 + 7 - 55, it will give you A = 3+7-55. Personally, I find the extra spaces easier to read, and I make fewer mistakes when I write this way.

2. When you have run errors it is a little awkward correcting them. You will probably have to edit your program and start it running from the beginning.

Dr. Pascal works on all IBM PC compatibles with at least 512K, and most other MS - DOS and CP/M-86 systems. The developer is working on other systems such as the DEC Rainbow((. By the time you read this, Dr. Pascal may be ready for other systems. If you are interested, check with the company about availability and considerable discounts for quantity purchases by educational institutions.

Dr. Pascal is an excellent set of tools and a very good buy.

An Animation of Distillation
Part I
The Flame Vic Bendall, College of Natural and Mathematical Sciences, Eastern Kentucky University, Richmond, Kentucky 40475

The pedagogical value and overall attractiveness of computer aided instructional software is greatly enhanced by the incorporation of animation. Unfortunately good animation is much more time consuming to author than the routine program. To get fast animation, particularly when two or more actions appear to take place simultaneously on the screen, requires careful planning and logistical skills. In this series of articles, I will describe how the animation of a distillation was approached.

In the usual distillation most of the apparatus is static and can be easily drawn using conventional graphic commands on the monitor screen. The animated portion consists of a flickering flame to illustrate heating, a rotating stirrer bar in the heated liquid, liquid drops falling from the condenser and liquid accumulating in a collection vessel. Our task is to write a routine which will appear to execute all four actions at once. For illustrative purposes, we will write separate routines to perform each action separately and then combine them into a smoothly working whole.

This first article will describe the flickering flame routine. We need a routine which can be easily correlated with the subsequent routines. A BASIC program which shows a flame is given in the documentation of CHEMUTIL-2 (1). That routine works by defining nine 7x8 bit characters which each show a different view of a flame and having those characters available in the alternative character set. The nine characters are then printed successively directly on top of one another and a flickering flame results. The CHEMUTIL-2 documentation code cannot be used directly for the distillation because it is not general enough to allow the incorporation of other routines without excessive jerkiness in the combined routines. For example, rapid showing of the nine flame images followed by a falling drop and then nine more flames, will result in the flame being stopped while the drop falls. We need the drop to fall smoothly without the flicker being interrupted.

As a general rule, code written for a single application will be more efficient than that which can be easily adapted for other purposes. But general purpose code is more useful. Thus, we can write distillation code which is specific for that alone or the code can be written so that with just minor modifications, it would be used to demonstrate heating under reflux with or without a Dean-Stark trap. To allow for those and other future unanticipated uses, the code we shall write will be made as flexible as possible.

The first step is to allow for movement of the flame image to any screen location. In a distillation the flame is in a different relative posi-

tion with respect to the falling liquid than in a refluxing situation. Being able to move the flame independent of the falling drop and stirrer will made the coded routine more useful. It will certainly help when we attempt to combine flame and stirrer together. The code will be a loop of the form: -

Go to flame location: Draw flame image: Go to stirrer location: Draw stirrer image: Loop back

The stand alone flickering flame routine consistent with the previous loop will have to be in the form: -

Go to flame location: Draw a flame image: Loop back

The delay is deliberately introduced so that the routine can be tested independent of the stirrer and drop routines to be added later. Those routines will later replace the delay, but it is needed now so that we can be sure that the independent routine will be easily integrated with the others.

Now we cannot draw the same image of a flame each time through the loop or it will not flicker. We could blank out the flame image on alternate passes to make it flicker, but the result would be rather dull. The CHEMUTIL-2 method is to overprint nine different flame images rapidly. The loop will not be of the form: -

Go to flame location: Draw flame (1): Delay: Go to flame location: Draw flame (2): Delay: etc.

After nine flame images have been drawn, the loop starts over again.

Finally, we do not want the loops to be endless. There must be some way to turn off the flame. It could be done by stopping the execution of the code after a predetermined number of times through the loop. More generally useful is to us a keystroke to terminate it. The advantage of that is that the code now allows easy introduction of alternatives depending upon which key is struck. For example, striking the S-key could stop the animation and turn off the flame, while striking the C-key could leave the flame on but start the stirrer, too. The point is not that we intend to use the C-key in that way, but that we allow for that possibility before starting to write code. In a complex animation it is not easy to add additional features after coding without undesirable effects appearing.

