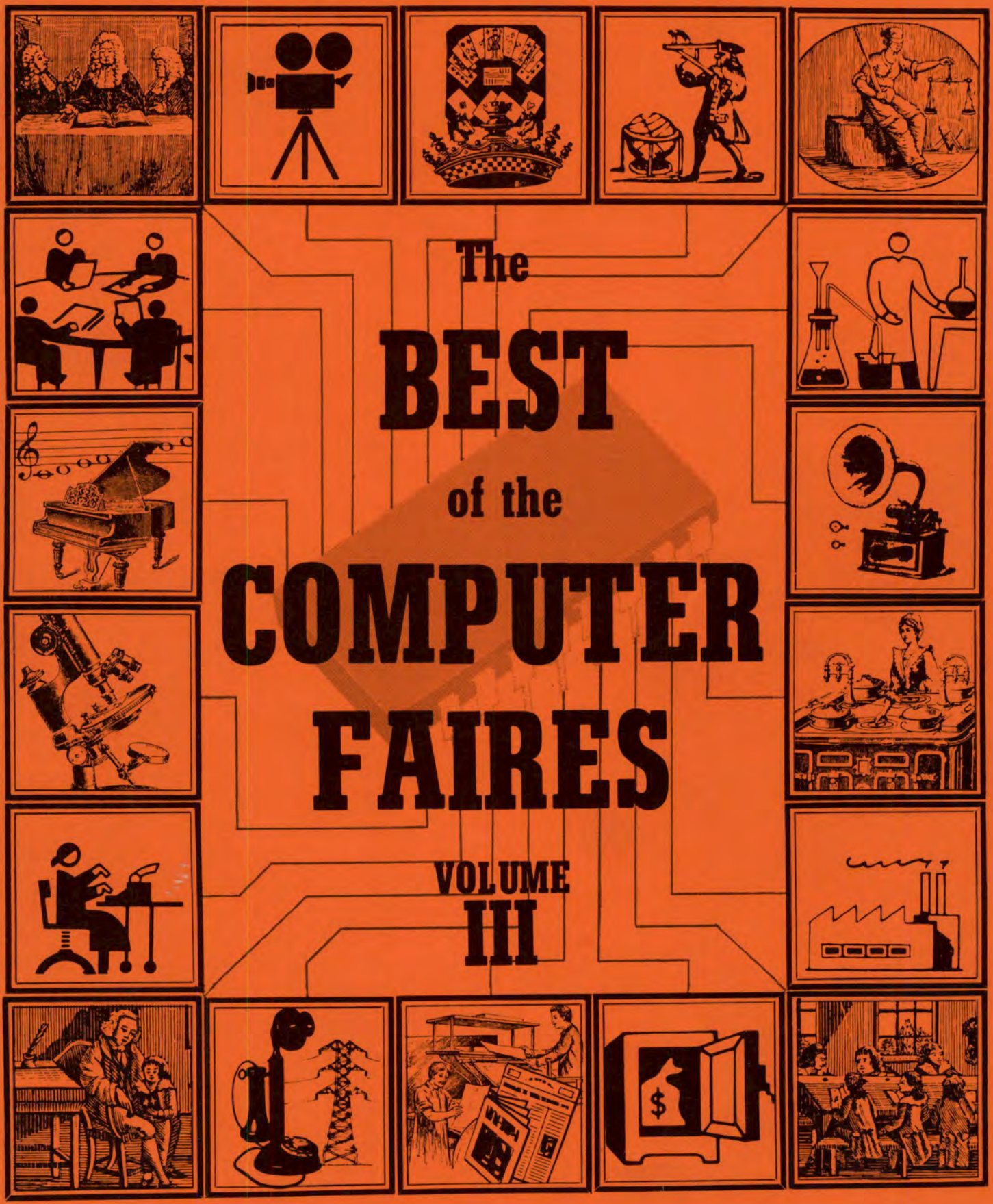
The Best of the Computer Faires
Vol. III



The BEST of the COMPUTER FAIRES

VOLUME III

CONFERENCE PROCEEDINGS OF THE THIRD WEST COAST COMPUTER FAIRE

Los Angeles, California

THE BEST OF THE COMPUTER FAIRES, VOLUME 3:

Conference Proceedings of the 3rd West Coast Computer Faire

Jim C. Warren, Jr., Editor

The Third West Coast Computer Faire
held in
The Los Angeles Convention Center
in
Los Angeles, California
November 3 - 5, 1978

COMPUTER FAIRE
Box 1579
Palo Alto CA 94302
(415) 851-7075

ISBN 0-930418-02-X

Library of Congress Catalog card number 78-71092

These *Proceedings* were constructed with the gracious assistance of Marguerite Brosing and Bill Baumann of Computer Faire; Nancy Hamilton, Dave Brown, Gary Markesen, Jawn Pope, and helpful others at Nowels Publications, Menlo Park, California; and Toby Forshee and friends of Redwood Trade Bindery, Redwood City, California. If you received this *Proceedings* through our mail order department, then it additionally received the adoring attention of Faun Jackson, Lygia Brosing, and Steven Reiling. ...and last, and most, this *Proceedings* would not have been without the guidance, both spiritual and otherwise, of Elmer Malloy.

PREFACE

Less than 18 months ago, the First West Coast Computer Faire took place in San Francisco. Drawing almost 13,000 people, it offered a hodge-podge of almost 100 speakers and 200 exhibits. The speakers were addressing various aspects of a brand new topic – hobby computing: an environment in which computers were so low-priced that they had become a specialty consumer item. The exhibitors were, for the most part, new, small companies – more often than not formed by one or two highly competent, innovative engineers or programmers who had considerable technical capability and creativity, but little or no business or manufacturing experience.

Less than a year later, the Second West Coast Computer Faire was held in San Jose, drawing a similar number of speakers and exhibitors, and over 14,000 attendees. Things had changed considerably, in the ten and a half month interval between the Faires. For instance, in the first Faire, several ideas or proposals were presented for possible uses of home computers. In the second Faire, actual implementations of some of those proposals were detailed. The exhibitors and their products had shown a major change for the better. Their products were more reliable, more varied, better manufactured and – to some extent – more available. Additionally, both the Conference Program and the Exposition illustrated something of a shift from a pure hobbyist, computer fanatic orientation towards the broader topics of consumer computing and general low-cost computing power for a large variety of applications – many of which had not previously been economically or technologically viable, e.g., aids for the physically impaired, individual computers for students, electronic music systems, etc.

Now, the Third West Coast Computer Faire is to take place next month, in Southern California for the first time. It appears likely that there will be an increase over the two previous Faires, both in number of exhibits and in the number of attendees. And, the Conference Program will illustrate that the marketplace and applications of truly low-cost computing is continuing to change and mature.

Although these *Proceedings* include sections addressing such things as computer games and music – hobbyist and entertainment applications – the vast majority of the Conference Program focuses on more "serious" applications of very low-cost computing. For the first time, we see a major Conference Section addressing the application of microcomputers to biomedical and hospital environments. Other sections appearing for the first time address the use of inexpensive computers to improve local government services, and examine business systems software on micros. The sections addressing computers in education, legal aspects of computing, and computers and communications illustrate a continuing maturing process in those fields, relative to the papers addressing the same topics, presented in the previous Faires. The software sections offer some major proposals for much-needed improvements in microcomputer software – i.e. better high-level languages, and proposals for floating point standards that have been well thought out by highly competent professionals.

Two other sections are particularly noteworthy: The "Potpourri" section – papers that defied classification under traditional computer topics – has grown in comparison to past Faires, and shows a greater variety of unusual and exotic views and proposals. Finally, the section addressing "Visions of the Future" explicitly addresses the topic of probable futures, providing projections that are much more viable and likely than we could reasonably foresee, a year and a half or two years ago.

The Computer Faire, from the beginning, has had a major commitment to conducting a significant and useful Conference Program – and to the production of *Conference Proceedings* from each of these Programs. We feel that they are much-needed and uniquely valuable communication and reference publications in this explosively changing field – intelligent machines for the general public.

Jim C. Warren, Jr., Faire Chair
Woodside, California
78 Oct 9

TABLE OF CONTENTS

YOU DON'T HAVE TO BE "GOOD IN MATH" TO FALL IN LOVE WITH COMPUTERS

Donna Norris
31 Tallman Place, Nyack, NY 10960
914/358-7485

Just over a year ago I knew only random facts about large computer systems and absolutely nothing about microcomputers.

Today I'm going to tell you what I did to get involved with computers, recommend some materials that I found particularly helpful, outline the computer literacy course I wrote and taught to students in New York (this project justified buying my computer), make a few recommendations about setting up classes in your community, and hopefully dispel some myths you may have heard about computers and the people who involve themselves with computers. You don't have to be "good in math" to fall in love with computers.

A phone call from a business acquaintance in August of 1977 asking if I would teach a computer literacy course to a group of gifted fifth and sixth graders prompted me to get started.

I had been a teacher in Los Angeles but my knowledge about computers was frankly limited to three facts: a computer system has five parts, a byte of information contains eight bits and garbage in garbage out. These assorted and unlinked facts had been among the least essential things taught in an assembler language course I had taken ten years before and were the only things I remembered.

Within three months I was an expert in some other's eyes. Within six months I had become an adviser to a friend in Oregon who was setting up a computer project at his children's school. And now fifteen months later, I have joined those who present papers at the Computer Faire.

I began my search for materials about computers on the shelves in our attic. All three of the books I found were ten years old and were related to computer assisted instruction. Don't bother to search in your attic. Books two or more years old are already nearly obsolete. And if you, too, should find copious notes from an assembler language course, discard them.

Reader's Guide to Periodical Literature proved to be more fruitful. It lead me to some recent articles about computers and their applications now and in the future. Secure some recent articles. They are generally easy to read and understand and will give you up-to-date information about computers today and predictions about their use tomorrow. I find the news exciting.

Next I visited some computer stores. I have decided they are not the place to learn about computers. I can only compare the atmosphere to that one might experience as a female entering a country store and attempting to ask questions about the subject being discussed by the group seated on cracker barrels around the pot-bellied stove. But go to computer stores to see small computers and to buy books and magazines.

I bought all the books and magazines that were available in August of 1977. It was not a large investment but there are many more in print today. People's Computers and Calculators Computers magazines are recommended for novices who don't have a science or math background. James White's book Your Home Computer, published in 1977 by Dymax, is my favorite among the many choices in books.

Finally I placed an order at our local bookstore for all the "in print" books that had been written for children about computers. These books, even the old ones, provided a basic foundation for understanding more advanced material. I have since used them as research materials for the students in the computer literacy courses. My three favorites are: What is a Computer?, Marion J. Ball, Houghton Mifflin Company, 1972; Computers, Jan Jonas Srivastave,

Thomas Y. Crowell, 1972; Computers--Tools for Today, Claude J. DeRossi, Regensteiner Publishing Enterprises, 1972. Perhaps you will be more fortunate than I and find them in your local children's library.

At this point I began to feel quite confident about my ability to communicate intelligently about computers, but the larger task of writing a computer literacy course still lay ahead. So I decided to use a plan I had devised when I planted my first herb garden and produced my first animated film. Find a child to work with you--yours or somebody else's. The youngsters I'd worked with before brought enthusiasm, interest and perseverance to each project and George was no exception. I was forced to accomplish a great deal more and in a shorter period of time than I would have working alone.

Here's what we learned and decided I'd teach to the students in the first six sessions. They were written to be used without a computer because frankly at the time we could only dream about using one in the course. Happily the dream came true!

SESSION 1: GENERALLY SPEAKING

Introduction--Find out what your students know about computers and why they're interested to learn more. Here is the place to eliminate the TV myths. Computers that carry on intelligent, two-way conversations in audible English and those that literally blow up buildings are the most common.

1. Teach them what a computer is and what it can be expected to do.

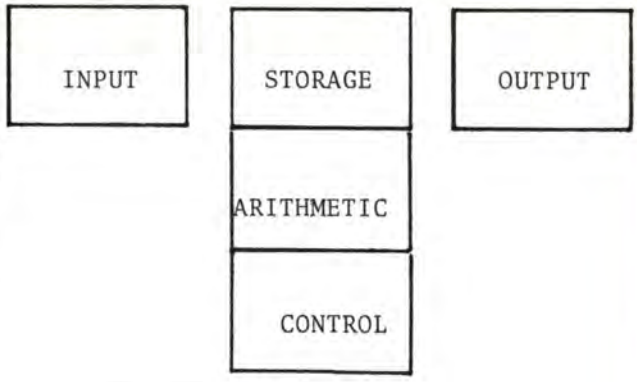
Use pictures of computers and computer systems to show various types that are in use.

Use computer generated credit card bills to illustrate some of the specific things computers can do--read, store, add, subtract, multiply, divide, follow instructions, compare and sort, count and print.

SESSION 2: COMPUTER PARTS

1. Teach the students about the parts of a computer, what each part does and how the parts work together.

Use a computer diagram on the chalkboard or constructed on a bulletin board to illustrate the parts of a computer system and how data and instructions move through the system.



CENTRAL PROCESSING UNIT

SESSION 3: IP/OP

1. Teach the students about input and output units and the materials they require.

Use pictures of various types of units.

Use sets of blank and coded Hollerith cards to illustrate one type of input and output material. The students can code in their own messages on the blank card with a small nail or pin.

SESSION 4: STORAGE

1. Teach the students about different types of storage or memory units that are commonly used in a computer system.

Use pictures or samples to illustrate some of the types--magnetic disks, chips, floppy disks, magnetic cores, magnetic bubbles, magnetic tapes, Hollerith cards.

2. Teach them how information is stored in a computer.

Use a series of lessons called "Bits and Bytes" written by Bob Albrecht. They appeared in Calculators Computers Magazine in the October, 1977 through March, 1978 issues.

Use an electromagnet to clarify the concept of magnetized (on) and not magnetized (off) bits within the computer storage unit.

SESSION 5: PREPROGRAMMING

Teach the students how to analyze problems they want the computer to solve.

Use the following steps and examples (or use your own):

1. Define the problem.

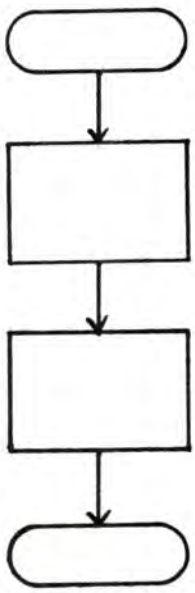
   Tim received a shipment of 999 cages. Gretchen received a shipment of 1001. How many did they receive together?

2. Develop an algorithm or step-by-step procedure for solving it.

   Add 999 and 1001.

   Write the sum.

3. Draw a flow chart to show each step.



By early October, two small $600 computers had been announced--the TRS-80 from Radio Shack and the PET from Commodore. I was convinced that I wanted one for myself and needed one for the course. Since reviews had not yet been published on either machine, I went to the computer show at the New York Coliseum and visited both company's booths. The demonstration pitches sounded similar. Both were filled with abbreviated words like RAM, ROM and 4K, and unabbreviated words like interface, bus, buffer and port. Lacking the indepth knowledge that I needed to make an intelligent comparative choice based on these hardware specifications, I asked about service. The Radio Shack representative said it could be done through their stores. The Commodore representative said plans hadn't yet been worked out. This situation has since changed and I tell the story only to avoid having my choice of a TRS-80 construed as an endorsement of it or a criticism of the PET. If you have only $600 to $800 to spend and want a machine to help you learn about microcomputers or to teach others about them, either machine will serve your purpose.

The machine arrived in late January and the last five sessions of the computer literacy course were designed to give the students an opportunity to have some "hands on" experience with the computer. These were the most exciting sessions to plan and teach but I am convinced this was primarily due to the fact that the students and I had the background of the first five sessions.

If you can't buy a computer, your local librarian, real estate agent, or whoever is the best source of information in your community can probably help you find someone who has one. An article by Joann Verplank in the March-April, 1977 issue of People's Computer Company was entitled, "Have Computer, Will Travel." A person with that attitude would suit your needs perfectly.

If you want to buy a computer for under $1,000, visit the Commodore and Radio Shack booths and stores or distributors and try their computers. Obtain promotional materials and copies of recent issues of Calculators Computers, Personal Computing, Creative Computing, and People's Computers which have reviews of both.

Games on tapes are available from both computer manufacturers, as well as from many small companies. I have particularly enjoyed "Stimulating Simulations" available from Personal Software in Cambridge, Mass.

Several monthly magazines and newsletters specifically designed for PET or TRS-80 owners are currently being published. Some print their articles and programs on cassettes

which can only be read on your computer. Others print complete program listings which you can enter into your computer and copy on your own tape.

User's groups have also formed in many parts of the country. Membership requirements vary as do benefits.

Information about games, magazines, newsletters and user's groups can be found in the computer magazines.

If you want to learn Level I BASIC for use with the TRS-80, use Radio Shack's User's Manual for Level 1. COMMODORE BASIC for use with the PET is being featured in a series of articles beginning with the February, 1978 issue in Calculator Computer magazine.

Using many of these suggested publications as resource materials, here's what we decided I'd teach in the last five sessions of the computer literacy course using a TRS-80 computer.

SESSION 6: WRITING PROGRAMS IN BASIC

1. Teach the students how to write a program for the problem they analyzed during session five.

2. Teach the students how to write a program to solve the problem A+B.

SESSION 7: RUNNING PROGRAMS

1. Demonstrate the use of the computer by entering and running the programs you wrote in Session 6.

2. Teach the students how to write a program to solve a problem of the following type. Run it on the computer.

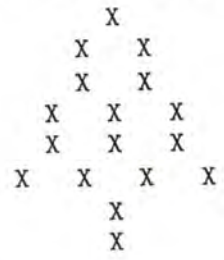
As you recall, Tim and Gretchen have 2000 cages. They want to find a cricket to put in each cage. They get eight friends to help them. Five teams of two are formed. Each team is asked to find a minimum of 200 crickets within four hours. When the teams return with their crickets we learn that Team A found 56, Team B 346, Team C 23, Team D found 200 and Team E.198. The program must tell the computer to print the number of crickets each team found and whether or not they exceeded their quota or missed their quota of 200.

3. Teach the students how to change the program so it will print out whether each team exceeded, missed or met their quota.

SESSION 8

Teach the students how to write a program for a graph paper design. Let them run their programs on the computer.

Use a tree design as an example.



Demonstrate the use of a Go To statement in a graphic program. It creates a constantly moving design on the screen.

And, if you and they are ready, change the program by introducing FOR-NEXT. These limit the number of times each line in the design is printed.

SESSIONS 9 and 10

Demonstrate some games and/or simulations for the students and let them play in teams.

I've presented the complete set of classes four times--twice in a public school to gifted students in grades five and six; once at a summer library program to a group of sixth, seventh and eighth graders who responded to an announcement which appeared in the local newspaper; and currently to a group of 4th, 5th and 6th graders on Saturday mornings at our community library. Other youngsters from our neighborhood have appeared at the door asking to see my wonderful machine (communication travels fast) and have spent whole days exploring the use of the computer on their own. There's an idea some of the young people may be able to use!

If you're planning to run a series of lessons there are a few recommendations:

Keep the group small. 8 or 10 is the

maximum, and limit the age range.

All students can be taught basic programming, but only those with good basic math and reading skills will be able to successfully use most of the educational games that are currently on the market.

Work with gifted kids only if you're self-confident and are not easily intimidated. They're challenging and wonderful.

Find someone who can be called upon to answer technical questions or supply other information requested by the students that is beyond your level of expertise.

Be relaxed about having youngsters use your computer. Those who have used mine have shown great respect and it does not appear to have suffered from overuse.

The steps I took to get involved with computers are outlined for you. They can be followed by anyone. You don't need much money, particularly if you're resourceful, you don't need a math background, and you don't need special skills.

Once involved, I hope you will plan to take your knowledge and your computer into your community and share it with others, particularly youngsters. I am convinced that computer literacy will soon join reading, writing and arithmetic as a basic survival skill in our technological age. Your participation in helping young people prepare for their future should help justify your need for a computer in all but your harshest critic's eyes.

An Introduction To Personal Computing : A Beginners Guide

Bob Moody - Mike Triolo - Jerry Fox
2233 El Camino Real, Palo Alto, Ca. 94306

Introduction

Computers are now personal! Whoever you are, whatever you do - whether housewife, small businessman, student, professional, musician, or real estate salesperson - computers are being made and sold at prices you can afford for applications beneficial to you. A personal computer will open possibilities that will almost certainly change your lifestyle at work and play.

You may ask yourself,"What is a personal computer? The only computers I've ever seen have been huge. How can anyone afford to put such an expensive piece of equipment in their home, and where would I put it?". Almost everyone's first impression of a computer is of a monstrous machine, owned only by wealthy businesses, that towers over them, overseeing all and doing all. This may have been an accurate impression 10 to 15 years ago, but not today. The technology that has been developed over the past few years has not only reduced the physical size of computers, but has lowered computer prices tremendously.