A BASIC program to show a flickering flame can now be written. The following one is for Apple II+, IIe or IIc microcomputers and will show the flame on the text screen.

```
10 Home
20 For I = 97 to 105
30 VTAB 10 : HTAB 10
40 PRINT CHR$(I)
50 X - Peek(-16384):POKE(-16368),0: IF X=211 THEN I = 105:GOT0 80
60 NEXT I
70 GOTO 20
80 VTAB 10 : HTAB 10 : PRINT " "
90 VTAB 20 : END
```

The flame image is made up of successive images from ASCII(33) through ASCII(41). Line 50 stops the illusion when the S-key is struck and Line 80 extinguishes the flame. If CHEMUTIL-2 is available, BLOAD it and replace line 10 by CALL 25042:PRINT "&". The routine will show a true flame on the graphics screen.

BASIC is not the best language for this routine. A satisfactory illusion is obtained here only because the routine has no other tasks. The only delays are the usual ones as the computer reads and interprets each line. If even a small delay loop is added to simulate an additional task, the illusion is lost. For example, add the line: -

```
55 FOR K = 1 to 30 : NEXT K
```

Upon execution, the routine shows each character clearly and is unacceptable. Yet this delay is too short to allow the introduction of the stirrer and falling drop loops. We must resort to 6502 machine language code.

The source code shown was generated using the Toolkit Assembler (2) and used CHEMUTIL-2 to print the flame images contained in character set 2. To see this program in action you will need to enter the monitor (CALL-151) and enter the program directly. e.g. 7C10 : A9 OB 8D 00 7C A9 09 etc. Exit the monitor and BSAVE FLAME, A$7C00, L$8E then, with CHEMUTIL-2 in place, run the following BASIC program.

```
10 CALL 25042 : CALL 31760 : END
```

In the next part of this series, I will discuss and show a listing of a routine which shows a rotating stirrer bar suitable for combining with this flame sequence. Subsequent parts of this series will all require CHEMUTIL-2 for the successful execution of the code.

(1) Bendall, V. "CHEMUTIL-2, A chemistry Programming Utility"; Project SERAPHIM, NSFScience Education; Department of Chemistry, Eastern Michigan University, Ypsilanti, MI 48197, 1985.

(2) "Applesoft Tool Kit"; Apple Computer, Inc., 20525 Mariani Avenue, Cupertino, CA 95014.

BOOK REVIEW COLUMN

This issue's column includes books that cover a variety of computer topics, ranging from chip architecture to campus wide networking, and most readers should find at least one book that is of interest. As usual readers are invited (and urged) to express their opinions about this column. Whether you would like to review a book, suggest a topic area that doesn't receive enough coverage, or just make a general comment, write to Dr. Harry Pence, Professor of Chemistry, SUNY-Oneonta, Oneonta, NY 13820.

CAMPUS NETWORKING

STRATEGIES
Edited by Caroline Arms
Digital Press, Bedford, MA, 1988
336 pages, hardbound, $30.00
Reviewed by Harry E. Pence*

This second volume in EDUCOM's "Strategic Series on Information Technology" is similar in approach to its predecessor, Campus Computing Strategies (reviewed in the December, 1984 issue of the "Newsletter"). Ten institutions that are considered to be leaders in the implementation of computing technology are reviewed, each in an individual chapter, to assess their current status and future plans for computer networking. The universities included are Wesleyan University, Dartmouth College, Carnegie Mellon University, Rensselaer Polytechnic University, Massachusetts Institute of Technology, Stanford University, Cornell University, The University of Michigan, The University of Minnesota, and The Pennsylvania State University.

Even though these schools represent a variety of campus types, many readers of this Newsletter probably will not find a school in this list that could serve as a model for their own campuses. Pioneering on the scale described here requires both deep pockets and extensive prior expertise. Wesleyan University, the smallest institution included, has a combined graduate and undergraduate enrollment of 3,100 and consists of 118 buildings on a 120 acre campus. Many small undergraduate schools may find that their needs are adequately met by much simpler networking systems than any in these examples.

Each chapter is written by administrators who are closely concerned with the implementation of the campus network. The authors provide considerable technical detail without becoming too immersed for the reader with little previous networking background. Perhaps more important, their narratives seem to be unusually candid about the problems that have been encountered. Therefore, these presentations should be useful to a broad audience, networking experts, campus administrators, and users.

Perhaps the most important generalization that can be drawn from these various experiences is the need for careful planning and especially for oversight of the actual construction work. Many of these reports dramatize the problems of being on the "bleeding edge" of technology. Indeed, one author was moved to formulate the First Networking Aphorism, "The sooner you start, the longer it takes." Even though these campuses did possess considerable expertise in computing, many emphasized the importance of a good consultant. This advice was underscored by the reports that at least two of these institutions encountered significant problems with the contractors who installed their networks.

The last three chapters of the book offer concise, but very informative, discussions of protocols and standards, campus wiring, and national networks, three important topics that are frequently confusing to those who are just becoming interested in this networking. The glossary provided at the end will also be very useful for readers who are not yet familiar with this variety of computer jargon.

Exaggeration and hype are not uncommon in describing computer products, but the development of computer networking appears to have been accompanied by more than the average amount of hyperbole. It seems that almost every year for the past decade has been identified by some group or other as "The Year of the Network." As a result, it's been easy to overlook the steady progress that has really been made. This book realistically describes the current status of the field, both the problems and the potential, and so provides an excellent overview of an important area.

The case studies described in this volume indicate that campus networks are becoming a major education resource at many institutions; it seems likely that these changes will have effects far beyond the few institutions that are chosen for the case studies. Although this book will probably be most useful for those who are directly involved in the formulation of campus or department computing plans, it also provides some excellent perspective on the problems and potential advantages of computer networks. It should be considered for purchase on that basis.

*Chemistry Department
SUNY-Oneonta
Oneonta, NY 13820

PROGRAMMING THE 6502
by Rodney Zaks
SYBEX, Berkeley, CA
1983, 408 pages, paperbound, $17.95
Reviewed by William K. Nonidez*

As the new fast generation of microcomputers begin to invade our lives with their vastly enhanced computational and graphics capabilities, it is a temptation to shove our older systems over to a quiet part of the lab and quietly let them die. This may well be a blessing for those of us in education because, if we are clever, we can beg these machines from our more affluent colleges and use them in our own labs and classrooms to teach the fundamentals of computers and microprocessors.

Machines based upon the 6502 microprocessor are ideal for this purpose due to the relative simplicity of 6502 architecture and to the openness of most manufacturers in documenting the architecture of their machines. The most prevalent example of such an instrument is the Apple II series, which is shipped with a detailed reference manual which explains in quite readable detail how the instrument works and even includes detailed ROM listings and entry points. Although these manuals contain sections on the 6502 instruction set and brief descriptions of their structure, most of us can only make sense of this material after we have a sufficient background in the basics. I can strongly recommend Programming the 6502 for obtaining this background.

Programming the 6502 is divided into eleven chapters and nine appendices. Chapter 1 is the obligatory chapter on basic concepts which must be understood before continuing with the rest of the book. These topics include basic elements of programming, information representation and flowcharting. Zaks' treat-

ment of this topic makes the entire book worthwhile because (Wonder of Wonders!) he has included examples and exercises which appear in the text directly after the explanation. Since the answers and explanations appear at the end of the text you can read this chapter and test yourself on the concepts presented as you go.

Programming the 6502 requires a thorough understanding of how the various registers in the microprocessor are organized and exactly what is the function of each. Chapter 2 explains the organization of the 6502 and, in general terms, how it works in enough detail to make programming possible. The minute detail which would lead to a deeper understanding is mercifully omitted. Chapter 3 presents the fundamental techniques necessary for writing a machine language program. Additionally register management, loops and subroutines are discussed. The author uses very simple programs as examples, such as addition and subtraction in both binary and BCD mode, for this purpose as well as more complicated programs such as 16-bit multiplication and division. Only the most simple addressing is used in these programs since the concept is discussed in detail in a later chapter. Again the author makes use of highly instructive exercises to insure that the reader understands vital concepts as he moves through the text.