You have been in direct contact with computers in one way or another most of your life without knowing it. For example, many school records and schedules are stored in computers. Your family or business tax returns are processed by a computer. Most major department stores handle their buying stocking and billing by computer. Many newspapers and magazines are set in type under computer control. All credit card purchases are handled by computer. Every check written is processed by computer. Dentist, doctor, hospital, gas, and phone bills are handled by computer. The reasons are obvious.

In a city the size of San Francisco, for instance, it would take a tremendous amount of manpower to process all the checks San Francisco banks receive in one day, or to log the number and duration of all phone calls made in one day. Computing power is needed to handle that amount of information, and that kind of computing power is at your fingertips today.

This presentation is principally for two reasons: (1) to bring a little light into misunderstandings about computers and (2) to show you show personal computers can help you in your everyday lives.

A Brief History

The first digital computer, ENIAC, was built at the University of Pennsylvania in 1946. It filled a large room (1,500 square feet), and weighed 30 tons. ENIAC used vacuum tubes (18,500) to store ten-digit numbers. The tubes were so short-lived, ENIAC would not run for more than seven or eight minutes without a tube failure. Special vacuum tubes were developed to cut down its tremendous power requirements(130,000 watts), so the machine could run at a cooler temperature. After constant and deliberate work,ENIAC would run for several days with out tube failure.

Constant improvement over the years made ENIAC a very useful machine. The U.S. census of 1950 was possible largely because of ENIAC, which was retired in 1955. It took tremendous amount of knowledge to operate and talk to ENIAC and its early relatives. ENVAC, IAS, UNIVAC, and the early IBM machines. Engineers and scientists labored for months to understand and implement their usefulness.

The ENIAC computer cost more than half a million dollars (that's 1950 dollars!) to develop. Today you can buy the same computing power for under $1000 (that's 1978 dollars).

Through constant improvement these early computers grew into the most influential tool of the twentieth century. The advancement of many important fields could not have come about without them. Medicine, comunications, space exploration, travel, and mass merchandizing (to mention just a few) could not have come as far as they have without computer technology.

The Best Software Buzz Words

The first major hurdle you have to face in home computing is learning the lingo. As a group, computer people have probably abused the English language more than those in any other industry. Our specialized language had overgrown, trying to distinguish word by word, to the point where English speaking people have great difficulty deciphering our apparent "gibberish". If it were up to me I would like to start all over again and use terms that everyone would be able to inderstand and not have to adapt to a complete new way of conversing. I can't do that, so all you folks who are reading this book to gain an insight as to what personal computers are all about will have to forget the accepted definitions of some of the words that you have learned in school and take up Computerse. It's not going to be that difficult to do - just deprogram your brain and THINK COMPUTER.

The first computer terms we will deal with describe the way we talk to our computers. Software, as it is called, is the list of instructions we give to the computer to do a job. It is also the way the computer takes this list and converts it to the language it knows.
Bit=The smallest unit of measure in a computer word; several bits make up a byte or a computer word.
Byte= The space which a letter or digit (one character) takes up in a computer. Space in a computer is measured in bytes. A megabyte is a million bytes.
Ascii= American Standard Code for Information Interchange: this is a seven-bit code that defines letters and numbers that the computer uses as its input.
Alphanumeric= Information that is made up of letters (the alphsbet) and digits (numbers).
Input= The information that goes IN to your computer system. The computers food.
Program= A list of instructions to the computer telling it what to do and when to do it.
Data= The information that gets WORKED OVER when your program runs. Data is all the information you have your computer use, everything that you put into your computer to store and retrieve.
Output= Using the program you have put into your computer as the instructions and the data for information, the output is the finished product your computer system produces.
Basis= Beginers All-Purpose Symbolic Instructional Code. A "Language" that you will use to write your programs for your computer, using English words and phrases.
Fortran= Formuls Translation: another computer language used in programming that is composed of English words and symbols for instructions.

Programming

The best way to start this chapter off is by defining the term program. A program is a list of instructions that we give a computer so it can do a specific job for us. Though there are certain rules that we have to follow, it's no more difficult to write or follow a computer program than it is to bake a cake from scratch ingredients. Both take time and practice to be done correctly, but they can be done.
The computer is a machine that has been designed to know two things, on or off. However simple this may sound, it is true. Knowing that a combination of circuits is either on or off, enables the computer to "recognize" letters and numbers.
Programs contain strange and familiar words (and familiar numbers). Let's break down what we in the computer industry call a word. First of all, the name is changed. It's not a word but a byte, and it's not a complete word like those you are reading, but a single character or number.

From Ascii to Basic

The normal alphabet as we know it has to be broken down to a simple code so a computer can understand it. This code is called Ascii character set. Ascii stands for American Standard Code for Information Interchange. This code plays a very important part in the role of conversing with a computer. Because it is a standard, we can safely say that all digital computers use the Ascii code to communicate with humans.
The Ascii code is made up of seven bits. These one's and zero's, when in certain combinations of seven, will make up our English alphaber and numbers. Being computer people and tending towards being difficult at times, we have added one small detail to this code: an eight bit. This one or zero is placed in front of the other seven and called a parity bit. This parity bit allows the computer to double check the information we have put in. Depending on how the computer and the sending device are set up, The sending device chooses the eigth bit so that the total number of ones in the byte is even (if it is using even parity) or odd (if it is using odd parity). Once the computer has received the byte from the sending device, it can check to see of the parity is the same as when the byte was sent. If not, an error occured in transmitting the byte or character from the sending device to the computer. If all this has confused you, don't be alarmed, you really don't need to know it anyway. Just think of a parity bit as a way the computer checks to see of the character received is the one we sent. That's all!
Getting back to bits and bytes! The Ascii code, which you will hear about throughout this book, is listed in the back of this book in the appendix in a table that shows the alphanumerics (alphabet and numbers 0 through 9) and the symbols that we will use in programming.
Having a special code that the computer understands isn't enough for us computer people to confuse you. What we have also done is make it necessary to learn a special language using our standard alphabet to talk to computers. No! It's not Southern Mongolian, but something a little easier than English.
The language that most home computers use is called BASIC BEGINNERS ALL-PURPOSE SYMBOLIC INSTRUCTIONAL CODE is what BASIC stands for. This home computer language is designed as an interpreter. It is interpretive in that when we write a program in Basic it is done in a sequence of lines. Each line is given a line number, and when we give the program to the computer to run, the instructions that we put in those lines

are carried out in order of their line numbers, unless the program that we wrote directs otherwise.

Getting Down to Basic

This first program that I am going to show you was probably written by someone who wanted to see something on a terminal* screen rather than have it blank. The program looks like this:

```
10  REM: This is a dumb dumb program
20  Print "Help!! I'm stuck inside this box"
30  Go to 20
```

Let's examine this program and see what it will do. The first statement starts off with the line number 10, that's simple, but REM must stand for something. It is a REM ark. REMarks in Basic are used to make comments in the program to let other programmers know what we are doing, and to remind ourselves what's going on in any spedi l area of the program. OK! The second statement (line number 20) is a command -PRINT- that tells the computer to print on its communication device or terminal to us the message "Help!! I'm Stuck inside this box". The third program statement (line number 30) has the computer, after it is finished with the second command, go back to line 20 and, therefore, repeat the command over again. The result at the terminal would be this:

Help!! I'M stuck inside this box.
Help!! I'm stuck inside this box.
Help!! I'm stuck inside this box.
Over and over and over again.

Now this doesn't seem too exciting, but to the passerby, if he didn't know how to stop the program with a special command, nothing he typed into the terminal would stop the damned thing from printing out "Help!! I'm stuck inside this box," forever. These two commands combined (20 & 30) constitute what we call a loop. Starting with printing "Help!! I'm stuck inside the box" and going back and doing it over again loops us back into a previously done portion of the program. The reason that the line numbers are increased by 10 is because if we want to add any commands at a later date we have space to put them in. Otherwise, if we used line numbers like, 1,2,3, we wouldn't have any in between to add things on. (The computer doesn't recognize half numbers like 2½ or 2 3 4).

Let us put the computer to a little more difficult task, like counting from 1 to 20. This program has a few more commands in it, but don't let it scare you. We'll take it a few steps at a time.

```
10 REM: A program to count from 1 to 20
20 PRINT "Type in the number one".
30 INPUT N
```

*A terminal is a type of sending device used to talk to a computer. It will be described further in another chapter.

Let's look at the first three steps before we add more. The first command is to give us a remark, the second command will have the computer print out "Type in the number one". This is actually going to be a prompt for the user to respond to. The computer will be asking us a question. The semicolon at the end is a special character that the computer uses to type a response on the same line that it printed the statement. The next command, "INPUT N", is the response that the user types in (The "N"). It's what the computer will use as data. Oh! By the way, the question mark at the end of the print statement was put there by the computer. It did this automatically because it knows that the user has to type something in and it is the way it asks us to do so.

The significance of the letter N, in the third command, is nothing special; we can use any letter we wish. Let's add a few more statements to the program.

```
40  LET N = N +1
50  PRINT N
60  IF N  20 THEN 40
70  PRINT "Finished".
80  End
```

Line 40 gets us into a little bit of math. What I want the computer to do is take the number that the user types in and add a 1 to it, very simple. Here's our friend the PRINT statement again. "PRINT N" will have the computer show us the value of our original input plus 1. Let's push on. In line 60 we come to a very common statement that is used in the programming language Basic a lot IF-THEN. The IF-THEN is used in this cade to check and see IF the number that was typed in at step number 30 plus 1 (that's what we did in step 40, add a 1 to it) is less than(<) 20. < is a special character that is used to represent "less than". The second part of the statement is THEN 40. What I mean is to tell the computer to go back to step 40 and do it again, if the number (N = 1) is less than 20.

This will keep going on until we got to the number 21. Why 21? Because, if we look at the statement in line 60:

```
60  IF N< 20 THEN 40
```

We see that at N + 1=21, N is more than 20 and the program moves to the next step, line number 70, which is our friend the PRINT statement. At line 70 the computer will tell us that it is finished. Line 80 lets programmers know we are at the end of our program. Lte's put the whole program together and see how it looks.

```
10  REM: A program to count from 1 to 20
20  PRINT "Type in the number one".
30  INPUT N
40 LET N = N+1
50  PRINT N
60  IF F  20 THEN 40
70  PRINT "Finished"
80  End
```

When you first start out programming, it pays off to design a flow-chart diagram to give you a visual conception of how the program works. You can see the different stages in building the program and its paths to the end.
Let's add a different ending to our program, to jazz it up a little:

```
72  PRINT "If you want to run again typeY"
74  INPUT A
76  IF A ="Y" THEN 20
78  PRINT"Boy!This sure is dumb!"
```

OK! See if you can follow along on this one. Line 72 brings back the PRINT statement asking if we want the computer to run our program again. It will wait for an answer. In 74 we repeat the input routine, using a different letter. 76 is an IF-THEN again, but looping us back to the beginning of the program at line 20. Line number 78 is self-explanatory. We'll list the whole program and see how much we have added:

```
10  REM: A Program to count from 1 to 20
20  PRINT "Type in the number one"
30  INPUT N
40  LET N = N+1
50  PRINT N
60  IF N   20 THEN 40
70  PRINT "Finished"
72  PRINT"If you want to run again type Y"
74  INPUT A
76  IF A= "Y" then 20
78  PRINT "Boy!This sure is dumb!"
80  End
```

Our forethought paid off when we see the whole program listed. If we hadn't numbered our lines with the spacing of 10 we would not have been able to type in these extra commands. OK! I'll run the program on paper here and show you how it would look on a terminal:

```
Run
Type in the number 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Finished
If you want to run it again type in a Y? N
Boy! This sure is Dumb!
```

Hardware Bizz Words

Now that we have gone through the basics of how we talk to personal computers, we will next deal with the equipment itself. Hardware refers to the parts of a computer system we can actually put our hands on. Much like a stereo system, a computer system is made up offdifferent modules or parts In a basic stereo system you need a tuner, receiver, amplifiers, and speakers. Each working independently cannot produce music but all working together will give you the desired result.
Of course, when we talk basics in any system we can always make it bigger and better. You can add things like casserre decks and turntables to have your stereo system do more. In a computer system we can do the same thing. Most of all I need to define the terms we will be dealing with. So put your training hat back on & I will go through some basic hardware terms.
Circuit Board - A special board on which specific circuits have been etched, or "printed".The basic material is a thin sheet of insulating material that a layer of copper has been bonded to, then a layer of tin goes next. By using a photographic technique areas of the tin and copper are etched away, leaving thin lines called traces.

Motherboard = A printed circuit board with a specific bus structure that many different kinds of printed circuit boards can be plugged into.
Bus = A wire that carries an electrical signal to two or more printed circuit boards. A bus structure would by many bus lines that carry many different signals.

Micro.Microcomputer,Microprocessor = These terms mean the same as the CIU. "Micro" came into being when we learned to make things small. The circuit itself is about the size of a pencil eraser.
CPU = The Central Processing Unit is the brain of the computer, the part of the computer that thinks.

Ram = Random Access Memory: Temporary storage used by your computer to allow you to change the data you have put into it, which is then transferred to a mass storage device.

Rom = Read Only Memory, This is memory also, but it has been programmed to give out the same kind of information all the time. It never forgets and cannot be changed.

Eprom= Much like Rom, Eprom(Erasable Programmable Read Only Memory) does not forget but has one added feature. You can erase the information that you have programmed in it by using an ultra-violet light and reprogram with different information.

Interface or I/O = A connector that "Translates" between two parts of a system. You generally need one interface for each peripheral that you hook to your computer.

Serial I/O = This type of interface transmits and receives data to and from the CPU through a single stream of electrical signals. Serial transmission is used when the requirements to hook up the peripherals to your CPU must travel through low-quality long distance channels.

Parallel I/O = This type of interface transmits and receives data on a batch mode, where there are several signals interchanged at the same time. These kinds of transfers can happen very fast, but can't be sent too far.

Buffer = This is a holding area for information to equalize or balance two different operating speeds. For example, the output of the computer might be too fast for a printer, so the information would be held in a buffer until the printer could accept the next batch of data to be printed.

Peripherals = The divices attached to your computer, such as the display keyboard, printer, etc.

Mass Storage = Any way of keeping a lot of information OUTSIDE your computer, but available to it. This is your computer's memory. Most common kinds of mass storage are tape and disk.

Cassette Storage = In a personal computer we can use an ordinary auto cassette player to store information through a special interface.

Floppy Disk = A mass storage device that uses a flesible platter to store a large amount of information.

Hard Disk = Much like a floppy , a hard disk stores a tremendous amount of information, but its platter is much larger and not so portable.

Terminal = A unit for conversing (input or output) with your computer. It has a key board, plus display or print-out.

Tvt = Television typewriter. A Keyboard and electronics specially designed to turn your TV into a terminal.

CRT= Your computers TV screen showing you what's IN there. The CRT (cathode-ray tube) is your computers way of talking to you. It is also referred to as a display unit or terminal.

Bug = The cause of a malfunction, usually in a program. They're called "bugs" because they can be hard to find.

The Central Processing Unit (CPU)

The first and most important module is the CPU or CENTRAL PROCESSING UNIT. This is the brain of a computing system. The CPU does all the actual work: Calculation data alteration, and data reception and transmission.
There are a wide variety of CPU'S on the marketplace. Depending on who you talk to, one is always better than the other. Don't let this confuse you now. All you really need to know is that all CPU's function basically the same. We will get into the particular types later.
The CPU, which is sometimes called the controller, is an LSI circuit. large Scale Integration (LSI) is the process of putting thousands of components in a very small area. To give you an example using LSI technology, a dot this size,. could contain over 1000 transistors. Hard to believe, but true. Being able to reduce the physical size of the components required to do the job has made it possible for us to do the same type of computing work in desk-top microcomputers that it took a large room full of vacuum tubes to do 20 years ago.
The CPU acts like a receiver-amplifier in a stereo component system. Your stereo collects radio signals through an antenna and its receiver decipers those signals puts them in some kind of order, and sends them to the amplifier section. The amplifier boosts them up and transmits them to your speakers, where they are outputted as sound. Your amplifier can also receive signals from other devices as well turntables and tape decks provide stored signals that are processed and outputted. The CPU, too, collects or calls in signals (data) from different modules, acts on them and outputs signals through the same or other modules.

Ram
Random Access Memory is the module in a computer system that is the temporary storage device for programs and data for the CPU to process. This storage device keeps all the instructions, and data for those instructions, in an orderly fashion so that the processor can act on them.

Memory is organized much like the sorting boxes in a post office. Each memory location can hold (or store) a fixed amount of information. In a micro, each memory location can store one byte or eight bits. They also are numbered so the CPU can find them quickly and are labeled by an "address"(a sort of name).

Rom
Read Only Memory stores information just like Ram does, but it has one property that Ram does not. IT NEVER FORGETS. Even when the power is turned off, it will still remember what is stored inside.
This type of module gives us a fantastic tool that we use constantly. In the programming chapter we mentioned that the basic computer system itself can not do any - thing until we give it a list of instructions. We also mentioned that there was a list of those instructions that had to be used all the time. These are stored in ROM. Thus ROM allows us to store important, often-used instructions, so we don't have to keep inputting them, time and again.

Mass Storage

The mass storage module allows us to store a tremendous amount of data outside the computer. It is our active file cabinet, so to speak. Paper tape,audio cassette,floppy disk, and hard disk are the most common mass storage devices.

1/O or Interface

To get into the CPU in a way useful to us humans, we need input output devices, which connect to the CPU through interfaces. The interfaces are electronic circuits that permit controlled data flow from the CPU and its memory components to us on the outside at our various terminals. Our use to exchange information with the CPU.
There are two basic 1/O interfaces used in home computing: serial and parallel.

Serial Interface

Serial interface is widely used because it is the easiest to use. In the basic hook-up only three wires are needed: transmit data, receive data, and ground.
The serial 1/O takes the Ascii code, or the seven bits, and sends and receives that data one bit at a time. The serial 1/O is used where your computer system is linked to outside peripherals over a long distance. (The distance limitation is dependent usually on how the interface and the peripheral are constructed, either current loop or RS232. Both of these types of serial interface are used).

Paralled Interface
Parallel 1/O is a little more difficult to hook up than serial 1/O. Parallel types of interfacing require all the data bits to be connected at one time, so you usually have nine or ten wires going between the computer and the peripheral devices. You have lines for the seven data bits that make up the Ascii letter or number, a ground line, and one or two lines that we call "hand-shake" lines. These hand-shake signals communicate back and forth information between the peripheral and the computer. This information lets the peripheral know when the computer is ready to accept another character and vice versa.
This type of data transfer is done on a batch mode, in which the entire byte of word is transferred at once. This type of 1/O is used when you want to have a very fast data transfer. Its drawback is that the computer cannot be very far away from the peripheral.

What Can I Do With It?

The question most often asked of me is "What can you do with a personal computer"? The possibilities are endless, so I usually answer "Anything that you need a tablet of paper, a pencil, and a hand-held calculator for, you can do faster, more reliably, and more accurately with a personal computer. "The following list covers just a few uses. Let your imagination go and you will be able to come up with many more possibilities!.

Recordkeeping
Inventory Management
Routine Correspondence & Form letters
Filling out forms.
Calculations of all kinds
Receiving & placing phone calls
Matching any information with any other information
Polls and surveys
Indexing
Cataloging
Solving problems
Printing out results
Engineering design aid
Maintaining lists,especially;
- Mailing lists
- Shopping lists
- Stocklists
- Packing lists

Sales analysis
Travel & route planning
Scheduling
Ticketing
Distribution
Scientific calculations
Education
Accounting & billing
Invoicing
Sorting

Receipts
Taxes
Addressing
Budgets
Forecasting
Reducing the physical size of
   Files(customer files, corres-
   pondence, product files, etc)
   No more file cabinets!!
Playing games
Cashflow
Maintaining "tickle-files"
   (calendar-reminder systems)
Filling out checks
Playing the stock market
Filing
Remembering all transactions
Making & keeping card catalogs
"Simulating" results of one action
  versus others, so you can compare
  their outcomes.
Editing

The majority of people who have looked at the personal computer marketplace today have seen only the game playing side of it. They probably think computers are cousins to the video games now on the market. Though you can use a personal computer to play video games, it is not fair to limit personal computers to game playing. Don't get me wrong, video games are a great pastime and a different application for the home TV, but a personal computer can do so much more. I will touch on just a few of the possible home, hobby, and business applications possible with a personal computer.