Chapter 4 is an exhaustive summary of the 6502 instruction set. The first part of the chapter classifies the instructions which follow into five classes and discusses each class in detail. Then each instruction is explained in detail as well as how they may affect flags or can be utilized with various modes of address. I would suggest that the reader go over this chapter the first time with the object of obtaining an overview of the instruction set since the comments about addressing and flags will make little sense at this time. Then as he proceeds through the book he can come back to this chapter for learning greater detail about instructions as needed.

Addressing techniques is the hardest topic, in my opinion, to be mastered in machine language programing. Zaks patiently goes through the various modes available on the 6502 with examples of each in Chapter 5. I found the use of diagrams called data paths extremely helpful in this section (once I figured them out). Again this is a chapter whose broad concepts should be appreciated on a first reading. During actual programming attempts the reader will have to return to this section many times to pick up the details necessary for successful program writing.

Chapters 6 and 7 are concerned with the problem of microprocessor communication with the external world. The 6502, since it is memory mapped, treats memory and I/O ports alike and considers both to be memory. In other words the 6502 does not have separate pins for a read and write signal as do the Z80 or 8080. A write operation to the proper memory location can serve as an output operation, and a read operation from the proper memory location can serve as an input operation. This feature makes the 6502 particularly simple to interface. Zaks discusses in Chapter 6 both the simple task of communication with common I/O devices as well as the more complicated task of dealing with several of these devices through interrupts. The reader should be cautioned here that various microcomputers which utilize the 6502 have been designed with various systems in ROM for handling interrupts. Once you understand how the 6502 handles interrupts it would then be well to see how this is implemented on your own particular machine.

Chapter 7 discusses interfacing hardware mainly the 6520 family of programmable input output chips. These chips are the mainstay of most simple I/O boards. The author refers the reader to the documentation supplied by the manufacturer of these chips for the details of their use. In my experience this will give the reader a valuable exercise in understanding technical jargon which can be at least for a while a most frustrating experience.

The remainder of the book is comprised of valuable chapters on application examples, data structures and program development which allow the reader to put into practice the principles he has presumably learned in the preceding chapters. These chapters are not only valuable to the 6502 programmer but of value to anyone interested in these topics.

One important development in 6502 technology which has occurred since the publication of this book is the introduction of the CMOS (complementary metal-oxide semiconductor) version of the 6502. This chip is pin and software compatible with the earlier versions with the added advantages of lower power consumption and an extended instruction set. Additionally this chip has been changed to iron out some of the minor anomalies which are poorly documented in most of the literature including Programming the 6502. I would suggest that anyone with a serious interest in 6502 programming read and keep an article in BYTE magazine called "The CMOS 6502" by Steven Hendrix (December, 1983) for a complete discussion of the CMOS 6502 as well as the various anomalies of the older version.

On the whole I found Programming the 6502 an excellent learning tool for 6502 machine language programming. A student armed with a good assembler and this book as well as a 6502 can quickly master most of the techniques necessary for successful program writing. As I pointed out in an earlier review in this newsletter, however, the art of programming is best learned by actually sitting down and writing a program. Mr. Zaks has made this chore easier.

*Assistant Professor of Chemistry
University of Alabama in Birmingham, University Station
Birmingham, Alabama 35294

ARTIFICIAL INTELLIGENCE APPLICATIONS IN CHEMISTRY

Edited by Thomas H. Pierce and Bruce A. Hohne American Chemical Society, Washington, DC, 1986, 395 pages, hardbound, $59.95.
Reviewed by Harry E. Pence*

After many years of steady progress, practical applications of

artificial intelligence techniques in Chemistry now seem to be burgeoning. Chemistry is said to be an ideal field for the application of artificial intelligence and expert systems, because there are many situations where decisions are made on the basis of past experience or a set of rules that has been empirically derived. Expert systems catalog these rules in a form that is accessible to a computer, then use this inventory of rules to make decisions.

This book consists of papers presented at a symposium held at the American Chemical Society Meeting at Chicago, Illinois in September of 1985. Like most of the volumes in this ACS series, the articles were submitted as camera-ready copy. The individual papers cross reference each other more than usual and the type is generally quite readable. The index provided seems quite adequate, and each paper includes a bibliography for further reading.

The various papers do a fine job of representing the diversity of effort in this field. The discussion includes not only those few programs that are already available commercially, but also many different types of software that are in various stages of development. Concern has been expressed about the large amounts of programming time required to bring a new expert system to market, and so the reports on the use of rule-making systems to decrease development time may be particularly significant. The use of a rule-maker simplifies the development process both by decreasing the time required and also by decreasing the complexity of the programming process. These systems are already playing an important role in AI development, and they will probably become more prominent as they become more available and better known.

This book will be of primary interest to those who are either currently engaged in chemical AI research or are planning to do so in the near future; however, as more of these systems become commercially available, it is increasingly important for chemistry instructors to be aware of the capabilities of this new technique. Those who are interested in maintaining contact with progress in this field may well find this to be a solid way of doing so.

*Chemistry Department SUNY Oneonta, Oneonta, NY 13820

(See front cover for resulting graphs of the program below.)

```
1   'PROGRAM PHTITRAT BY Dr.R.J.LANCASHIRE DEP'T OF CHEMISTRY UWI, JAMAICA.
100 GOTO 120
110 FOR W=1 TO 5000:NEXT:RETURN
120 CLS:COLOR 5,0
130 ' SET UP RANDOM CONC OF ACID (APPROX .06) AND BASE (APPROX .001)
140 FOR I=1 TO 10*RND(1):A=RND(1):NEXT
150 V1=.09+RND(1)/10:VT=V1
160 MB=.001+RND(1)/10000:MB%=MB*1E+06:MB=MB%/1E+06:MB1=MB/2
170 MA=.06+RND(1)/100:MA%=MA*10000:MA=MA%/10000:MA1=MA/1000
180 VF=500*MB/MA:VF%=VF*100:VF=VF%/100
190 ' IGNORES DILUTION SINCE ONLY 10 cm3 ADDED TO 500 cm3
200 LOCATE 22,4,0:PRINT USING"TITRATION OF ##.####";MA;:PRINT " M STRONG ACID VS
. ";
210 PRINT USING"500 cm3 OF ##.###^^^^";MB;:PRINT " M STRONG BASE";
220 'STAND + BASE
230 LINE(20,164)-(630,164)
240 LINE(100,5)-(100,163):LINE-(104,163):LINE-(104,5):LINE-(100,5)
250 LINE(90,163)-(114,163)
260 ' BURETTE
270 LINE(98,95)-(157,95),2
280 LINE(145,7+VT*10)-(155,7+VT*10),1
290 LINE(145,5)-(145,120):LINE-(148,127)
300 LINE(155,5)-(155,120):LINE-(152,127):LINE-(148,127)
310 LINE(138,110)-(160,112):LINE-(160,115):LINE-(138,117)
320 LINE-(138,110)
330 PAINT(150,11),1,5
340 PAINT(150,120),1,5
350 FOR I=0 TO 10