A CONSUMER'S GUIDE TO PERSONAL COMPUTING AND MICROCOMPUTERS

Stephen Freiberger, P.O.Box 1427, Fremont, Calif.94538
Software Specialist, Digital Equipment Corporation

INTRO

Introduction

Hundreds of thousands of people are finding an interest in the new consumer electronic product called a microcomputer. The microcomputer is the basis of a complete microcomputer system which is often called a personal computer, home computer, or hobby computer. Over 30,000 people will probably attend this computer fair for the main purpose of finding out more about these new products. Many people attending this fair have no previous exposure to the technical areas of electronics, microcomputers, or large computer systems.

Due to the fact that many manufacturers assume the potential customer has a technical background, not all advertisements are written in an easy-to-understand language. Out of over 150 manufacturers of microcomputer products, only a small number take the time to prepare ads that are easily understood without prior experience. This presents a major communication problem to persons newly interested in microcomputers. Selection of a microcomputer system without full comparative data on others can be an extremely risky proposition.

This presentation and the book by the same name are designed with the sole purpose of helping someone new to the area of microcomputers to solve this problem. No previous knowledge or experience is assumed in electronics, computers, or microcomputers. The design is to define and build up basic knowledge in an easy-to-understand format. This will enable one to easily gain the basic information needed to make some kind of a comparison between the 1,000 or more microcomputer products available.

Selection criteria can be subtle as well as critical in selecting a microcomputer system that is suitable for the job. Information on choosing a microcomputer product will be presented. Many forms of microcomputer products will be discussed with their basic characteristics and cost ranges.

Some of the questions that will be answered by this presentation are: What exactly is a microcomputer, What makes up a microcomputer system, What forms of microcomputer products are there, How to choose a microcomputer or personal computer, What are the price ranges, and What to look for in selecting a good microcomputer or system. Other questions that arise can be answered after the presentation if they haven't already been answered by then.

Computers and Systems

What exactly is the distinction between computers, systems, hardware, and software? Most people have been exposed to a larger computer system either directly or indirectly. Computers are general purpose devices that can be programmed to process information or perform various tasks at incredibly high speeds. A computer program is a list of instructions for the computer to follow to complete a given task or process. An example would be a business accounting program or perhaps a Star Trek game program. Once programmed, the computer can execute the program at tremendous speed and accuracy, making complex caculations or decisions in less than a thousandth of a second.

A computer system, as its name implies, includes not only the computer itself, but also all related equipment needed to support it. This includes such things as printers or terminals that allow a user to communicate with the computer system. All of the physical equipment in a system are called hardware. A complete computer system also includes all of the computer programs, lists of instructions, and documentation necessary for it to perform useful work. These are called software.

An interesting note is that if you place a piece of software firmly in the electronics of the hardware, then it is called firmware. It's roughly analogous to silverware since it helps a person to easily access the piece of system

software without making a mess of things. The firmware might even be extended instructions to make programming easier or more efficient.

Mainframe-, Mini-, Micro- Computers

Three main categories of computers exist. The large computers or computer systems are called mainframes. They may cost anywhere from around $200,000 to over a million dollars or more. A mainframe computer system such as a large DECsystem-20 (1) is capable of sharing its computing power between up to a hundred or more users. Each user would appear to have her/his own computer. This is called timesharing.

The second category of computers is minicomputers. Often they are intermediate sized computers or systems which are not quite as powerful or expensive as most mainframe computer systems. Prices and relative performance vary widely according to the design and size of the minicomputer. The term minicomputer is difficult to define when you consider for example that the VAX 11/780 (2) minicomputer is faster in scientific applications than many large mainframe computer systems, but yet is less expensive.

The third and newest category of computers is the microcomputer. A microcomputer system is composed of at least three basic components. First is the microcomputer itself, which can be thought of as a small octopus capable of controlling many other devices. The microcomputer processes data and makes logical decisions about that data according to the program it is running at the time. The second basic component is some sort of device to input data into the microcomputer. This input device converts some form of data into electronic signals to be passed along wires to the microcomputer. A common input device is a keyboard similiar to the keyboard on a typewriter. The third basic component is some sort of an output device which permits the microcomputer to output information in some useable form. This might be a printer or a device to punch the information on paper tape. Any of the three basic component parts of the system may be physically distinct, or even combined together. For example a floppy disk unit is a device that can input information from a flexible magnetic disc, it can also output information to the same flexible disc. It could be called an input/output device. But since its main purpose is to store large amounts of information, it's more properly called a storage device.

The Microcomputer's Structure

A microcomputer is really a smaller less expensive form of computers that has certain attributes. Most microcomputers can easily fit on the top of a small desk or table. Like other computers, a microcomputer is normally a general purpose device capable of processing data and making decisions. The microcomputer has four main sections which are: a central processing unit, a memory section, input/output circuitry, and other support circuitry.

The main attribute of a microcomputer is the fact that it has a central processing unit based upon one or more microprocessors. The central processing unit, or CPU, controls the operation of the microcomputer and the flow of data within it. The CPU section of many microcomputers is often a single printed circuit card. Sometimes the circuit card is called a module or a board. The CPU section includes the microprocessor and the electronic circuitry to interface it to the rest of the microcomputer. The microprocessor is the nerve center or thinking center of the CPU section and microcomputer as well.

The memory section of a microcomputer holds information and programs that are immediately accessable while the microcomputer is running. The memory section consists of electronic circuits to hold the current information and programs, and to make it available as needed by the microprocessor. Information or programs can be stored in various types of memory for immediate access by reading or writing to various locations in the memory. The normal working memory is random access memory or RAM memory. Any location can be read or written immediately. A minor disadvantage of RAM memory is that if electrical power is removed, the data in it is cleared. Another popular type also found in many microcomputers is read only memory or ROM memory. Various types of ROM memory are used to permanently hold data or programs. They are read-only and the data in them can not be lost or changed normally.

The input/output circuitry permits information to be transferred into or out of the microcomputer. This section is the microcomputer's link to the outside world. Almost all input/output devices could be connected to this section. There are literally hundreds of types of input/output devices available. Each translates some form of data into a form suitable for the electronic circuits of the microcomputer. Usually the input/output circuitry includes several "ports" or various

interfaces to communicate with several input/output devices.

The other support circuitry includes all of the other electronic circuitry needed for proper operation of the microcomputer. This includes such things as the power supply to provide the other circuits with the power they need. It also includes such things as the circuitry needed to connect all of the sections of the microcomputer.

Each of these four sections of a microcomputer may be physically distinct or combined with one another. A larger microcomputer might include a memory section made up of more than 5 circuit cards just for memory. In another microcomputer the memory section might be found on only a small area of the CPU card. All of the four main sections of the microcomputer are interconnected electronic circuits that operate at high speeds.

The Microcomputer's Internals

Some of the design goals for these critical electronic circuits of the microcomputer are: low cost, high speed, maximum computing power, and small physical size. To attain these goals, the electronic design makes extensive use of small parts called "integrated circuits" in a "dual-in-line" package.

The integrated circuit or IC is a small electronic part that looks like a mechanical centipede with small metal feet. Actually the IC is a thin small rectangular package with 2 rows of metal pins. It is made from a tiny chip of silicon which has a microscopic electronic circuit on it. Small wires connect the metal pins of the IC to various points in the microscopic circuit inside. The size and number of metal pins depends upon the complexity and design of the circuit inside it. It may have as few as 8 pins or as many as 40 or more pins. An IC may contain the equivalent of as few as 5 to more than 50,000 electronic parts such as transisters, capacitors, and resisters, all integrated into one tiny chip of silicon inside a package less than a few inches long.

Thousands of various types of ICs have been designed and produced. Most new electronic circuit designs can be constructed from these basic building blocks. Each IC performs some specific task or function that it was designed for. It may be a counter circuit, a logical decision gate, a timer, or a multitude of other possible types of ICs. Many ICs can perform their task or operation accurately in less than a millionth of a second. This is what allows the microcomputer to operate reliably at incredible speeds, but yet be small in cost and physical size.

Large Scale Integration

As the semiconductor technology advanced, large complex electronic circuits became available on a single integrated circuit. These very complex ICs contain a large number of microscopic functional parts combined together. The technique used is called "large scale integration" or LSI. These LSI ICs may appear similiar on the outside to a standard IC, but often they are a little larger. LSI ICs may require as many as 40 or more metal pins in order to use the complex circuit inside them.

The small chip of silicon that holds the microscopic circuit actually only uses up a small area of the physical package. Ironically, most of the size of the package is required for the metal pins or leads. Enough spacing of the pins is needed so that it can be easily connected to a circuit card.

Many ICs of all types are connected directly to a circuit card in the industry. The normal method is by soldering the IC pins to printed metal traces on the card. For personal computers or hobby computers, it is much more desirable to use IC sockets on the card. This allows easy insertion or removal of the ICs if needed for repair.

The Microprocessor

A microprocessor is composed from one or more LSI integrated circuits. It is the main component of the CPU section. A key feature of the microprocessor is the ability to recognize and perform a variety of small tasks and instructions. The complex internal switching circuits allow it to perform or execute anywhere from around 30 to as many as over 400 instructions for some types of microprocessors. Hundreds of different microprocessors are available. They vary greatly in such things as speed, relative power of the instructions, the number of instructions, and their use. The microprocesor is the nerve center of the microcomputer and actually controls the electronics of the whole system.

The microprocessor makes decisions and processes data within the microcomputer. In order to accomplish this, the microprocessor reads a list of instructions of what to do next from the memory section of the microcomputer. It sequentially executes each instruction in the list of instructions or program one at a time unless directed otherwise. The program(s) to run are placed into memory for quick access when being run by the microprocessor. A branch or jump instruction, or a re-

set switch on the microcomputer may redirect the microprocessor to start following a different list of instructions also in memory. Other instructions may do arithmetic operations, or move a single character of data, or many other small tasks. The single character of data could be moved to a space in memory called a "byte". Since there are many locations available in the memory section, the microprocessor must also specify an exact location or address in the memory.

The microprocessor instruction might also output a single byte or character of data to the input/output circuitry of the microcomputer. In these cases, the microprocessor in the CPU section and the other sections within the microcomputer need to communicate. A specific address and a specific byte of data may be exchanged by these sections of the microcomputer. An "address bus" and a "data bus" are simply the common paths that run between the sections of the microcomputer as needed. An address or a byte of data then may easily be passed between the sections of the microcomputer along these common paths.

The System Bus and Flexibility

A system bus includes the data bus, address bus, and other needed paths between sections in the microcomputer. Those microcomputers that utilize a system bus design allow easy changes in their modular design. Most microcomputers utilize this type of design for flexibility and expansion capabilities. A small base platform with connectors is often used to connect all of the circuit cards to the system bus. It is usually called a "motherboard" or "backplane". This allows the microcomputer to be designed with a number of small modules or circuit cards. They can easily be plugged in or removed from the motherboard to allow changes or future additions. A user may want to plug in additional memory cards to increase the number of bytes of storage in the memory section. If a problem should arise in the hardware, it can often be traced to one card and fixed or replaced. Some varieties of microcomputers are designed as a single large circuit contained on a single card. These designs may be more compact, but do not make use of the modular design capable with a common system bus and cards to plug into it.

The Complete Microcomputer System

There are roughly five main categories of microcomputers and microcomputer systems. The first is a complete microcomputer system, which as the name implies is a complete and operational system. It includes a microcomputer with a CPU, memory, input/output, and support circuitry. It must have at least one input device such as a keyboard to enter information, and at least some kind of output device. The output device might be a video interface and RF modulator to display information on a television set. Many types of input/output devices can easily be seen at any computer fair, including devices that listen and talk back to you.

Most microcomputer systems provide for some kind of storage device to permanently hold whatever data or programs you have entered. Data or programs can be read or written to or from a storage device as needed. They are slower than memory, but can hold over 20 times as much information as an average size memory section. Some floppy disk units allow for permanent or temporary storage of up to 500,000 or more characters on a double sided flexible magnetic disc. The disc can be removed for safekeeping or a new one can be inserted easily to hold new information.

Software programs for the user should be included, but sometimes only a minimum of programs are included or even available. Since the natural language of the microprocessor is just numbers and codes, some better form of language is used to write programs. Assemblers are available to run on the microcomputer and let you program in mnemonics, which is a little better but not much. Other software such as a Basic compiler lets you write and run programs in Basic language. This is a lot easier to learn and use. A system monitor or operating system program allows the user to easily control and use the microcomputer system. A properly designed operating system program will have simple easy to use commands, but yet perform many powerful functions. Commands like type, copy, or run another program can make life easy for the user. An editor program of some kind allows the user to easily examine or change programs or data.

It should be noted that the software or programs for a system are as important as the hardware equipment. The software is often more expensive in time and cost than the hardware it runs on. If desired software programs are not available for your type of microcomputer, then the time spent writing them could be worth more than the hardware costs. On a large computer system it may take a month fulltime to design and write a complex program that works correctly. On a large computer system

the costs for software and programming may easily be over twice as much as the hardware expenses. Each year the hardware cost/performance ratio gets better, while the software costs continue to climb. Programming on a microcomputer will take longer than on a large computer. Powerful high-level languages such as COBOL (3) speed up design time on large computers. Many microcomputer systems have adequate software, but it is very important to verify the fact.

The complete microcomputer system is of course the most expensive category. It offers the advantages of being assembled, tested, complete, and last but not least, working and ready to use. Many are available in assembled form with prices starting at $500 at the low end to $9,000 for high performance systems at the top end. Some systems are also available in kit form, but they often need technical experience to assemble.

The Limited Microcomputer System

The second rough category is microcomputer systems which are more than just a microcomputer, but not quite a complete system. A complete microcomputer may be included with the CPU, memory, input/output, and support circuitry. However the system may be missing an input device, or it might have some other limitation. It may have only a storage device and no input/output device for the user. A prime example of this category is the microcomputer with a built-in floppy disk unit. It is much more than a microcomputer, but clearly not an operational and useable system by itself. Something must be added, but it would make a good starting point. A user can easily purchase input/output devices separately, provided that they are available and compatible.

Another type of limited microcomputer system is the small learning system contained on a single card or two. It usually has all the sections of a microcomputer included, but perhaps only a small amount of memory. Often it has a limited form of input/output device. The input device may be only a caculator style numeric keyboard, or perhaps a strangely formated keyboard to save space. The output device may be an 8 digit numeric display. This form could be called a system on a card and may be complete, although still a very limited system. This type of product may be priced reasonably at around $200 or more, but expansion into a full system may be expensive or even impossible. They are often designed as a learning tool or evaluation kit for a user to learn about a specific microprocessor type.

The Microcomputer in a Cabinet

The third category of microcomputers is the standard microcomputer in a cabinet. It should include all of the four sections and parts needed to make it an operational microcomputer. It might have a front panel with switches and lights, or it might not need one. It might have only an on/off switch. Prices for a standard microcomputer range from around $350 to as much as $2000 for more powerful versions. Some do not include any memory, even though they are called a microcomputer. Some include only the CPU card and power supply. Since options can be priced very high, it is best to also consider the cost of the total package. When adequate memory and input/output sections are added in, the given product may not be the best buy.

The Microcomputer on a Card

Many manufacturers produce a microcomputer based upon a single card. This fourth category of microcomputers has most of the sections of a microcomputer. It may only lack a power supply for the single card to be operational. Such single card microcomputers are convenient and are priced at around about $200 or more. They do require some work by the user to build a system around them. The main advantage is compact, low-cost computing power. Often thay can be used with some standard microcomputer system bus. This gives flexibility in designing and building a system. CPU cards are similiar, however they may not have any memory or input/output circuitry on the card. The CPU card is designed to be used in some type of system bus design with other cards such as memory cards. CPU card prices start as low as about $100. Circuit cards for other sections are similiarly priced.

The Microcomputer on a Chip

The last category of microcomputers is the microcomputer on a chip. Chip refers to a piece of silicon in an integrated circuit. The microcomputer on a chip is actually a special type of LSI integrated circuit. It is similiar to a microprocessor, but also contains memory and input/output circuitry. Microcomputers on a chip and microprocessors may cost as little as $5 to much more than that for more powerful versions. At this level, a great amount of work may be needed to complete a working system.

Selection Criteria

Selection from hundreds of manufacturers and thousands of microcomputer products can be extremely difficult. The hardware equipment should be evaluated carefully to see if it will meet all of the current and future needs. Replacement of a whole system in order to go to a larger system can mean a major loss of time and money. Many systems are adequate for learning during the first 6 months, but a user can quickly gain expertise and outgrow the limits of the system.

The software or programs provided should be user-oriented and powerful, but also easy-to-use. You shouldn't have to read a list of codes to figure out what command code to enter next. One should look closely at the memory and input/output circuitry of the microcomputer, first to see if there is any, and second to see what amount. A common measure of the size of the memory is the number of "K" bytes of space. The "K" stands for a size of 1,024 bytes or locations of memory. If the memory size is specified in bytes, then a 2K byte memory card is equal to the space needed to hold 2,048 characters or bytes of data.

Another important factor in selection is the ruggedness and reliability of the product. If the product proves unreliable, repair service may be very expensive or not even available in some cases. If the input/output devices work sometimes but not always, the system is ten times as hard to use.

Careful attention should be given to input/output devices to select those that best fit your use. One most important step is to check that a variety of options for the system or microcomputer are available. At some point they may be needed. This includes not only optional cards and devices to expand the microcomputer, but also a variety of programs to make the system easier to use. The user should not be expected to have to sit down and write programs with only limited help from the system. Useful programs should be included or available to help the user.

In general, a serious purchaser should carefully review the detailed specifications of a product to see that they are not exaggerated. A comparison of features and items included at no charge will help. One often used technique which may answer a lot of questions, is to locate and talk with a previous buyer. Even if the product is a different model or version, it will indicate some of the good or bad features of a specific microcomputer product.

Summary

It is hoped that this presentation and the book by the same name will help a person new to microcomputers to wade through the technical jargon of the industry. Given a little time, these same people will someday notice that they have mastered even the most technical aspects of personal computing.

Further information can be found in the book "A Consumer's Guide to Personal Computing and Microcomputers" by Hayden Book Company. The book is an introduction and guide to microcomputers and related topics, including a detailed purchasing guide with prices. The book assumes no previous knowledge or experience and is available from Hayden Books, Hobby World, or Jade Computer Products.

The cooperation of the co-author Paul Chew, Jr., and over 50 manufacturers has made the book possible and is greatly appreciated. I will be available after the presentation or through the mail to help answer questions that may arise. I extend my thanks for your participation and interest in the computer fair.

Acknowledgements of Assistance

1. The co-author of the book, Paul Chew, Jr.
2. The microcomputer industry manufacturers whose cooperation is greatly appreciated.
3. The publisher, Hayden Book Company

Reference Notes

(1) The DECsystem-20 is a registered trademark of Digital Equipment Corporation. It is a family of large scale mainframe computer systems.

(2) The VAX 11/780 is a registered trademark of Digital Equipment Corporation. It is a powerful minicomputer which outruns many mainframes in scientific uses.

(3) COBOL is a widely used business computing language with extensive programming statements and file handling features.

(4) The title of this presentation is taken from a copyrighted book, "A Consumer's Guide to Personal Computing and Microcomputers", copyright 1978, by Hayden Book Company.

THE VISIONS OF A FUTURIST

Alan P. Hald
The Phoenix Group, Inc.

Today's nonsense often becomes tomorrow's realities. Dreams and visions are powerful forces in influencing the directions we choose. What may have been dreams in past decades can become matter-of-fact in the near future. Right now, a doctor in Anaheim is probably completing her prototype of C3P0. She plans to sell the kit version early next year, just in time for Star Wars II. And how about that engineer who's been working on a $2995 tabletop IBM 360/158 emulator in his spare time? He hopes to sweep the small computer market next fall. Think of all the possibilities. I personally feel that information technology, and especially microcomputers, have initiated a period of explosive innovation that will be unequaled in modern times, indeed, a New Renaissance.

Have you ever thought what it might be like if you could carry on an intelligent conversation with an inanimate object - like your house? Home computers can already talk, interpret speech, and control appliances. Throw in a few sensors, a modest vocabulary, the Bell Telephone system and your house could talk to about anyone or anything in the world.

Imagine, you're at work, the phone rings. It's Fred, your house. While monitoring the morning news reports for stories of recent burglaries, Fred picked up a weather bulletin warning of pending heavy rain. This jogged Fred's bubble memories to run a routine roof maintenance check. A potential leak was found. Before calling you, Fred phoned Slim for advise. Slim is a ranchstyle home down the block. Normally, one reference wasn't enough for Fred, but Fred and Slim often shared data banks and each knew they were programmed with an effective search technique for identifying household services. Having accepted Slim's advise, Fred placed that call to your office.

You've learned to trust Fred's judgment, and approve the repairs. The rest is rather straight forward. Fred calls the roofer and directs her to the leak. After it's repaired to Fred's satisfaction - Fred is conditioned to be assertive - funds are electronically transfered from your account to the roofer's account (remember you already approved this). Fred promises to give her a good reference and wishes her well as she leaves the yard.