360 SYMBOL(155,5+I*10),"-"+STR$(I),1,1
370 NEXT
380 'BEAKER
390 LINE(110,115)-(120,110)
400 LINE-(120,162)
410 LINE-(240,162)
420 LINE-(240,110):LINE-(250,115)
430 LINE(120,140)-(240,140)
440 PAINT(122,142),1,5
450 ' STIRRER BAR
460 BAR$="<"+STRING$(3,219)+">"+STRING$(5,8)+" <"+CHR$(219)+"> "
```

```
470 X1=21:Y1=20:SY2=60
480 LOCATE Y1,X1,0:PRINT BAR$;
490 ' PH ELECTRODE
500 CIRCLE(220,148),6,.87,.65,.508,3
510 LINE(217,146)-(217,70),3:LINE-(224,70),3
520 LINE-(224,146),3
530 LINE(217,80)-(224,80),3
540 PAINT(220,148),4,3
550 LINE(224,85)-(300,85)
560 LINE(300,45)-(620,160),3,B
570 PAINT(310,50),1,3
580 LINE(330,135)-(575,135)
590 LINE(330,135)-(330,65)
600 IY=-2:FOR Y=128 TO 58 STEP -10:IY=IY+2 :SYMBOL(300,Y+3),STR$(IY),1,1:SYMBOL(
325,Y),"_",1,1:NEXT
610 SYMBOL(325,50),"pH",1,1
620 IX=-1:FOR X=327 TO 570 STEP 24:IX=IX+1:SYMBOL(X,135),"¦",1,1:SYMBOL(X-5,145)
,STR$(IX),1,1:NEXT
630 SYMBOL(590,135),"vol",1,1
640 LOCATE 1,46:PRINT "HOLD DOWN <T> TO TITRATE";
650 LOCATE 3,42:PRINT "TYPE <F> FOR FAST AND <S> FOR SLOW"
660 LOCATE 5,50:PRINT "TYPE  <E> TO END";
670 LINE(1,33)-(50,33):LINE-(50,50):LINE-(1,50):LINE-(1,33)
680 LOCATE 3,1:PRINT "INIT. VOL.";:LOCATE 6,2:PRINT USING "##.##";V1:LOCATE 6,8:
PRINT "cm":SYMBOL(77,36),"3",1,1
690 LINE(1,80)-(50,80):LINE-(50,97):LINE-(1,97):LINE-(1,80)
700 LOCATE 9,1:PRINT "CURR. VOL.":LOCATE 12,8:PRINT "cm":SYMBOL(77,83),"3",1,1
710 LOCATE 12,2:PRINT USING "##.##";VT;
720 Q$=INKEY$:LOCATE Y1,X1,0:PRINT BAR$;:LINE(150,127)-(150,139),0
730 IF Q$="F" THEN SY2=10:GOTO 720
740 IF Q$="S" THEN SY2=60:GOTO 720
750 IF Q$="E" THEN 840
755 'IF Q$="<" THEN 10000
760 IF Q$="*" THEN LOCATE 1,1:PRINT VF:LOCATE 1,1:PRINT SPC(5)
770 IF Q$<>"T" THEN 720
780 VT=VT+RND(1)/SY2:LOCATE Y1,X1,0:PRINT BAR$;
790 IF VT*10> 100 THEN 720
800 LINE(146,7+VT*10 )-(154,7+VT*10 ),0:LINE(150,127)-(150,139),2
810 VT1=VT-V1:CCC=MB1-VT1*MA1:CCB=(7+7*SGN(CCC))+SGN(CCC)*LOG(ABS(CCC))/2.303
820 IF CCC=0 THEN PSET(330+VT1*24,105) ELSE PSET(330+VT1*24,135-5*CCB)
830 GOTO 710
840 LOCATE 22,1:PRINT SPC(79)
850 LOCATE 22,1:PRINT "TITRATION STOPPED"
860 LOCATE 22,30:PRINT SPC(100)
870 LOCATE 22,30:INPUT "ENTER THE END POINT (VOLUME) ";T1
880 IF T1 > VF + .15 THEN LOCATE 23,1:PRINT "YOUR VALUE IS TOO HIGH";:GOSUB 110
:GOTO 860
890 IF T1 < VF - .15 THEN LOCATE 23,1:PRINT "YOUR VALUE IS TOO LOW";:GOSUB 110:G
OTO 860
900 IF T1 = VF THEN LOCATE 23,1:PRINT "SPOT ON -CONGRATULATIONS":GOTO 920
910 IF T1>VF-.15 AND T1<VF+.15 THEN LOCATE 23,1:PRINT "WELL DONE";
920 LOCATE 23,30:PRINT "THE COMPUTER HAD A VALUE OF ";VF
930 GOSUB 110:LOCATE 22,30:PRINT SPC(130)
940 LOCATE 23,1:PRINT "TRY AGAIN <Y/N> ?";
950 Q$=INKEY$:IF Q$="" THEN 950
960 IF Q$="N" THEN END
970 IF Q$<> "Y" THEN 950 ELSE 120
```

NOTICE: The recent election for chair of the Committee On Computers in Chemical Education resulted in a tie. Let's welcome our new cochairs: Patricia Flath of Paul Smith College, NY, and Alfred Alta, University of Kansas.