One footnote. That night, after you've relaxed, Fred tells you some bad news. It seems The City has passed a minimum maintenance ordinance requiring that all homes must maintain themselves to standard, regardless of the owner's financial capability. From Fred's projections, it seems you must either get a better job, find a roommate, or another home. By the way, Fred has already found three potential roommates that.....

Could you imagine being a child, growing up in a home like Fred? Slam the door or write on the walls and Fred asks you to stop. You'd develop a different sensitivity to your environment. Objects would become more like people. You'd relate to them with care and affection.

Will Fred and others like him become a reality? I think so. This application of microcomputer technology satisfies the basic desire for convenience and the need for conservation. Two conditions that often appear to be at opposites in our society.

Back to childhood. Growing

up with Fred, who is also a babysitter and playmate, can be dull at times. What else is available? Of course, there is the home computer everyone talked about in the late seventies. Sitting right there in your telephone. Where? Actually, they're all over your house, even the Robovac unit can play word games. But what really interests you today is the video disk library of learning. I remember as a kid how I would randomly thumb through the World Almanac, absorbing facts because it was fun. Now you're bouncing through your playground of knowledge, watching video on the screen with audio scaled to your learning level. From birth you've been exposed to this media, your thinking is more conceptual, you look for and see patterns. You're experienced in searching for and finding information on anything. You're only five years old. You're a genius by my standards. I'm from an era of Captain Video, Mister Wizard and Mickey Mouse. Could you really be interested in Lobaschevsky surfaces?

Well it seems you are. So are twelve other of your friends you network with regularly. Although recently, your family has become intrigued with Econogame, reducing your time on the net. Econogame has become the 21st Century equivalent of Monopoly. A funny thing happened to banking in the eighties, it faded away into electronic funds transfer systems - EFTS - and with it rose Econogame, the simulated real time economy. It's played interactively and concurrently by 50 million North Americans via the public net.

As more people began playing the game, they found themselves making better personal decisions on buying, saving, working, and sharing. In fact, it's becoming hard to distinguish the real economy from Econogame. The game is becoming a near perfect model, anticipating needs and results from large numbers of individual choices. The Government even uses the game to fiddle with the economy. It appears that direct intervention in the real economy is too risky - it upsets the game!

I'm amazed! Here we are in the future, playing computer games again. Yet, they've become very real and meaningful experiences. The distinctions between the game and reality have blurred. Look closely, we are witnessing the beginning of an era where the distinctions fade away. Where reality is a concensus among all the players. Where existence becomes an expression of the imagination. Truly a time of visions, dreams and nonsense.

About the Author:

Alan P. Hald is a futurist, co-founder and President of the Phoenix Group, Inc. During the past two years, the corporation has grown to become an integrated retail, wholesale, service, software and publishing organization, serving the world computer retailing and OEM markets as MicroAge Wholesale, MicroWorld, MicroSource, MicroShopper/ByteShopper. He currently resides in Phoenix, Arizona, and is a graduate of Rensellaer Polytechnic Institute and Harvard Business School.

PERSONAL COMPUTERS AND SOCIETY: WHAT NEXT?

Jack M. Nilles,

University of Southern California
Los Angeles, California 90007

(1)

Abstract

An overview is given of a recently begun research program at the University of Southern California. The program is assessing the development of "personal computer" technology, its present and potential societal impacts, and the public policy issues arising from them. The description covers the key factors in the assessment.

Introduction

A dominant fact about contemporary "developed" societies is that they are rapidly becoming dependent on the information economy. The manipulation of information occupies the working hours of at least half the labor force in the United States and other developed countries. A significant factor in this trend has been the electronic computer. In thirty-odd years, computers have become a pervasive influence in our society. Yet, until recently, the computers were always owned and operated by the "establishment": the big institutions, the government, the large corporations. Now, or in a few years, they will be available to almost everyone - or will they? - or should they? Will they bring on the millennium or will they bring us to some sort of social Armageddon?

Will personal computers:

- o raise educational standards?
- o lower educational standards?
- o increase employment?
- o expand the information society?
- o produce a new set of culturally disadvantaged - the information poor?
- o decrease employment?
- o increase interpersonal communication?
- o increase depersonalization?
- o improve work skills?
- o decrease pride of workmanship?
- o decrease energy use?
- o invade our privacy?
- o increase energy use?
- o protect our privacy?
- o cause economic chaos?
- o reverse our deficit trend in the international balance of payments?

The answer to these and many other similar questions is probably, yes. Personal computers will be instrumental in at least all of the above, although in different sectors of the economy and at different times. It is important to realize though, that personal computers themselves won't do any of these things. The people who own and operate them will.

The questions that face us concern the likelihood of each of these and similar possibilities over the next ten to twenty years, the potential controllability of the extent of these effects, and the implications of these alternative futures on the ways in which we govern ourselves. To paraphrase a statement made by a great statesman (Will Rogers), "personal computing is too important to be left to the computerists." The societal implications of these devices, or collections of devices and mediated thoughts, can be at least as great as those produced by our uses of the automobile. In fact, there are many parallels between the two technologies. Each gives the owner considerably more freedom of action than was available through earlier technologies. Each caused (or will cause) adjustments in the culture of the users. Each influenced (or will influence) the development of massive capital expenditures in the form of an associated transportation network (people in one case, information in the other - both networks are substantially developed at present). Both produce(d) effects totally unanticipated by the original entrepreneurs of the technology.

A Technology Assessment

All of which leads to the central theme of the research program at the University of Southern California. We have recently started a technology assessment of personal computers, the general purpose of which is to provide information to public policy decision makers about the probable avenues of development and uses of personal computer technologies, and about the policy options available to the decision makers to encourage, promote, discourage, or restrict particular avenues of development. We are not trying to predict THE FUTURE. We will try to forecast alternative future developments (at least in the near term) in sufficient detail to allow policymakers to anticipate potential positive and adverse impacts in time to react appropriately to them.

As an example, the FCC is just now undertaking its Second Computer Inquiry, primarily oriented toward the implications and responsibilities of large-scale computer data communications. Communicating personal computers could cause a widespread shift toward increased use of data communications, accelerating the trend toward all-digital transmission. This in turn, could have sig-

(1) This article is a condensation of a paper presented at the IEEE Western Microcomputer Conference, Fort Collins, CO, 28 August 1978. The opinions and conclusions expressed here are solely the responsibility of the author.

nificant effects on the telecommunications industry, the balance between specialized common carriers and "Ma Bell."

The telecommunications related issues are only part of the potential consequences of the widespread adoption of personal computers. Some of the potential opportunities for, and threats to, various interest groups (stakeholders) afforded by personal computers include: raised educational standards; increased employment in personal computer production and service sectors; education and employment of the physically handicapped; decreased property crime; geographic dispersion of employment; increased competition to the formal educational system; reduction in unskilled labor opportunities; increased copyright infringement; and changes in leisure activity patterns. In many cases these are competitive; realization of an opportunity by one set of stakeholders will mean materialization of a threat to another set. Conflicts will result. Some will be resolved by market processes but many will have public policy implications, requiring some form of legislative alteration of the "free" market, either to reduce perceived threats to the public welfare or to bring about or accelerate the desirable uses of the technology.

Forecast of Direct Impacts. The first goal of the technology assessment is to develop a comprehensive forecast of probable direct impacts of the use of personal computers. It is also necessary to get a better idea of the interrelationships between potential uses and potential users of personal computers. Another dimension to this analysis is the level of competence required of the user to exploit a particular manifestation of the technology. Users will vary from occasional dabblers to full-fledged professionals in the computer field. Because of the diffuse potential market for personal computers, this objective is more difficult to obtain than it might seem at first glance. At this stage of development of the technology, many of the potential uses of personal computers have been discussed in general terms, but few specific realizations, other than interactive games, have occurred. Hence, the development of a broad data base concerning existing, developmental, and planned applications of personal computers will constitute a major first task of the program.

Forecast of Secondary Impacts. The development of an accurate secondary impact analysis is always one of the major features of a technology assessment, as it is for this one. Emphasis is placed on how the technologies are used rather than which manifestations of the technologies are most used; on how the lives of the users and others are altered as a consequence of the technology use.

Identification of Actors, Stakeholders. In the process of identification of direct and secondary impacts, those parties interested in, or affected by, the use of personal computers will also be identified. The actors are those who play a direct part in the diffusion and/or use of personal computer technology. They range from individual owners of personal computers to major organizations interested in constraining or directing the marketing and applications of personal computers. Those individuals or organizations who have something to gain or lose as a consequence of the increased use of personal computers are stakeholders. Their position in the process can either be direct, such as a manufacturer of personal computers, or indirect, such as a citizens group interested in improving educational opportunities.

Identification of Public Policy Issues. Satisfaction of objectives just listed allows the development of some fairly explicit scenarios of the "who does what and to whom, and what happens then" variety. Various combinations of direct and secondary impacts, together with the interactions of the stakeholders involved, will result in various classes of conflict of which many will have public policy implications. The penultimate objective of a technology assessment, one which has not always been reached in the past, is to identify some explicit potential public policy issues arising from the use of the technology under consideration.

Formulation, Assessment of Public Policy Alternatives. The next objective in sequence of this process is the evaluation of potential alternative public policy positions to be arrived at as a consequence of the issues identified. These policy alternatives must necessarily take into account the realities of the political environment at the time the issues are likely to come to fruition. Consequently, the assessment of the alternatives must include consideration of the relative short and long term effects of the proposed resolution of a particular issue, the relative influence of various actors and stakeholders in the policymaking process, and the influence of many relevant, exogenous events and trends.

Identification of Key Trend Indicators, Analytical Methodologies. No technology assessment has yet been able to achieve 100% foresight. The purpose of many of the methodologies which we intend to use is to reduce the probability of error in forecasting the events and trends associated with development of the technology. The error cannot be eliminated entirely. Since the proposed policy alternatives resulting from the technology assessment will be predicated on these forecasts which may be erroneous in part, it is also necessary to provide the policymaker with some clues as to when he or she might start to consider investigating whether or not the forecasted issues are beginning to emerge. These clues take the form of brief descriptions of key events or trends which would tend to indicate that one or more of the scenarios presented in the technology assessment are in fact occurring.

Dissemination of Planning Information. The ultimate action objective of any technology assessment is the adequate dissemination of the information developed during its course. Three requirements must be met, assuming the technology assessment is otherwise of high quality, before it can be considered to be effective. First, the results of the technology assessment must be made available to the policymakers likely to be concerned with future policy formulation and implementation. Second, the information must be timely. Third, the policy options and indicators must be presented in a form easily understandable to the decision makers. It is this latter criterion particularly which may require a different reporting process, or at least additional modes of reporting, than the traditional technical report.

Anticipated Consequences of the Technology Assessment

The hoped-for consequence of any technology assessment is the reduction of entropy (wheel spinning) in the society to which it addresses itself. The hoped-for consequence of this technology assessment is a smoother and more societally acceptable transition from the present state of technology-driven development and dissemination of personal computers. This will occur to the extent that the results of the assessment are communicated to (and understood by) the various stakeholders and decision makers.

Plans for the Future

The program just outlined is scheduled for completion early in 1980. Since it is just beginning, we have no factual data at the time of this writing to support any conclusions concerning the hows, whys, and whens of the impacts of personal computers. As a matter of fact, we take this opportunity to solicit contributions from readers of this paper on present and potential uses of personal computers, together with documentation, where possible, of the effects of these impacts on household or business economics, life styles, education, legal issues, or any of the other factors already mentioned. Information submitted to us will be used to supplement our technological and market forecasts and impact analyses. [2]

Notes

[1] Jack M. Nilles is Director of Interdisciplinary Programs in the Office of the Executive Vice President and is Principal Investigator of the project discussed here. Coinvestigators include Paul Gray who is a Professor of Quantitative Business Analysis in the Graduate School of Business Administration, F. Roy Carlson, Jr., who is Director of the Engineering Computer Laboratory and Assistant Dean of the School of Engineering, Milton Holmen, Professor of Management in the Graduate School of Business Administration, and John Hayes, an Associate Professor of Electrical Engineering and Computer Science.

[2] Inquiries may be sent to Jack M. Nilles, Director, Interdisciplinary Programs, University of Southern California, Los Angeles, California 90007.

CHANGING PARADIGMS AND THE COMPUTER

Carl Townsend
Center for the Study of the Future
4110 N. E. Alameda
Portland, Oregon 97212
503-282-5835

Abstract

During the next decade our culture will experience a major paradigm shift. The Gutenberg press made information available to the masses, initiating the Industrial Age, the microcomputer will now allow us to control this information flow. This will cause radical changes in our political, economic, and educational systems as well as changing our concepts of transportation and communication.

The computer is only a tool. We can create the future we want and use the computer to help us implement the future. What we do with the computer will determine our future or whether we have one or not.

Introduction

The microcomputer was born into a very diverse culture. Never has the world been more complex. The opinion leaders of our world find themselves having to make decisions and determine our future from a people with widely divergent views of reality. A politician may find himself eating breakfast with a businessman arguing for strong economic growth, business expansion, strong central government, and continued deficit spending. At lunch he may be with a futurist arguing for a concept of limits, a balanced federal budget, and a decentralized government. Who is right?

History has always been controlled by ideas. Today, there is a critical need for new ideas, for transformal thinking. For the first time in history we find ourselves trying to change our entire culture by a conscious process. This will have profound impact on our political, economic, medical, religious, and educational systems as well as changing our concepts of work, leisure, transportation, and communication.

The applications to which we put our technology will have a radical influence on the type of future we will have and whether we will have a future or not. Technology is amoral--it is neither right nor wrong. It is what we do with the technology that will determine our destiny. Communications and information systems will be the frontier technologies in the next decade, and the computer will be a major component in these systems. Costs for most systems will drop by a factor of ten within the next decade.

Two Paradigms

It is perhaps dangerous to attempt to define the new paradigm. Many writers have tried to coin their own word for whe emerging age:

Harvey Cox	technopolitan
Ken Boulding	post-civilization
Daniel Bell	post-industrial
Robert Theobald	communications
Peter Goldmark	"
Alvin Toffler	super-industrial
Marshall McLuhan	electronic
Zbigniew Brzezinski	technetronic
Betran Cross	mobiletic revolution
Elton Trueblood	post-denominational

We may agree or disagree with these, but one thing is certain. All of these futurists are creating ideas and visions of the future, and in doing this they are influencing greatly the decision makers in today's society. The great decisions before us--genetic manipulation, behavior control, resource management--will be made by someone. Unless we become more involved in these it will be someone else who makes the choices, and we will have to live by the choices made by someone else.

In the first figure is a chart showing two paradigms. One paradigm assumes the industrial age "extended". It assumes the future will be like the present, only more so. The second is a transformed paradigm showing some pos-

sibilities if we are willing to commit our resources.

Changing Paradigms in History

Change today occurs much more rapidly than in the past. The Agricultural Age lasted thousands of years until 1760. The Industrial Age followed the Agricultural Age and has lasted two centuries. Our Industrial Age paradigms are now failing us. How long will our next age last?

The rapid change in paradigm shifts causes stress in our lives. Some people are not able to adjust to this change and live in a world of the past trying desperately to hold onto outdated models. Others of us are anxious to travel into the new age and see new possibilities with hope.

Toward the end of the Agricultural Age the Gutenberg press was invented. This technological break-through made information available to the masses. Gutenberg died a pauper, never realizing the full effect his invention would have on the world. A Bible was printed that could be read by anyone, giving a base for the Reformation from which our current political system emerged. The Gutenberg press also made possible the Scientific Revolution and the eventual paradigm shift to the Industrial Age. The information explosion from the invention of the Gutenberg press made it possible for the ideas that controlled history to be transmitted rapidly through the cultures of the world.

Not everyone was enthusiastic about the new age. In 1811 in Nottingham the "Luddites" rebelled against the machinery that was destroying their jobs in the textile mills. The revolt spread over much of Scotland before it could be stopped.

The Microcomputers Influence in the New Paradigm

In Figure 2 are shown some new paradigm possibilities with the microcomputer. Either of these possibilities will cause major cultural changes. Like the Gutenberg press, the microcomputer will alter the way information flows in our culture, creating a new culture.

Currently information flow has increased to such a level that we can no longer assimilate the majority of information that is sensed. Hundreds of advertisements touch our senses every day. Several dozen computer magazines are printed every month, but who can read all of these? Each of us can identify a dozen books we have seen here we would like to buy. When would you read them? We scramble about the faire trying to interview people for articles and books we are writing as those we are trying to interview scramble about the faire to gather their information on products, plans, and events for their purposes.

The personal computer will soon allow us to control our information flow by "windowing" information against our personal specific needs. The computer will be our "gatekeeper", only allowing the information that we specifically need to reach our senses.

Television, for example, will change from broadcasting to narrowcasting. Videocassettes and disks will carry programs that relate to our specific interests. The television will be two-way, permitting response for gaming, shopping, voting, and such. A pilot system is already operational in Cleveland. The newspaper will be printed on a disposable videodisk. Libraries will carry disks and cassettes with catalogs available on some type of media storage that can be rapidly scanned at home by a personal computer against keywords of interest. Programs that normally reach only a small audience as operas and special religious programs will be made economically. One church in Portland has already proposed the idea of using a small computer to access selected portions of video tapes to train small group leaders.

Advertising will be transmitted to our homes by FM (as Jim's Digicast) or by cable television, and scanned by our personal computer to locate products we need. Advertising will be much more broad-based (including an expanded classified section of a wide variety of products) and more competitive. Imagine a real-time garage sale by computer! Electronic bartering will emerge as a convenient means of bypassing the problems of inflation, and the government will not be able to control this.

Our computers will become nodes in networks and linkage systems. These will be linked by satellites, radio, telephone lines, and lasers.

The laser will drop in price from $1000 to $20 in 10 years enabling it to become more cost/effective in communication systems. The computer will manage the information flow between these nodes so that information is transferred when costs are lowest so that only necessary information is transferred and so that the information is transferred by the method that costs the least. Each of us will be building personal data bases of information that meets our specific needs.

These networks will also support computer conferences that will continue 24 hours a day. A few weeks ago I checked into one of these computer conferences. Access was through a video terminal and modem at a friend's home. The "attendees" included names that would be familiar to most of you here as leaders in the computer revolution. I left messages for certain people and received one myself. To attend the conference I only drove 30 minutes from my home. The cost was about three dollars an hour.

Where Does It Begin?

We each carry a vision of what the future will be like. The picture may be nebulous or unclear, but the vision is there. Your personal paradigm of the future is determined by your own past and the cultural forces that are acting upon your life daily.

We can each choose. We are not robots controlled from Madison Avenue or from outdated Industrial Age systems. We are free, and the future is ours.

TWO PARADIGMS (WORLD VIEWS)

INDUSTRIAL AGE PARADIGM	TRANSFORMAL PARADIGM
Insensitive to cultural environment.	Culture sensitivity.
Reductionism, logical analysis, rationalism (we could solve any problem if we had a big enough computer.)	Holistic-emphasizes the functional relationship between the parts and the whole.
Structures and organizations are result oriented.	Structures and organizations are process and change oriented.
The future is determined from trend analysis of present data using the rules of extension.	The future will be of a different form than the present. Trend analysis has little meaning.
The future is determined from the present.	The present is determined from the future.
Organizations are rigid, as large as possible, with a good bit of inertia.	Organizations are fluid, changing, dynamic, and short-lived.
Institutionally oriented.	Ecosystem oriented.
Elitism-control of the future is determined by a few people in key positions.	Participating-control of the future is determined by the whole acting as an an organism.
We live in a homogeneous world, with little variation with time and geography.	There is considerable cultural diversity in time and space.

Figure 1

INDUSTRIAL AGE PARADIGM	TRANSFORMAL AGE PARADIGM
Forced dependence on an economic system using electronic credit cards and centralized data bases on personal credit status.	Independence of economic system using electronic bartering and distributed data bases to match needs and resources (minimizes taxes).
Automation creates extensive standardization and routinization of life.	Personal computers used to maximize individual creativity, providing almost endless variety.
Computerized educational tools used to develop behaviorally conditioned response to existing Industrial Age paradigm.	Computer used in alternative future simulations for energy and resource control. Participatory democracy.
Centralized information control with large computers for manipulation and regulation using large data bases.	Decentralized information flow with decentralized networks of smaller computers and personal data bases. Computers used as gatekeepers to control information flow and prevent local information overload.
Madison Avenue advertising created by computer analysis of markets used to sell products we do not need by falsely identifying these products with projected needs.	Individual and personal definition of needs and these used as gatekeepers in personal computers. Computer selects products based on real needs using computer networking and electronic bartering.
Centralized work locations with defined working hours and large transportation overhead (transportation intensive).	Decentralized work locations (from homes) using terminals (communications intensive).

Figure 2: TWO ALTERNATIVE PARADIGMS

A MICROCOMPUTER MUSIC SYNTHESIZER

Henry L. Pfister
USC, ISE Dept, OHE 400, University Park,
Los Angeles, California 90007

Abstract

The microcomputer music synthesizer is a personal system for the creation of computer generated music. If functions either as a real time instrument that is played with a keyboard or as an entertainment system that will play programmed music or compose new music. The goal is to achieve a low cost system that allows the user to produce high fidelity audio output using both digital signal synthesis and digital control of analog synthesizers.

The system is designed using a KIM-1 along with a number of kits and custom hardware. The individual components consist of a five octave electronic keyboard, a modified TV display,an ASCII keyboard,16K of ram memory, a cassette recorder, a stereo amplifier, 16K of prom, and an analog to digital converter for joysticks and pots. Some of the methods implemented or that are planned for sound synthesis are multiple square wave outputs, a two channel D/A converter direct digital control of VCO/VCF/VCA modules, and digital filtering of a psuedo random noise source.

The system software consists of the KIM ROM Tiny BASIC, an assembler, an editor, and real time I/O subroutines. The applications software includes real time performance programs and interactive composition programs.

At this time the system hardware is partially assembled and the software is being developed. The system is currently able to perform moderate quality music with square waves and with the D/A converters.

Music Synthesis System

The microcomputer music synthesizer project was conceived after a variety of electronic music experiences. First, I used a KIM-1 to play simple tunes. Second, I acquired a surplus five octave electronic keyboard. Third, I went to a demonstration of the Bell Laboratories portable sound synthesizer. Fourth, I heard a recording by Walter Carlos of Bach performed with a synthesizer. The project grew naturally from these experiences and from a strong desire to build an inexpensive personal computer system.

My first attempt was to interconnect all of the hardware with no particular plan for the system. This resulted in the classic computer system failure of not specifying the requirement and then not achieving the desired goal. After a careful evaluation of the existing mess of hardware and software. It was apparent that the apporach for the development of a personal computer system required the same discipline and standards that I would use in any computer system engineering project.

The steps I chose to develope the system are as follows:

1. Identify the goals of the system,
2. Specify the constraints on the system,
3. Define the operational requirements,
4. Design the system in detail,
5. Develope an implementation plan,
6. Construct the system in a modular way,
7. Redo steps 1 thru 6 until satisfied.

It should be noted that steps 1 thru 5 require only writing and rewriting. The construction of the system in step 6 is a modular approach so that some version of the system will always be working.

Goals and Constraints

The goals of the system are to be able to perform, compose, record, replay, and edit electronic music. These goals are to be done in real time by a digital approach that will allow the synthesis of several instruments and voices. In addition, the system should have high fidelity audio and function as a general purpose computer.

The constraints on the system were to use the existing hardware and to only add other hardware as required. In addition, the amount of money available was limited as was the time for development. Thus careful planning would be required if the project were to ever be completed.

System Requirements

The requirements for the system were stated at a high level in order to define the environment and the intended use of the synthesizer. They include the following features:

1. Interactive performance of music.
2. Multitrack recording and editing.
3. Computer controlled composition.
4. Modular design for development.
5. Flexible design for many users.

Each requirement is expanded in the following paragraphs.

The interactive performanc of music includes the interface with an ASCII keyboard, with an electric piano keyboard,with the KIM-1 keypad, and with numerous joysticks, pots, and switches. All of these elements are fully programmable and will respond in real time to performer inputs.

The multitrack recording and editing of music implys the simulation of an analog multitrack tape recorder as used in a sound studio. These recorders allow you to play back on selected tracks while simultaneously recording on other tracks. One of the key advantages of

digital recording is the independence of playback speed and frequency. With this capability it is possible to independently adjust the pitch and duration of each note. The editing software will be required to support time adjustments by instrument, track, or score.

Computer controlled composition allows the microcomputer to specify instruments, select notes, and generate musical output without human intervention. The composition software will support mathematical composition algorithms to be written in a customized version of BASIC.

Modular design for incremental development implys that the system will always be in an operable configuration. As early as possible, a limited version of the system was constructed so that it could be incrementally expanded into the final system. The emphasis on modular development is to be able to test new ideas while the system is being developed.

A flexible design is required to serve a variety of users. This requirement implys the use of the system by the whole family. The members range from a six year old who is learning music to an experienced adult. The system should the allow a performance range from simple tunes to complete compositions involving several instruments.

System Design

The advantage of writing the requirements before you begin to design is that you can immediately construct a final picture of your system. This is especially important in designing something as unique as an interactive musical instrument. The layout of the controls and their intended use should be carefully considered from a human factors point of view. This synthesizer is patterened afterthe Bell Laboratories portable digital sound synthesis system. The major difference is the number of interactive controls. The reason for the much smaller number of controls is to avoid manual mistakes and to reduce the cost The primary human factors considered in the design are as follows:

1. Any external control should be reachable by either hand.
2. The keyboards should be usable while seated or standing.
3. The system should be light weight enough for one person to move.
4. The enclosure should allow easy access to any component.
5. The finished system should not appear out of place among average home furnishings.

Figure 1 illustrates the synthesizer external design. Once the external design was complete, the internal placement of the components was quite natural. Figure 2 illustrates the placement of the system components. Table 1 lists the components by manufacturer.

System Operation

The concept of operations of the synthesizer is simila to that of the Bell Labs model. The most notable differences are a 200 to 1 cost ratio versus a 10 to 1 performance ratio. Table 2 gives a detailed comparison of the two systems. In both systems a performer can in real time do the following tasks:

1. Play notes at a keyboard,
2. Modify the instrument he is playing,
3. Replay and edit his performance,
4. Merge a new instrument with an existing score.
5. Simulate a multiple instrument orchestra.

In comparing the systems the key features are the keyboards, the controls, the computer, and sound synthesizer methods.

The Bell synthesizer features two five octave multiposition keyboards while this system has one five octave single position keyboard. However, both systems allow for independent programming of each key. The Bell system has 72 slide pots for inputs while this system has only 8 pots. However these are multiplexed and the liklihood of simultaneously varying more than eight are rare. The replay, edit, and merging capability are similar.

A more funamental difference exists in the sound synthesis methods. The Bell system uses a 64 channel digital synthesizer that is one of the best sounding in the world. However, the Bell system is integraly designed using this synthesizer. This system is built to use an S-100 bus so that any synthesizer that becomes available as an S-100 card will be available for use.

Implementation Plan

The implementation of this synthesizer was planned as a number of compatible configurations. Each later configuration will support the functions of the earlier configurations. Table 3 lists the planned set of configurations.

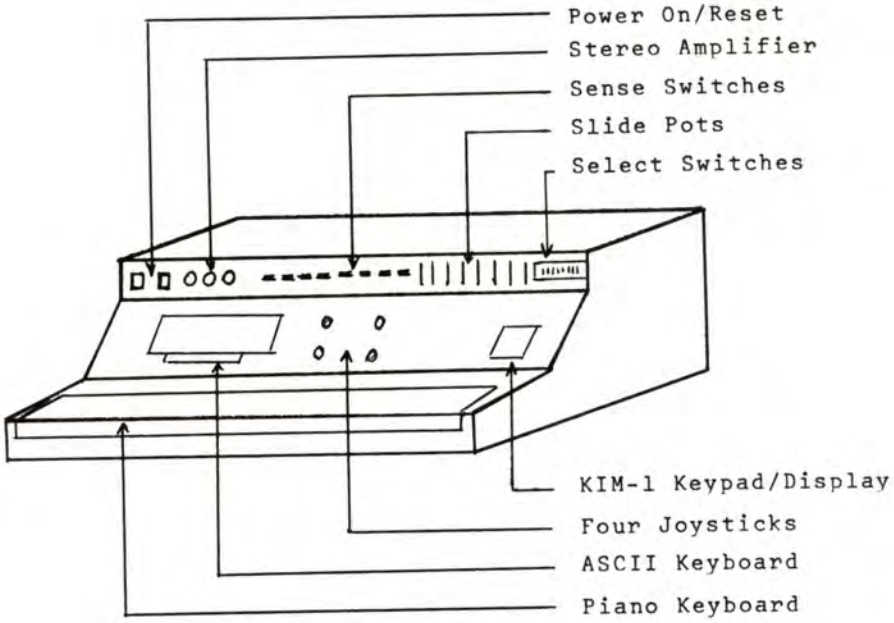


Figure 1. System Design

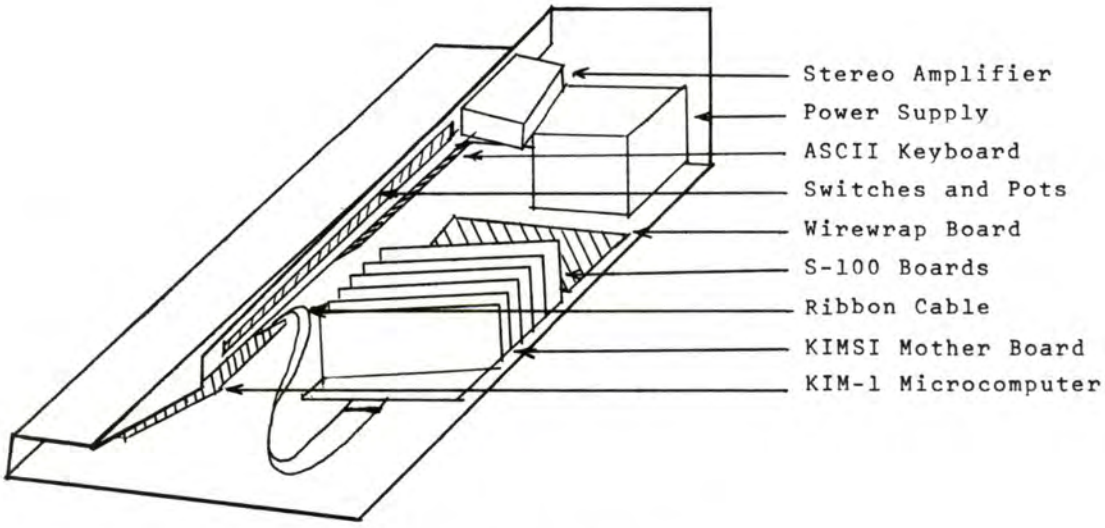


Figure 2. Component Placement

Figure 3. Synthesizer Block Diagram

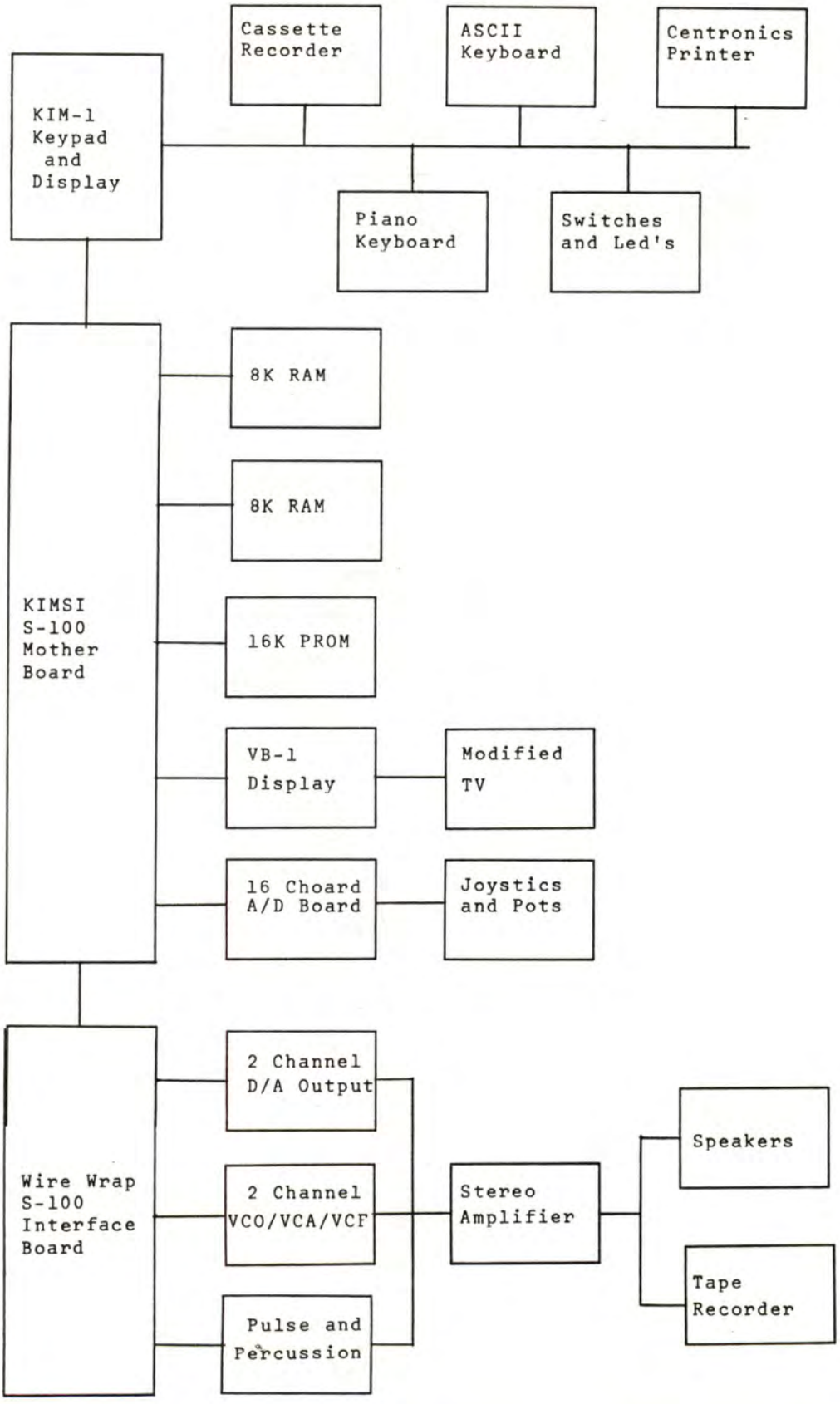


Table 1 Microcomputer Synthesizer Components

Component	Manufacturer
Power Supply MP-P	Southwest Technical Products
KIM-1	MOS Technology
KIMSI Motherboard	Forethought Products
ASCII Keyboard	Electronics Warehouse Inc.
Video Display VB 1	Solid State Music
Ram Memory	Ithica Audio Problem Solver Systems
Prom Memory	Ithica Audio
Piano Keyboard	Aries 5 Octave
Audio Amplifier	Formula One Inc.
Synthesizer	E MU Prototype Board
S-100 Interface	Custom
16 Channel P/A	Custom
Audio Interface	Custom
2 Channel A/D	Custom
Tiny Basic	Tom Pittmon
Micro Ade	Peter Jennings
I/O Interface	Custom

Table 2 System Comparisons

Capability	Bell Labs Synthesizer	Microcomputer Synthesizer
Processor	LSI-11	6502
Ram Memory	64 K	18 K
Rom Memory	--	18 K
Mass Storage	Floppy Disc	Cassettes
Music Keyboard	10 Octave	5 Octave
Sample Rate	30 KHZ	10 KHZ
Controls		
ASCII Keyboard	Yes	Yes
CRT	Yes	Yes
Joysticks	4	4
Pots	72	8
Hex Key Pad	No	Yes
Sense Switches	Yes	16
COST	$200,000	$1,000

Table 3 System Configurations

Capability	Hardware	Software
Monophonic Music	KIM-1, MP-P, Audio Amplifier	Music Program by Jim Butterworth
Video Display and ASCII Input	VB-1, EW-100	I/O Driver
Stand Alone Micro Computer	KIMSI, 8K RAM	Tiny Basic, Micro Ade
Polyphonic Synthesizer	2 Channel D/A Converter	Adaptation of Chamberlin Program
Keyboard Input	Aries Keyboard and Interface	Interface Software
Control Inputs	Joysticks and Pots with A/C Converter	Interface Software
Expanded Storage	16 K Prom	-----
Hardcopy Output	Centronics S 1	Interface Software
Analog Synthesizer	E MU Prototype	Control Software
Percussion Synthests	Percussion Interface	Control Software
Enhanced Computation	MM 57109	Custom Basic
Enhanced Storage	Floppy Disc	FDOS

LOW-COST MULTI-PART MUSIC PROGRAMMED IN BASIC

Dorothy Siegel
Vice President

Newtech Computer Systems, Inc.
230 Clinton Street
Brooklyn, New York 11201
(212)625-6220

<<PLAY "WEIRD" & "SWEEP" ON PRE-RECORDED CASSETTE>>

What you just heard was a sampling of some of the sounds one can produce using a digital-to-analog converter as an output device for a computer.

We used a Newtech Music Board, which is a digital-to-analog converter with audio amplifier and speaker, in an IMSAI with an 8080 processor.

We can also produce music with this system. I'd like to demonstrate a quick chronological tour of the development of the Newtech music software.

First we hear a familiar piece of music, Chopin's "Minute Waltz", played in a minute and twenty seconds. You will notice, near the end, that we vary the tempo of the music to give it more expression. We also coded in grace notes, trills, and other ornaments.

<<PLAY "CHOPIN" ON PRE-RECORDED CASSETTE>>

You just heard a square wave with a single eight-segment amplitude envelope applied to each note. The 8080 Assembly Language program that produced this wave used a short timing loop for each half cycle of the waveform, and an integral number of cycles for the duration of each note. This program read the delay loop parameters from a "score" area in memory. The "score" area was poked into memory by a BASIC program which accepted as input the 800-odd notes from the musician in the form of data statements.

In order to add more interest to each individual note, we next provided optional envelope selection. This was accomplished by simple modifications to our original BASIC and Assembly Language programs. In the short piece you'll hear next, you can hear that some notes have a short, staccato envelope, while others are legato or shaped like a crescendo/decrescendo.

<<PLAY "GODREST" ON PRE-RECORDED CASSETTE>>

The next step was multi-PART music, playing more than one voice at a time. Our technique is similar to that described by Hal Chamberlain for a different processor.

<<PLAY "DIXIE" ON PRE-RECORDED CASSETTE>>

The computer-as-musical-instrument is, like any other, only a means to a musical end. It is a tool, a device that enables the musician to express himself. And the MUSICIAN is the key to the quality of the musical experience.

In the past computers were too remote and inaccessible to most of the music community. Now, however, we have the opportunity to unite the technology of personal computing with that form of human expression called music. And I expect much to come of this collaboration in the next few years as musicians begin to explore this new resource. I'd like to see more musicians playing with these instruments, and more composers writing for them.

My own interest in this field stems from my training as a clarinetist. I've performed a number of pieces written for clarinet accompanied by such non-traditional instruments as tape recorders and synthesizers. Tonight I'll be playing a piece, in real time, with a personal computer.

For the performance I'm using three Newtech Music Boards in a Horizon computer. The music has been arranged. The notes have been entered as BASIC data statements, converted to pitch, duration and timbre parameters, and stored on a disk. Now an assembled 8080 program will sample three different waveform tables at phase increments corresponding to the frequencies of

the notes, for a specified length of time. Note that our method of sound production is software-intensive -- that is, the waveforms are COMPUTED, and the Newtech Board converts this digital representation to analog form.

We use a different waveform for each of the three computer voices. The highest voice contains the first three terms in the Fourier series of a square wave; the middle voice has the first three harmonics in equal proportions; and the bass voice consists of the first five terms of the Fourier series for a sawtooth, or ramp, waveform.

We could have put all three voices on one Newtech board (at less than $20 a voice) by adding the waveform samples before outputting them, but using three boards allows us a full dynamic range for each voice.

We used a 4MHz Z-80, instead of a 2MHz 8080, to obtain the benefits of a higher sampling rate.

The piece I'm playing is the Rondo from the Sonata in Bb for clarinet and piano by Johann B. Wanhal, who was a contemporary of Haydn and Mozart, living from 1739 to 1813. I've arranged the piano part for three computer voices.

<<<<PLAY THE WANHAL ON THE COMPUTER>>>>

HIGH QUALITY DIRECT MUSIC SYNTHESIS USING MICROPROCESSORS

Hal Chamberlin, Manager of Engineering, Micro Technology Unlimited
29 Mead Street, Manchester, NH 03104 603-669-0170

By now everybody should have had their fill of the "square kazoo" and "bee symphony" quality that seems to characterize music programs for microprocessors. If these machines are to have any musical value whatever they must at least be able to play multipart harmony with a variety of easy to listen to timbres.

One way to do this is to interface the microcomputer to a music synthesizer or other musical instrument but this requires that a large amount of specialized hardware be purchased. Furthermore, the quality and complexity of the resulting sound depends almost entirely on the characteristics of this added hardware.

The other alternative, which is the subject of this talk, is to directly compute the sound waveform desired with the computer. Theorectically this waveform may represent the combined sound of hundreds of individual instruments all with completely different timbres, etc. Actually there is no hard limit to the sophistication and complexity of the resulting sound. Lastly, except for an inexpensive digital-to-analog converter, no specialized hardware is needed to apply this technique.

Traditionally the direct synthesis technique has been implemented solely on high performance minicomputers and large scale machines. Also, the waveform computations were done at a rate considerably slower than real-time. However there is nothing inherent in the technique that prohibits its use with microcomputers or in real-time.

This talk will focus on the mechanics of the direct synthesis technique, its hardware and software needs, its limitations, and finally its implementation on a popular microcomputer system. The talk will conclude with a demonstration of a 4 voice music program, each voice having a different, arbitrary, Fourier derived waveform. It runs in 2K on a KIM-1 (6502 based) system tied to a $35 digital-to-analog converter. Another demonstration further illustrating the power of the technique will also be given.

FURTHER DEVELOPMENTS ON AN INTERACTIVE LANGUAGE FOR THE SEVERELY HANDICAPPED

Michael S. Bodner, Ph.D.
Guy M. Hoelen, M.S., M.S.I.E.
William J. Zogby, M.A., M.B.A.

Interactive Systems Design Group, Inc.
312 Highgate Avenue, Buffalo, New York 14215

The recent appearance of the personally affordable microcomputer has no doubt caused a ripple of excitement in many sectors. Interactive Systems Design Group (ISDG) believes that no where can this revolution be better applied than in the field of communication rehabilitation. When we speak of those unable to communicate, we include the severely physically disabled, cerebral palsied, mentally retarded, and those, who for a variety of other reasons, are unable to use language. The idea that once placed in the hands of the interested user, a complimentary relationship soon develops is no surprise. ISDG is attempting, through the integration of such a microcomputer, appropriate response devices (interfaces), and software reflecting a blend of current educational theory and practice to aid the appropriate therapeutic discipline in providing the same opportunity to this population.

In a previous article, the Interactive Systems Design Group (ISDG) described the early stages in the development of a computer/symbol system integration designed to aid in the alleviation of severe communication handicaps. Those handicapped by the inability to communicate range the gamut of potential with respect to the possible forms, intensities, and combinations of perceptual, cognitive and physical dysfunction. In other words, although we may characterize a communication disability, the bottom line in assisting in such a problem is the complexity manifest where each case is as unique as the individual afflicted by such a disability. As a consequence, any device intended for this population must be as flexible and comprehensive as the problem is complex. In no way are we advocating that computer based prosthesis will be the panacea of all communication disorders; rather that as we grow in familiarity with the vastness of the problem as well as the potentials present in the sophisticated application of communication rehabilitation theory and clinical practice, software/hardware combinations have shown great promise placed in the hands of competent and concerned therapists.

If we add that, once trained in their use, computer assisted prosthesis will allow for formerly impossible modes of expression in the absence of qualified therapeutic help. Originating from this premise, and upon contact with facilities serving the communication handicapped, certain key advantages were realized with respect to the unique presentation matrix serving as the core of the Blissymbolic software. They are: 1) The ability to organize symbol groups into configurations that conform with the individual's and/or institution's needs, 2) The original program matrix is not limited to the presentation of symbols, but also word **groups** for those unhindered by receptive problems or mental handicaps, 3) For the future, working off the basic skeletal structure, one could enhance its potential development through the addition of color and sound for use in areas not previously investigated.

One essential point is clear; the current configuration of the program does satisfy the three fundamental functions kept in mind during its creation, that being its broad ability to be utilized in communicative, evaluative, and educative applications. To more properly demonstrate this point, several actual client contacts shall be described: An upstate New York Developmental Center expressed a need for a system capable of assisting in the evaluation and education of mentally retarded individuals, many of whom are not able to communicate with written or spoken language and who are now utilizing the Bliss System. The software that evolved from this commission is described in the proceedings of the National Computer Conference held in Anaheim, California, June 6-9, 1978, where the symbols developed by Charles K. Bliss were incorporated into

a unique presentation process requiring a single affirmative response to key the central processor into the acquisition of and temporary storage of client selected symbols in order to facilitate communication in word, phrase, and sentence combinations. Although these clients are often multi-involved physically and mentally, the man/machine systems, once properly interfaced and timing variables customized, there is little doubt revolutionary strides could be made in language development. Furthermore, therapists are able to more specifically evaluate an individual's present intellectual level and learning ability. Thus, the Bliss Symbol System provides this institution with the communicative, educative, and evaluative materials it needed to more effectively ascertain each patient's abilities and needs.

A second situation concerned a young man in his early twenties who had experienced brain damage that limited his range of motion to two voluntary head movements. These two movements represented the only means of communication that his still active and vital mind had with the outside environment. ISDG's reaction to the problem has been the development of a program employing English words, phrases, and alphabet cradled within the previously described presentation process, allowing in combination with the appropriate interface device, a quicker and more efficient means of communication, but also an ability for this individual to write without the presence of another individual. This person, previous to his accident, possessed good language skills. The modified program became a personalized form of rapid, detailed and sophisticated communication.

As explained, when the Bliss software was first introduced, it was created for a population of moderate to severely retarded patients. However, almost immediately new applications have been suggested by concerned individuals throughout the country.

Several consultants and institutions have spoken of a need to develop procedures for testing and dealing with particular levels of expressive aphasia. According to Wepman (1976), aphasia is defined as an impairment of the acquired capacity to comprehend and use verbal symbols for interpersonal communication. From preliminary discussions, ideas for the possible application of color and tone modalities within the present program configuration holds great promise. It is the hope of all those involved that future programs will enhance the therapist's ability to deal with certain types of aphasia where current methods remain inadequate.

In its present configuration, ISDG's computerized Blissymbolic System is built around the Apple II microcomputer with 48k of RAM and the new Disk II for the apple. The disk provides not only rapid loading of this rather lengthy program, but also leads to the possibility of chaining to add additional capabilities to the system.

As previously described, the software is oriented to enable a person with limited physical control to interact with the computer through a single contact switch rather than a keyboard. The computer presents symbols, phrases, and letters to the user in a manner with which he can interact reliably, thus broadening his ability to communicate with others and limiting the frustration involved in the inability to communicate.

The key words are reliability and simplicity. A patient can be trained to save a symbol by closing a switch. It is only an affirmative response that is required, as the timed lack of response is interpreted as a negative. The pattern of the matrix in which the symbols lie will be learned with use, and the speed by which the symbol seen is run can be increased as the user becomes more proficient.

A new wrinkle in our approach is the development of binary search rules for symbol acquisition in more intellectually unimpaired patients, i.e., accident victims. A binary search through sixteen colors makes the worst case seven scans. If the most used colors are to the "left" of the chart, a patient will spend far less time scanning. This is especially useful in the alphabet part of the system in which twenty-six letters and a number of punctuation marks are provided (total of 32), leading to a binary search. It is felt that with user training, these binary searches can increase throughput considerably.

The power of the Apple Computer is extremely important in this application. In integer basic, with low resolution color graphics, we can generate symbols in sixteen colors with very high speed.

The program makes use of numerous algorithms to remember the addresses of the desired symbols, phrases, letters, etc. and the order in which they were chosen. The generation of 300 symbols, using the disk system as a loading device, does require the 48k maximum RAM in the microcomputer. It is felt that more symbols can be generated within 48k and that if one were really pressed for space, one could go back to tape loading and reclaim the 10k or so that DOS uses. The use of the systems is envisional at three levels. The major system in terms of cost and size is a "teaching and evaluation" system, institution owned and operated, in which patients in a classroom of sorts are trained in Blissymbolics and in which personalized interface switches can be tested and evaluated. Once a patient is proficient enough in the language, he will be able to make use of a "personal" system, currently thought of as a 5" color Sony TV mounted on his wheelchair, his interface switch coupled to a wireless transmitter and an Apple II loaded with a program containing the symbols he and his therapist have decided to use. The Apple can respond to a wireless switch closure just as it does in the other system configuration, sending out video imagery through an RF generator to a specific UHF channel. By giving each patient his own sending and receiving frequencies, a large number of patients can be operational at once. This method also alleviates the problems involved in moving the computer to the wheelchair.

In the case where a patient returns home, a "home" unit is being developed. Such a system will not even have a keyboard but will have a simple "start" button, which will start the system. In the Apple DOS, this is just a ROM machine language program that boots the disk, since the program that is initialized on the disk will run after the boot without the need for a RUN, carriage return to be entered.

The home unit will enable the patient to continue using the system outside the hospital using Bliss. (The communications interface board and a modem are required at each end).

For family members who do not understand Blissymbolics, the software does currently translate the symbols into an English sentence which can be printed and saved.

Currently in development is a program that will enable one to write an English sentence and have the computer translate it into Blissymbolics. The possibilities here are enormous in as much as no such capability has ever existed with the mechanical Bliss boards.

It is also possible to adapt these programs into any foreign language using English letters. Indeed this is the power of the Bliss symbols as proposed by Bliss himself.

All in all, the potential for an improvement in the condition and quality of life for these types of patients has made the development and distribution of these programs an extremely rewarding experience and has shown a unique and valuable use of the current microcomputer technology.

ISDG looks forward with enthusiasm to the potential alleviation of many present difficulties in the rehabilitation of mentally and physically handicapped people. Communication is both welcome and encouraged with those individuals possessing true concern and expertise in this area. Only through mutual cooperation and exchange of ideas can the needs of these handicapped persons be clearly identified and reacted to by new program configurations.

References

1. ARCHER, L.A., Blissymbolics - A Nonverbal Communication System, Journal of Speech and Hearing Disorders, Vol. 42 (Nov. 1977)
2. BODNER, M.S.; HOELEN, G.M., An Interactive Language For the Severely Handicapped, Proceeding National Computer Conference - Personal Computing Digest (June 6,7,8, 1978).
3. WEPMAN, J.M., Aphasia: Language Without Thought or Thought Without Language? ASHA 18 (3): 131-6,(March 1976).

OPTACON TRACKING GUIDE FOR BLIND PERSONS READING INFORMATION ON CRT SCREENS

Yvonne S. Russell, Engineering Coordinator
Susan H. Phillips, Vocational Coordinator
Sensory Aids Foundation
399 Sherman Avenue, Suite 4
Palo Alto, California 94306
(415) 329-0430

Sensory Aids Foundation funded development of a Tracking Guide to use with an Optacon Reading Machine and CRT (cathode ray tube) lens attachment. An Optacon allows blind persons to read ordinary newsprint, books, or CRT alphanumeric information without conversion to braille. SAF Vocational Coordinators identified the need for a tracking guide while working with blind Optacon users in job placements such as computer programmers, word processing operators, data entry clerks, and reservationists. Individuals in these positions are required to hold an Optacon CRT Lens Module against a vertical display screen and follow lines of characters in rows or columns without losing their place.

A Request for Proposal for the Tracking Guide was issued and the design submitted by Clement Laboratories, Mountain View, CA was selected for development. The design concept allows the Tracking Guide to be mounted on a large number of data terminal or word processor CRT displays. Using the Tracking Guide, personnel who employ Optacons equipped with CRT Lens Modules in their jobs will be able to track lines and columns of characters on the CRT display without excessive arm fatigue. In addition, they will be able to leave the camera in place at any position on the screen for later reference. A prototype unit has been demonstrated by and placed with a client who works as an airline reservationist at United Air Lines. Her evaluation concludes that the Tracking Guide significantly increased response speed and reduced fatigue.

DIGITAL INDEXING SYSTEM FOR CASSETTE AUDIO TAPE RECORDER

Yvonne S. Russell, Engineering Coordinator
Susan H. Phillips, Vocational Coordinator
Sensory Aids Foundation
399 Sherman Avenue, Suite 4
Palo Alto, California 94306
(415) 329-0430

Sensory Aids Foundation funded development of a prototype audio tape recorder system with digital indexing capability. The system will allow a totally blind person to enter digitized numerical titles at the beginning of blocks of audio recorded information. The titles will be entered from a calculator-style keyboard on the tape recorder. Later, the user could again enter the numerical title and ask the machine to search at high speed for that particular block of audio information and stop at the beginning of the recorded section. For example, a government personnel manual with numbered sections can be recorded and indexed with the same numbering system as in the ink-print version. Each recorded section could be rapidly retrieved for quick reference and would reduce or eliminate the need for large volumes of braille reference materials.

A Request for Proposal (RFP) for the Digital Indexing System was issued and Telesensory Systems Inc., Palo Alto, CA, proposed a digital indexing system which met the requirements of the RFP. The system will record the digital indexing information directly onto a tape cassette, which will allow reproduction of indexed tapes, necessary in multiple job placements where the same reference materials are used. Direct indexing onto the tape will eliminate the need for extra braille cross-reference indexes of information located on each tape.

POTENTIAL APPLICATIONS FOR SMALL COMPUTERS IN THE PRACTICE OF MEDICINE

ABSTRACT: As software for office practice billing becomes more reliable and hardware less expensive, health professionals will be among the first to widely adapt small computer systems, yet few will receive the full range of benefits possible unless innovative applications are made available in an easy-to-use form. Now is the time to get started, preferably in a language that is machine independent and obsolescence proof, such as Pascal. Markedly increased reliability of histories, cost savings, reduction of drudgery, improved diagnostic and therapeutic skills, closer physician-to-patient communication, and ease of record keeping and documentation are all achievable.

INTRODUCTION: Although for a generation the practice of medicine has been associated with increasingly sophisticated technology, computer applications for diagnostic and therapeutic aspects of medicine have been relatively few and not widely adapted, as opposed to the somewhat more widespread utilization of data processing for the business side of health care, especially in hospitals. Most billing in private physician's offices is still done by hand, in spite of an increasing influx of onsite and distant billing services, largely handled by outside contractors, since many MDs dislike the business aspects of private practice and prefer to hire others so "all I have to do is practice medicine." Further, at least until recently, practice management consultants were rife with harrowing tales of gross errors in billing and financial record keeping with such contractors, due to "infant" software that had not yet been debugged. However, as hardware becomes less expensive and software more reliable (particularly in the area of error-detection routines, improved verification of data, and more difficult to carry out yet more easily detected embezzling), it is my belief that and other health professionals will be among the first to recognize the value of computers in small business. Potential applications of computers in medical offices may be frustrated, however, if vendors succomb to the temptation to make their systems as incompatible with others as possible, in order to create a captive market. This danger may be overcome if useful software is created and made available early in this process, ie now, with the specific aim of being as widely applicable as possible.

This paper, then, will describe a number of exciting, easily realized applications for computers in the practice of medicine. Because of the intense focus that business and financial computing is already receiving, I will avoid these areas for now, except to say that quick commercial success of a software system for health professionals would be greatly enhanced by the inclusion of such programs as a complete personal income tax information and record-keeping system, complete with updates as the law evolves.

HARDWARE: I am assuming that any commercial system would have available a CPU with adequate (>48k) memory, a disc system with easily changed discs (for now, all software would be supplied on disc), one or more CRT terminals (memory mapped or standalone), and a printer. Once the video disc technology or other inexpensive, high-density medium of software exchange became available, this would be included. The one nonstandard item I would feel is important is an appropriate data entry system for patients or other keyboard-naive people, including physicians. Clearly the most reliable and rapid data entry system when one is working with a menu (multiple choice entry versus raw alphanumeric strings) is a touch system overlaying the CRT, where one can simply touch the answer directly. A simple, home-brew series of phototransistors and infrared light emitting diodes allowing a 16 x 16 grid (256 points, one byte of data) was recently published in Byte, and conductive or capacative transparent CRT overlays have also been used. I feel that the success of computers for lay people depends in large part on easy, simple, reliable human-machine interfaces.

SOFTWARE: One should settle for nothing less than total machine independence, so that obsolesence will depend only upon the program itself. Further, it is unacceptable to write in a language so slow, so poorly designed, and so widely altered as BASIC. I am thus an unabashed enthusiast of UCSD Pascal, which is an inexpensive system of great power. Pascal utilizes an intermediate stack-oriented interpreter that can be easily implemented in any machine whose word length is a multiple of 8 bits. All the output of the compiler and other parts of the system (now including a BASIC compiler) is intended for the interpreter. Since the interpreter performs all I/O as well as all arithmatic functions (including exponentiation, trigonometric conversion, etc), new generations of hardware can be incorporated with simplicity, leading to noticable improvement in speed. Further, the code is more compact than native microprocessor code, and it resembles machine language in that a source program cannot easily be recovered from a listing of its interpreter code. The interpreter itself occupies about 5K of memory.

APPLICATIONS

DIAGNOSIS

1. A comprehensive review of systems history (eg, "Do you have headaches?"..."Do you frequently become short of breath?"...etc) is probably the best means of detecting occult forms of illness. Performed in depth by a physician, it takes approximately five to twenty minutes, depending on the patient, plus some additional time to document in the chart. Thus, it may cost the patient as much as $30 for this portion of the evaluation alone. Further, few physicians take the time to delve into areas of psychological, social, sexual, or other nonphysical difficulties, in spite of the fact that it is well known that more than 70% of the patients visiting the average physician's office have complaints that are primarily not physical in origin. Diet, life stresses, and work history are also rarely covered.

A variety of self-administered health questionnaires have been devised, many based on the Cornell Medical Index Questionnaire (MIQ). However, the number of positive responses to the MIQ correlates best with the level of anxiety, not illness, and there is no internal validation to make sure the patient understands the questions and is not faking most of his responses. More important, everyone must answer the same list of questions, since branching into areas of difficulty is not possible.

Clearly, this area is a tailor-made opportunity for computer questioning, and some work has already been done.

2. One of the most harrowing aspects of being a general physician (including general internists and pediatricians) is the difficulty of keeping up with advances in diagnosis and treatment. When I was in private practice, I would estimate that 50 pounds of educational material crossed my desk per month. Often, what I did learn was difficult to assimilate into my practice. My fear, of course, was that I would miss that exceptional patient with a rare illness, treating him or her in blissful ignorance as if he had something else.

It turns out that many areas of medical diagnosis can be expressed in algorithmic form quite easily. In fact, physicians who have created diagnostic computer programs have found out that once debugged, the program is generally smarter than they were in practice, since the algorithm was conceived in the peace and quiet of one's library, where one can think slowly and clearly. Obviously, if experts in a given area are responsible for creating up-to-date diagnostic algorithms which are then converted to self-prompting computer programs, one could immediately upgrade the standard of care for whoever used the program. The output of the program would be a summary of data collected, a listing of the likely diagnostic possibilities, plus suggestions for further workup. Further, the algorithms could be structured to avoid calling for unnecessary and expensive laboratory and X-ray procedures, and their printouts would lend themselves easily to iron-clad defenses against inappropriate malpractice

actions. Thus, it is possible that they could promote substantial savings in the cost of medical care.

PATIENT EDUCATION

Computer-assisted instruction is already a fact, but I do not know if it has been applied to patient education. In addition to the obvious cost savings with good programs, the utilization of computers could assist with two well-known problems: Patients have been shown time and again to forget what was told them, and their memory is aided when one writes it down. Also, documentation of adequate patient education is now a legal requirement of all surgical therapy. The major disatvantage of the machine-assisted approach is that human caring greatly aids understanding and retention as well, not to mention patient compliance with the recommendations.

TREATMENT

Much of the discussion of computer aids to diagnosis also pertains to certain areas of therapy as well, especially in complicated metabolic problems, etc.

RECORD KEEPING

Often, office records contain much that is routine, particularly the physical examination and certain sections of the history. A menu-oriented program used by the physician could type out a detailed summary of the evaluation, or a file containing the usual format and content of the report could be edited by a secretary in a word processing environment, greatly shortening the time required for documentation.

Clearly, word processing itself would be an invaluable and essentially free service, requiring only a printer with a typeface of adequate quality.

ATVANTAGES AND DISATVANTAGES OF COMPUTERS IN MEDICAL PRACTICE

One of the most common cries of the consumers of health care is that physicians and others don't spend enough time with them and seem exclusively concerned with technical matters. Computer aids to medical practice would probably have very different effects in different practices; it is likely that a few health professionals would utilize their increased efficiency to spend even less time with their patients. It is my hope, however, that as a rule the opposite would be the case, and that as the technological role was shared more with the technology itself, a humanizing effect would occur between doctor and patient. Certainly, a number of patients may object to interacting with a machine so that the routine aspects of their histories may be obtained. Yet at least one study has shown that patients understand the value of computer assistance and are willing to utilize it. Finally, there is the question of the relative validity of computer-aided diagnosis, history, etc, as compared with a well-trained health professional. It is my understanding that to date, computers have fared as well as MDs in such areas as electrocardiogram interpretation, complicated diagnostic dilemmas, etc. It certainly remains to be seen if this role can be reliably extended as noted above.

USE OF COMPUTER AND BIOFEEDBACK IN PSYCHOLOGICAL LABORATORY FOR TREATEMENT OF EMOTIONAL ILLS

Russell N. Cassel, Ed.D.
PROJECT INNOVATION, Box 566,
Chula Vista, California 92010

Use of the microcomputer and biofeedback equipment promises to improve greatly the 'state of the art' characteristically associated with 'mental health delivery services.' A critical aspect of such services deals squarely with the emotions and affect qualities of individuals. Precise objective data bearing on the state and changing nature of one's emotion had been direly needed, but for the most part has been sadly lacking in the 'helping relationship' arena. For many years the Galvanic Skin Response (GSR) along with other data has served effectively in 'lie detection' but little or no use has been made of it in connection with the further detection or treatment of emotional ills.

For nearly a decade PROJECT INNOVATION has sought to implement and wed the use of the computer and biofeedback instruments in a functioning Psychological Laboratory. This paper describes one aspect of this program where real objective evidence is sought depicting states of emotions for persons in 12 well defined areas of one's life-space. Such evidence is believed to be critical for both diagnosing and treatment of emotional ills, and where the newly emerging concept of "Clinician Accountability" is ordered to the scene. Many aspects of the newly conceived idea are no in 'operational readiness', and await the completion and standardization of others for use in a planned Psychological Laboratory somewhere in the Chula Vista, California area.

Equipment

This aspect of the Laboratory weds several mamor pieces of equipment into a single functioning whole: (1) microcomputer, (2) galvanic skin response box, (3) pulse monitor box, and (4) an automatic 2x2 slide projector.

Microcomputer

This is a Vector ++ developed and manufactured by Vector Graphic Inc., Westlake Village, California 91361; with a Zilog-80 CPU; and two Micropalis Floppy disks (II) by Micropalis, Canoga Park, California 91303.

Galvanic Skin Response

The Model 800 GSR/BSR Electrodermal Instrument manufactured by Acquarius Electronics, Albion, California 95410 (now out of business).

Pulse Monitor Box

The Cardiotach manufactured by Banke Engineering, 5565 Marengo, La Mesa, California 92041 with a liquid crustal display unit is used for this function.

Slide Projector

The Kodak Carousel Slide Projector, Model 570 by Eastman Kodak, Rochester, New York 14650 is used for this function.

Diagnostic Areas

Twelve areas of the life-space have been defined as a basis for determining emotional producing stimuli. Each area is presumed to be independently organized, and sufficiently common from person to person to have generalized meaning for both diagnoses and therapy purposes. In many ways the areas parallel the developmental nature of man, and ranging from early home and family relata to nature and environment:

I. Home and Family
II. Affiliation and Inner Development
III. Peer and Social Relations
IV. Authority/Responsibility and Law
V. Learning and Self-Actualization
VI. Romance & Psychosexual relations
VII. Risk taking and Sports
VIII. Biological Needs and Health
IX. Recreation/Avocation and Travel
X. Aesthetic -- Art and Music
XI. Productivity and Economics, and
XII. Nature and Environment

Operation

The electrodermal unit, pulse monitor and the slide projector are interfaced with the Vector ++ microcomputer. The microcomputer uses the Micropalis BASIC language for the principal operation, and with assembler language to record input from the various instruments. All signals carry 'optical isolation' as a means for both protecting participant subject from the electrical current used in computer, and for reduction of noise from the electrical lines.

Different sets of 2x2 slides are presently prepared to afford the typical 'before and after' application in relation to diagnosis and treatment regimen. Slides are arranged by areas of the life-space, and with 12 slides for each different area included. Typically, the computer is programmed to flash a new and different slide each 15 seconds, and with a time factor that may be adjusted for longer or shorter periods of time. The subject is asked to view each slide and reflect on the content as if it were related to their own particular life situation.

Each of the different sets of slides are

beingnormed for different populations of varying age and other dimensions. Slides are retained or eliminated on the basis of the degree to which emotional responses are reflected for the norm groups. Slides within areas of the life-space are distributed so as to define more precisely the nature of emotional producing stimuli.

Critical Emotional Areas

The microcomputer program provides for an emotional setting on the computer ranging from low to high, and in relation to the Basic Skin Response (BSR) from the electrodermal unit. For persons whose case history reveals them to be highly emotional in nature, such setting might be rather high; while for others less emotional in nature, such setting might be quite low.

Whenever the BSR for a particular picture exceeds the critical setting, the computer halts the projection of new slides and retain the emotion producing slide until the emotional response precipitated by the BSR restores to normal. After the normal emotional condition is restored slides are again advanced at the 15 second interval. A clock board in the microcomputer affords the programming of length of time necessary to gain self-control-the time from when the critical level of the BSR is reached, until the normal level is again restored.

Individualized Treatment

For those patients involved in diagnosis and planned long-term treatment, slides may be utilized from own individual relationships, i.e., husbands/wives, family, work, play, avocation, etc., and for each of the life-space areas involved in the general pattern utilized. Comparative assessments are planned to determine the disparity between the individualized slides and the general standard ones.

Assessment Information

A computer print-out depicts a profile for the 12 areas of the life-space for both the BSR and the Pulse of individual involved. This includes both the mean (m) and standard Deviation (SD) for a given period of time (usually about 7 seconds, but may vary). In addition, there is a list of slides, by specific life-space area, that produced BSR's greater than the critical setting, and with the length time necessary to regain self-control by particular life-space area. Here the M and SD is provided for the slides included for that area. An indication is also given of the level for the 'critical' setting used to establish such critical areas. A copy of the print out form is indicated in Figure 1.

Therapy Monitoring

Provisions are planned so that the therapy of a particular patient could be monitored through the use of phone lines attached to the computerized Psychological Laboratory. Here visual displays on a Cathode Ray Tube (CRT) or actual print-outs may be afforded. Thus, there would be immediate knowledge of progress in relation to the self-control established by a patient or client through such procedures.

PROJECT INNOVATION'S
COMPUTERIZED EMOTIONAL ASSESSMENT SYSTEM

This system was planned by Dr. Russell N. Cassel, Diplomate in School Psychology ABPP, Box 596, Chula Vista, California 92010, with the electronic and programming by Marc Yaxley, Computer Metrics, ElCajon,California. It is comprised of the following major equipment: (1)Vectorgraphics ++ Microcomputer with 64K of memory,(2)GSR Biofeedback instrument, (3) Pulse monitor, and (4) a Kodak 2X2 slide projector.Generally, the computer records emotional responses to picture slides from 12 areas of one's life space, identifies slides where critical emotional responses occur, and assess length of time to gain self control for all such critical slides.

Areas of One's Life Space	Pulse Mean	Pulse SD	BSR Mean	BSR SD
I. HOME AND FAMILY.				
II. AFFILIATIONS & INNER DEVELOPMENT . . .				
III. PEER AND SOCIAL RELATIONS.				
IV. AUTHORITY/RESPONSIBILITY & LAW.				
V. LEARNING AND SELF-ACTUALIZATION.				
VI. ROMANCE & PSYCHOSEXUAL RELATIONS. . . .				
VII. RISK TAKING AND SPORTS				
VIII. BIOLOGICAL NEEDS & HEALTH				
IX. RECREATION/AVOCATION & TRAVEL				
X. AESTHETIC - ART AND MUSIC.				
XI. PRODUCTIVITY AND ECONOMICS.				
XII. NATURE AND ENVIRONMENT				
TOTALS				

Critical Slides from Life Space	Pulse Mean	Pulse SD	BSR Mean	BSR SD
I. HOME AND FAMILY ( )				
II. AFFILIATION & INNER ( )				
III. PEER AND SOCIAL RELATIONS( )				
IV. AUTHORITY & LAW( )				
V. LEARNING - SELF-ACTUALIZATION( )				
VI. PSYCHOSEXUAL ( )				
VII. RISK TAKING ( )				
VIII. BIOLOGICAL NEEDS( )				
IX. RECREATIONAL ( )				
X. AESTHETIC -Music & Art( )				
XI. PRODUCTIVITY ( )				
XII. NATURE & ENVIRONMENT( )				
TOTALS . . .				

Critical Setting_____ GSR Long Term _____ GSR Short Term _____

THE MICROCOMPUTER AS ANTIDOTE
Medical Data Base Applications in the Home and Office: Accidental Poisoning Information; Medical Journal Abstracts

Roger O. Littge, M.D., M.S.P.H.
Medical Consultant to the Lawrence Hall of Science
Berkeley, California

Vice-president, Berkeley Medical Data Associates, Inc.
Box 5279 , Berkeley, California 94705 (415) 653-6707

Abstract

Work is proceeding on two interesting projects which utilize the rapid text indexing capability of the personal computer in applications of medical information storage and retrieval. The examples are: (a) rapid determination of appropriate steps to take in the event of accidental poisoning and (b) scanning of medical journal abstracts indexed by key words.

Poison Information

The first of the above problems (information retrieval for determing the appropriate action in the event of an accidental poisoning) consists of creating a data base of poison and household products information as a combination of coded and non-coded character strings which then may be accessed by a response-recognition program, written in BASIC. The recognizable responses would be grouped as to their properties which imply a particular set of therapeutic steps, e.g. all of the various hydrocarbons would be grouped together which had toxicity to the pulmonary tree (trachea, bronchi, brochioles and alveoli) of sufficient degree to warrant serious hesitancy in inducing vomiting; all the penicillins would be grouped as to their likelihood to cause various side effects (rare or benign as they are), all the various brand names which share the same active poison could, with several lines of program and great ease of modification be included, with the responses limited to maneuvers and remedies available at the particular site of the computer. Studies at the Stanford Research Institute indicated that the time necessary for printout of the appropriate response to questions on the order of:

"Is hypotension a possible interaction between guanethidine and anesthetic?"

"Can aspirin interact with probenicid?"

"Can aspirin interact with any other drugs?"

was approximately 0.97 sec. (plus the time to print "Yes, guanethidine and anesthetic produce hypotension") in the first case, 2.74 seconds (plus ...) for the second and 8.09 sec. (plus) for the third. Thus for table look-up programs of this sort, where the possible number of responses of the program is really rather limited, the access times will be the sums of very short computational delays and whatever time the memory storage takes to find and read the pertinent addresses for a relatively limited number of responses. Disc storage has the obvious advantage of speed in handling from 90K to 1MEG bytes of tables and text, while cassette-handling systems pay a price of access time measured in minutes (instead of msec. for the disc) for the greater text and tables memory they allow. The system on which the developmental work for this project was done consisted of the 8080A based Sol 20 mainframe with Northstar minifloppy disc drive. While heeding the advice of Slack (1) and others to restrict storage of the more critical components of the directories to RAM storage and relegate more copious text, assuming it is not needed for computation and is merely the character string to be printed or displayed upon branching to the most appropriate response, to non-RAM mass storage devices, the disc has such rapid access times that it can be used to store some

of the character strings needed to recognize inputs to the keyboard. It would be eventually necessary, in order to store the large number of various names under which poisons would be found, to have at least several hundred text-dependent or text-recognition words stored in a program built around recognition of the actual words typed into a keyboard. A comment on the wisdom of putting entire words into memory might be warranted, since multiple choice entries on a keyboard could also be used to select proper responses and could save memory. For the looking up of an appropriate action in the case of a poisoning in the home, it may be easier, considering the psychological setting, to reduce to a minimum the amount of thinking and clear-headedness necessary to implement the program. The program which might thus be most appropriate for a home would be text oriented, while, for an emergency room, physician's office or poison control center a number oriented program which could store a larger number of entries to cover the wider variety of poisonings likely to be seen in such places would be attractive. Number oriented programs being developed now would flip to chosen "pages" of text to arrive at more detailed text bearing on a particular problem. Each "page" of text initially would be sized to fill the CRT screen with choices of increasing specificity, giving numbers to enter into the keyboard. Typing in the appropriate number would then very quickly branch to the more specific text "pages" which would contain either more detailed choices of subtypes of poisons (with associated numbers to lead to the final text information) or the final text information itself. One example of the implementation of each of these types of programs is given in Appendix A.

The version of this program for use in the home is being developed by Berkeley Medical Data Associates, and should be available in early 1979. The initial plans are to release it in low cost cassette form for the TRS-80, the Comodore PET, the Apple II, and in CUTS format for the SOL-20 or other microcomputers using Processor Technology's CUTER cassette interface. Disk based versions are being developed for the home, and in a more sophisticated form, for medical offices and emergency facilities.

Journal Abstracting

The second project (storing and indexing of journal abstracts) takes advantage of an abstract generating and distributing service which is already mailing concise, indexed and coded abstracts in the U.S.(2). Their indexing and abstracts are not available on computer compatible media, but they have discussed the matter in their own organization and have agreed to assist our group in bringing this service to those who have invested in microcomputer hardware. As mentioned above, the indexing and cross-indexing for this project is similar to many other applications in the general area of literature searches. MEDLARS has used its codified system of key words for years to assist in the location and retrieval of new medical knowledge, but it has not been adapted, to my knowledge, for microcomputer systems operating on-line or stand-alone (as with on-site disc or cassette storage). More detailed information on the indexing being used is available in Appendix B.

Appendix A

The first case of the program being used at home for rapid determination of the best first steps to take after a poisoning can be discussed for both disc- and cassette-based systems likely to be found in the home. Currently, the largest selling personal computers are being marketed with cassette drives for reasons of price competition and perhaps the desire to supply "more bytes for the buck", while it can be safely assumed that disc drives will become very common as home users upgrade their systems. Tape systems will fairly easily store background data for each person in the family (Date-of-birth, height, weight, allergies or hypersensitivities, present medications and recent or ongoing treatments, pertinent medical history and hard-to-remember names and phone numbers) though retrieval of distant files may take several minutes. It would be prudent, thus, to have the tape or disc easily available if the real need were to arise to use it, and, in the case of the tape, to prepare it such that the most critical and common possible poisonings would be most quickly accessible, near the beginning of the tape where the monitor portion of the program would be stored and from where the tape would have to move to

retrieve the desired text. Power-on, bootstrapping and appearance of "What type of ingestion?" (or whatever initial message the particular home would like) takes as little as seventeen seconds if the disc is handy for a disc system, and would take from at least two minutes for tape-dependent systems, less if a larger PROM is used to store more of the tape-recognition and initial search program. An example of a typical "page" from which a distraught mother, for example, could pick a catagory to locate the text of the best thing to do for her child (who just swallowed the rest of whatever was in that bottle in the kitchen) is shown in Figure I. Keyboard entry of numbers can quickly be recognized to cause display of either more pages of more detailed menu or the final text display relating to that particular type of poisoning. Each page of final display could include the information found on the page called "numbers," i.e., phone numbers of the nearest emergency room, family or particular physician and nearest poison control center.

Appendix B

The abstract service with which our group has the most experience is located in Indiana, and, for a number of medical specialties they provide independent abstracts of several dozen journals, issuing them on 3" by 5" file cards. The number of journals abstracted varies depending on the specialty, ranging to over sixty. Their initial indexing is illustrated in Figure II for Family Practice. This is supplemented by the following suffixes as appropriate:

i. General
ii. Etiology, Pathophysiology
iii. Diagnostic Signs and Tests
iv. Complications and Prognosis
v. Treatment and complications thereof

The first heading under Master Index, General Topics, is further subdivided as shown in Figure III, and each these is also subdivided, giving an obvious method of finding abstracts bearing on more and more specific areas. The abstracts are not yet available on computer-compatible media and the abstract service has deliberated on providing such, but our group is also pursuing setting up such a system on minifloppy disc with a text scanning program to pick out key words and display or print the citations and/or abstracts.

Acknowledgements

The author wishes to thank The Lawrence Hall of Science for their generosity and assistance in the use of their computer facilities and staff time.

About the Author

The author completed his M.S.P.H. and M.D. at the University of Missouri, Internship at Riverside General and Family Practice Residency at Contra Costa County Hospital in Martinez, California before becoming a medical consultant to the Lawrence Hall of Science Computer Operations Section. He is presently a practicing physician at the Kaiser Foundation Hospital in Walnut Creek, California, and a Vice-President of Berkeley Medical Data Associates, Inc., a private consulting firm specializing in microcomputer applications in Health Care.

References

1. Slack WB, Hicks GP, Reed CE, et al: "A computer-based Medical History System", New England Journal of Medicine, 274:194-198, 27 Jan 1966.

2. Reference and Index Service, Inc. Indianopolis, Indiana.

Garage

Gasoline-like stuff.....(TYPE).....1
Thick or heavy oils,etc.....2
Paint, stains 3

Other ...TYPE the product name

Kitchen

Caustics, bases.(lye, Drano).....4
Soaps, detergents.....5
Insect liquids, etc.....6
Other...TYPE the product name

Bathroom, etc.

Medicines.....7
Cleansers.....8
Bowl cleansers.....9
Other...TYPE the product name

NUMBERS TO CALL.....0

FIGURE I: Home poison information

Master Index

I. General Topics
II. Dermatology
III. Diseases of Head, Neck and Respiratory System
IV. Cardiovascular System
V. Hematology
VI. Gastrointestinal System
VII. Liver, Biliary and Pancreatic System
VIII. Collagen Disease, etc.
IX. Metabolic Disease
X. Endrocrinology
XI. G-U System
XII. Musculoskeletal System
XIII. Nervous System
XIV. HEENT
XV. Psychiatry
XVI. Infectious Disease
XVII. Misc.
XVIII. Hereditary and Congenital Disease
XIX. Therapeutic Agents
XX. Trauma
XXI. Transplantation
XXII. Pediatrics
XXIII. Plastic and Reconstructive Surgery

FIGURE II: Journal Abstract Categories

I. General Topics
  A. Disease Concepts
  B. Systemic Manifestations
  C. Metabolism, Nutrition and Diet Therapy
  D. Social and Economic Factors

FIGURE III: Breakdown of "General Topics"

MICROCOMPUTER FEASIBILITY IN THE HOSPITAL SETTING:
A Microcomputer System as a Cost Effective Expenditure in Computer Application Feasibility Studies

Robert C.A. Goff, M.D. Research Associate
Children's Hospital Medical Center of Northern California
51st and Grove, Oakland, California 94609 (415) 654-5600 ext 276

President, Berkeley Medical Data Associates, Inc.
Box 5279 Berkeley, California 94705 (415) 653-6707

Abstract

Hospital administrations are understandably reticent to adopt microcomputer applications in settings which have always required the power of at least a mini-computer. The primary question seems to be whether or not a microcomputer is, in fact, capable of fulfilling the requirements of any of these tasks. An approach to this administrative impasse envolved the purchase of a complete microcomputer system equipped with a portable custom designed desk. The purchase of this single computer was based on its ability to perform at least one cost effective, though simple task. Its primary use, however, is for development of feasibility demonstrations of microcomputer applications throughout the hospital. The operating assumption is that as each application is demonstrated to be feasibile on a microcomputer, the responsible administrative sub-unit of the hospital will then purchase the requisite hardware for permanent installation in that application. The projected applications and general acceptance of this concept are discussed. The system described was implemented at Children's Hospital in Oakland, California. [Key words: **medical computers, computer feasibility studies, hospital administration, hospital economics**]

Introduction

Microcomputer systems are now capable of managing many of the computer applications within a hospital. Such implementations, however, are not very much in evidence. This is not for lack of appropriate hardware, nor because of the shortage of suitable software (though the software is hard to find), but rather a result of the reticence of hospital administrations to seriously consider microcomputers as a realistic alternative to the larger mainframes. This is understandable in light of the scarcity of examples of successful microcomputer applications in hospitals. It is also understandable when one considers that at most hospitals the computer consultant (if one exists in the institution) has a full grasp of the capabilities, shortcommings, and costs of larger computers, but little, if any knowledge of microcomputers. This is augmented by the contrast between vendors of large computers and vendors of microcomputers. The former utilize impressive advertizing investments to acquaint hospital administrators with their wares. The latter, of course, make almost no attempt to do so. This discussion focuses on the issues of the desirability of microcomputers in hospitals, and an approach to demonstrating their feasibility.

The Hospital Computer Environment

What tasks are currently performed by computers within the hospital setting? Accounting, billing, payroll, inventory, and census come quickly to mind. Many central laboratories have computerized their data. Pharmacies have automated their in-house dispensing of medications, and check for potential drug interactions by computer. The list

of applications is really quite long. Can a microcomputer handle these diverse tasks? Of course not. Not single handedly. But herein lies the strength of the microcomputer. Because a microcomputer system can be cost effective when applied to only one or a small number of simultaneous tasks, it provides immense advantages in terms of flexibility and implementation time.

Consider the large mainframe installation. Its cost requires that it be shared among a sizeable number of simultaneous tasks, perhaps even timeshared outside the hospital. As a result, its operating system must be capable of handling all of these tasks. Its applications programs must all be implemented before the installation will become cost effective. Two years is a reasonable estimate of the time required from planning to completion. The hospital must be prepared to commit an impressive amount of the institution's financial resources to computer purchase contracts which, more often than not, do not guarantee that the final installation (two years later) will meet their needs or, for that matter, be function reliably (1). The installation will require a substantial continuing costs for hardware maintenance as well as software updating and modification. The total dollar commitment is so large that changing systems, if the first is found to be unsatisfactory, is almost impossible.

A microcomputer, on the other hand, may be cost effective even if dedicated to a single task. Its total cost, in any particular application, is comparable to the annual salary of the lowest paid employee. In several applications being developed by the author, it may easily pay for itself within a few months. The continuing maintenance costs are minimal, and software updating may be successfully undertaken by any individual with a knowledge of the rudiments of computer processing. If, after a year or so, it is found to be unsatisfactory, its abandonment is not hampered by the need to restructure the data processing needs of the entire hospital. Development time for any particular microcomputer application ranges from one to six months, in most cases, and allows the finally developed implementation to more closely approximate the projected needs. It bypasses the nearly impossible chore of projecting the needs of the whole institution two or more years into the future.

Feasibility Demonstrations

After an institution has made the decision to install a large mainframe computer, the administration must set about the task of determining the feasibility of various hardware and software configurations for each of the projected applications. In some instances, similar applications at other institutions may be inspected to determine the strengths and weaknesses of their specific configurations, though such information is sometimes shrouded by the pride and embarrassment of administrators who have committed their hospitals to costly contract errors. More often, however, feasibility must be demonstrated "on paper" by attempting to simulate the final installation by outcome analysis techniques. While these estimates may be reasonably accurate for any particular application, their inherent error is cumulative for the system as a whole, and may therefore fail to predict fatal shortcomings of the contracted hardware and software.

The situation is much simpler for microcomputers. Since the needed feasibility study encompasses only one application, the likelihood of a valid conclusion is greater. An added benefit is that if the system ultimately purchased fails to meet the demands placed upon it in its intended application, then it is a relatively easy adjustment to assign it to a different task.

The Feasibility Computer

At Children's Hospital in Oakland, California, the problem of demonstrating the feasibility of various microcomputer applications was solved by purchasing a complete microcomputer system. This system took on, as its sole application, the job of demonstrating the feasibility of microcomputer implementations in several areas of the hospital. Its purchase was initially justified by demonstrating "on paper" that it was capable of performing a single cost effective task -- that of

monitoring the chargeable items used within the Intensive Care Nursery. If it could not be shown to be feasible in any other application, then it would be assigned to that dedicated task, and pay for itself in about three to four months.

The hardware and software were selected to be most compatible with the array of possible applications which it would test. Since one particular application required a triple floppy disk drive, three North Star Microdisk Drives were selected. The computer is a SOL-20 with 48K bytes of RAM. A Diablo 1610 receive-only graphics printer was chosen to enable the development of graphics output in certain of the applications. The software included the North Star DOS and North Star extended disk BASIC, Michael Shrayer's Electric Pencil II text processor, CP/M on North Star from LifeBoat Associates, and Microsoft FORTRAN 80. It was felt that this broad selection of software would enable the system to find maximal use.

A custom desk was constructed by a local cabinet shop, at a cost of about $160. The desk was designed with four heavy duty casters, and holds all of the hardware, a complete box of full width tractor feed paper, and has several compartments for easy access to manuals and a supply of floppy disks. Its height is that of a typing desk, and underneath is a hospital-grade electrical outlet strip so that the entire mobile unit may be plugged into a single outlet. Because of the easy mobility of the complete system, it may be wheeled to the site of any feasibility demonstration without the fear of transporting the numerous hardware components. In addition, the demonstration site does not need to be cleared of existing equipment to make room for the temporary computer. The mobile desk is considered a vital component in the use of the feasibility computer.

At the outset of this project, a list of possible feasibility demonstrations was drawn up by a small committee of administrators, physicians, and the chief biomedical electronics technician. The list was dictated by the specific current needs of the hospital, and included the following:

1. a monitor for the two Corning Blood Gas Analyzers;
2. a data base manager for the Neonatal Follow-up clinic;
3. scheduling of the more than 120 nurses on the Intensive Care Nursery staff;
4. on-call scheduling for the Resident house staff;
5. a bedside data base retrieval system for the Intensive Care Nursery;
6. an inventory and billing system for the medical materials used within the Intensive Care Nursery.

The list was only a guideline, and could be modified as needs changed, however the priority item was the monitor for the blood gas analyzers. For this particular application, an additional I/O board was purchased, with the provision that it would remain with the system eventually installed in the blood gas lab.

The obligation of the feasibility computer system is only to demonstrate that a particular application is feasible, after which the responsibility falls upon the clinical area envolved to purchase a new microcomputer system to be dedicated to that application. The feasibility computer then moves on to other demonstration projects. Part of the process of demonstrating feasibility is the development of at least skeletal software to perform the task being tested, so the chore of applications software development is also born by the itinerant system prior to the actual feasibility demonstration.

The cost of the system was approximately $10,000 including the hardware, 100 floppy disks, all the system software, the desk, and a supply of expendables. Most of the finally implemented dedicated systems should cost about two to three thousand dollars less, since most do not need a printer of the Daisy-wheel variety, and will need only one set of system software. The special interfacing required for some of the applications is only about $300.

The hardware and software were purchased together from a local retail computer store which was selected for its reputation of excellent hardware support and willingness to provide assistance in the planning of unusual applications. In lieu of a maintenance contract, buying from a responsible retail dealer within the vicinity of the hospital can be invaluable.

Results

This approach has not been applied for a long enough period to determine its general impact, but it has resulted in considerably greater interest on the part of many hospital departments in the possibility of microcomputer applications. As more and more microprocessor controlled biomedical devices are placed on the market, the potential users of this equipment are looking more toward microcomputers for processing the data output.

So far the system has developed the following:

1. a fully implemented monitor for the Corning Blood Gas Analyzer (2);
2. a feasibility demonstration of a patient bedside data base retrieval system (still under development) for the Intensive Care Nursery (3);
3. a complete user's manual for the hardware and software of the feasibility microcomputer system itself, to facilitate continued use of the system in the future;
4. an 8080 assembly language program for translating North Star BASIC to CBASIC (still under development);
5. a tabulation program for use in chart utilization review.

As further applications are completed, their outcome will be presented at this, and other forums. In June of 1979, a study of the real cost and results of this project will be concluded, and the results published.

Summary

An approach to feasibility demonstration of microcomputer applications in the hospital setting has been presented, with specific reference to the feasibility microcomputer system in use at Children's Hospital in Oakland, California.

Acknowledgements

The author wishes to express his gratitude to Mr. Michael Lehey, Dr. Barry Phillips of Children's Hospital -- Oakland, for their efforts to bring about this project, and to acknowledge the support of Children's Hospital Medical Center for providing the funds for the system discussed. To Mr. Loren Lewis, chief biomedical electronics technician at Children's Hospital, must go the credit for his design of the mobile computer desk and for his significant technical contribution to the continuation of this endeavor. The author must also commend Mr. Peter Hollenbeck and his staff at the Byte Shop of Berkeley for the high quality of their hardware support, and for their enthusiastic assistance.

References

1. Brandon DH, and Segelstein S: **DATA PROCESSING CONTRACTS: Structure, Contents, and Negotiation**. Van Nostrand Reinhold, New York, 1976. ($34.50)

2. Goff RCA: **Microcomputer Applications For Biomedical Instrumentation: A Monitor for the Corning M-175 Blood Gas Analyzer**. Proceedings of the Third West Coast Computer Faire. 11/78.

3. Goff RCA: **The Bedside Microcomputer in the Intensive Care Nursery**. Proceedings of the Second West Coast Computer Faire. 3/78.

A COMPUTERIZED CLINICAL SUPPORT SYSTEM AND PSYCHOLOGICAL LABORATORY

Russell N. Cassel, Ed.D.
PROJECT INNOVATION, Box 566
Chula Vista, California 92010

Almost since the inception of modern psychology the association of the Psychological Laboratory with the study and analysis of animal behaviour has been legion. Indeed, it is as if the understanding of animal behaviour would in some mysterious way enhance our understanding of man. Nothing could be farther from the truth; for humans and animals are as different from each other as day is from night. Failure of the psychologist to focus squarely on human behaviour in the Psychological Laboratory has forced the clinician in a helping relationship to fabricate substitutes in both the diagnoses and treatment processes borrowed from antiquity of a thousand years ago. Here subjective judgement combined with clinical intuition have subjugated the state of the art in our helping relationship arena to that of the religious minister of yesteryear.

PROJECT INNOVATION, with the author as principal investigator, seeks to develop a new and different concept for the Psychological Laboratory, and which more nearly parallels the use of the Medical Laboratory. Here the utilization of biofeedback equipment promises to offer the clinician in a helping relationship role scientific data analogous with tissue assessment, or the blood, urine, and feces analysis by the pathologist in the Medical Laboratory. Computer based 'gaming and simulation' similarly provides people interaction relata analogous with the jumping on one foot to assess heart functioning by the physician. The use of computer based psychological relata affords the individual immediate knowledge relative to improved understanding of both self and others. The utilization of video tape allows before and after comparative data reflecting emerging change, and the nature and degree of such change. In this concept of a Psychological Laboratory there is not the slightest reference to animal behaviour, and, indeed, meaningful data must of necessity reflect squarely on each unique individual involved in the helping relationship.

Holistic Health

The present concept of a Psychological Laboratory embrace completely the theory pertinent to 'Holistic' medicine. For to understand the dynamics of human behaviour, both the situation and the individual must be reckoned with. Indeed, an individual can not be conceived apart from both the group of which he is a member and the present situation and global environment of that group. Much more important man must be conceived in relation to the four part functioning unit that constitutes the whole, all parts of which are equally important for the inciting and direction of human behaviour: (1) physical, (2) emotional, (3) intellectual, and last but not least (4) spiritual.

Here emphasis is placed on reversing the traditional roles of the client and clinician where the patient (client) remains an active and committed partner in the helping relationship process; as opposed to where client or patient is passive and recipient. Focus is placed squarely on the use of internal healing as a useful and often necessary supplement to surgery, radiation, and drug or chemical therapy. The process is person oriented; as opposed to the typical disease or problem oriented approach.

The objective of this 'Holistic' approach remains the full, vibrant health (positive wellness) of the individual; not simply symptom amelioration as has been too often the case in our past. The most rigid standards must be imposed in terms of defining objective data, and the control of extraneous variables for the diagnostic process and establishing firm cause-and-effect relata. Correlates alone are not presumed to be sufficiently valid to establish a causative basis.

Transpersonal Psychology

Probably the most critical factor involved in health delivery services in the mental health arena has to do with human values. Repeated objective evidence depicts no more than maybe a dozen and half principal values we characteristically embraced by individuals. In terms of priorities for such values, human freedom is most often held as paramount. The most critical requisite for human freedom, almost without exception is the 'self-control' of person involved. A second equally important requisite is often self-understanding. Both of these phenomena must be dealt with squarely by the Psychological Laboratory, if it is to function efficiently. It should be noted that the principal objective for use of biofeedback equipment has to do with fostering self-control of individuals at higher and higher levels of functioning. Gaming and simulation, on the other hand, serves equally well to foster self-understanding. Thus, biofeedback equipment and gaming and simulation are to be some of the cornerstones for the Psychological

Laboratory.

Biofeedback Instruments

The entire repertoire of biofeedback equipment must be available in the present concept of the Psychological Laboratory. Typically, the monitoring of the use of such equipment will remain the province of specifically trained and appropriately oriented paraprofessionalsas regular employees. Always such paraprofessionals function under the watchful eye of mature licensed professionals. More often than not such application may involve a gradual transition from a chemical or drug therapy routine working in close association with a physician to a state where the biofeedback equipment is substituted for the use of such drugs.

Self-Help Concept

Here the client (patient) is instructed on the proper use of the biofeedback equipment. Personal demonstration is provided for the testing of batteries, use of sensitivity levels, volume control, approximate time and number of exposures, and outcome expectations. Such individuals are then personally monitored through one or more sessions in the actual use of the specific equipment involved; to insure that they know how to operate it on their own.

Adapting Feedback Stimulus

Typically, a wide variety of feedback stimuli are available for each of the different feedback instruments; both visual and auditory in nature. For example, there may be different types of auditory 'beeps' -those with a high shrill sound, those with a slow rat-tat-ta. There may be a lecture associated with the variety of sounds. There may be a selection of a variety of musical backgrounds added. The soundsor music may be fixed to activate either when the client is successful or unsuccessful in accomplishing the instrument objective. In addition to the vast choices of auditory tones, there may be visual accompaniments - such as lights, or meter readings. Careful attention will be given in orienting the user to the available feedback stimuli and on how to secure their own particular choice.

Equipment Rental Service

The Psychological Laboratory is prepared to offer users with either rental equipment of a high quality functioning on a daily or weekly basis; or to offer out-patient use of such equipment in specially designed rooms for such purposes. Such equipment may often be acquired on a rent/purchase basis; so that users who desire to have the equipment on a permanent basis may do so.

Personal Sensitivity and Self-Control

The ultimate goal in the utilization of biofeedback equipment is to develop in the individual higher and higher levels of sensitivity in relation to 'SELF-CONTROL' of the functions involved for the respective equipment. Typically such development might be expected to take a period of from 4 to 10 weeks per instrument. Always, such training should be planned for periodic professional immediate supervision, and where attention is given to client articulation of conscious phenomena associated with the developing sensitivity for self-control involved. In addition there should be personal demonstration of such control; so as to eliminate any undesirable features observed; for often faulty training is worse than no training in accomplishing the objective desired.

Biofeedback Goals

The focus in 'Holistic' medicine and in such health delivery service is primary prevention as opposed to crisis intervention; so we would expect a lot of clients who are not really patients in the usual sense of the word to participate. Indeed, it is more than preventive, as it goes beyond the avoidance of illness and accidents in pursuit of continuous and progessively improved self-control and self-actualization of the individual. It must be recognized that even in Holistic medicine there will be 'crisis intervention' incidents, and in such persons the specific crisis must be addressed sooner or later in connection with individually planned therapy.

Usual Laboratory Involvements

In the Psychological Laboratory there are usual routines which are designed toward the progressive increase of self-control of the participant. This, to be sure, is totally analogous to the procedures in the Medical Laboratory. In the latter blood, urine, and sometimes feces analysis are basic functions. So for all patients we would expect to deal with personal relaxation; then with some degree of self-control over the blood-pressure-heart beat and smooth muscles; then to order for self a feeling of 'well being' and finally to establish some degree of self-control over own emotions. Typically, there might be an assessment to what degree such competencies in relation to self-control are present, and an evaluation of needs present. What specific instruments are utilized for each objective may be prescribed by the clinician doing the assessment; otherwise a usual routine of progression of instruments would be followed: (1) Electromyograph (EMG), (2) Temperature Trainer (TT), (3) Electroencephalograph (EEG), (4) Galvanic Skin Response and Pulse MOnitor (BSR and PM).

Crisis Intervention

Where the immediate goal is more intimately related to a specific client described crisis, the Professional Clinician may prescribe specific procedures, or elect to prescribe the usual laboratory routine. Typical kinds of crises for which such amelioration has been effective are as follows: head aches of specific types, certain kinds of petit mal attacks, sexual impotence of certain types, frigidity in sexual response, constipation, tension and restlessness, certain types of mental depression, drug abuse, and most other kinds of mental health problems.

Control of Striaded Muscles

Typically the instrument utilized for this objective would be the electromyograph (EMG). Often opposing muscles function at high levels of energy, and sometimes unknowingly to person involved. The EMG can sense the degree of energy expanded through the tenseness of such striaded muscles and alert the subject to its presence. Where such tension is present, conscious or unconscious, asleep or awake, subjectscharacteristically are tired and restless because they lack energy for other functions. Where sleep inducing drugs (barbiturates, etc.) have been the rule, there must be a gradual transition to the self-control working in close relations with the physician involved. Where opposing muscle groups are not fighting each other, subject attain more rest in a shorter period of time, and become a more effectively functioning individual.

Control of Smooth Muscles

A major function of the smooth muscles has to do with the cardiovascular system in man, and the blood pressure and heart beat remain indexes of such control. Typically, the Temperature Trainer (TT) and the Pulse Monitor (PM) are the instruments involved in this function. There is convincing evidence that a necessary requisite for control of smooth muscles is the presence of a dominant 'parasympathetic' central nervous system control pattern. One means for establishing such a dominant pattern is believed to be the establishment of temperature differentials within different parts of body-two hands, or maybe forhead and hand, etc. Always, however, the problem remains the development within the individual some personal and often unique means for self-control, and a personal awareness of what such control encompasses; as well as how to implement the control. Since the parasympathetic dominance is often referred to as the 'accelerator' of life (as opposed to the 'sympathetic' pattern which is referred to as the brakes of life), this process serves to foster personal motivation, and a will to do. Thus, we have the substitute for the usual dispensing of amphetamine drugs. Where such prescription has been the rule, there should be a gradual transition working in close proximity with the physician to where the TT becomes a substitute.

Feeling of 'Well-Being'

Surely this is the goal of all mental healthy individuals, and must remain the goal in the total helping relationship arena. The instrument usually associated with this objective is the Electroencaphalograph (EEG). Even though this instrument is old since the 1930 when Gibs and Gibs first established the now famous 'spike and dome' pattern of the petit and grand mal activists, but the specific use is not well defined in this area. When self-control of brain wave electricity frequency is attained, there remains the problem of determining at what level of frequency optimum feelings of well being obtain for the specific individual involved: (1) Delta-0 to 3 Hz, (2) Theta-4 to 7Hz, (3) Alpha-8 to 11 Hz, (4) Sigma-12 to 15 Hz, and (5)Beta-15 + Hz. In a very real way, then, the use of the EEG becomes the substitute for the use of opiate and cocaine type drugs to accomplish this objective. This, to be sure, would include the alcoholic drugs, and may include coffee, tea, sugar, and even other highly seasoned foods-when full truth is known.

Control of Emotions and Affect

Here a whole host of equipment is typically involved,and which usually includes: (1) Galvanic Skin Response-where the Basic Skin Response to electricity is stimulus (BSR), (2) Pulse Monitor (PM), and (3) Kodak Slide Projector. The use of these instruments in concert is the subject of another paper.

Fostering Self Understanding

All of these programs are computer based, and many of them have a double functioning nature: (1) one where assessment is made of one's present functioning level, and (2) another where the goal is growth and development through a learning exercise. In the latter the participant may ask computer for HAZARDS-usual risks for a particular choice, and where orientation is in future; CONSEQUENCES-where participant may ask computer for likely consequence or consequences for a given choice, and where orientation is on past-after act; or TEACH-where computer depicts preferred choice, and based on a defined norm group. Most of these individual programs have been written-up as separate reports:

(1) Leadership Decision Pattern (BASPAT)-- where computer depicts pattern one uses in playing role as leader for decision making.

(2) Locus of Control (PRSUAD) -- where computer depicts one's orientation for

electing choices in decision making.
(3) Decision Making Competency (DEDEV)- where the scientific competency of one is assessed in the systems analysis approach to making decisions.
(4) The Drug I.Q. Test (Sex1)-where one's sophistication in the area of sex is assessed in relation to specific norm groups.
(5) The Drug I.Q. Test (Drug1)-where one's sophistication in relation to abuse of dangerous drugs is being assessed.

Social Interaction Assessment

All of these programs are similarly computer based, many of them have the two functioning areas (assessment, and personal development or learning), and most of them involve some aspect of 'gaming and simulation'; where participant is asked to play an assigned role in the answering of questions. Typically, choices afforded range from social conforming to the opposite position, and through a kind of middle-of-the-road choice. Typically each of the programs has been carefully normed, and where the assessment index is based on degree of agreement with established norms. Some of the principal programs in this group are as follows:

Computer Assist Counseling (CASCON)

This program is designed to both assess and teach the participant in relation to 'meaty' (problems that are extreme in nature, but nevertheless often the case-if it can happen, it is likely to happen) problems, and what society or norm group expects as the ideal response.

Professional Counseling (PROFAL)

A whole series of computer based programs that both assess and teach individuals concerning ideal expectations of respective contemporaries in the norm group. Some of the programs included presently are as follows:

(1) Counselors
(2) Teachers
(3) Nurses
(4) Principals
(5) Entrepreneurs
(6) Racial
(7) Health, and
(8) Marriage

Multiple Role Counseling (TWOROL)

Here individuals are asked to play an assigned role in computer-based simulated social problems, where they face problems involving the 'cutting-edge' of confrontation, and assessment is made in relation to expectations of a specific norm group. The second time through the same problems the role assigned involves manipulation of the 'cutting edge' in relation to others, and a comparative assessment is made against the first role choices.

Career Planning

Here individuals are provided a scientific basis for the narrowing of choices in relation to the widest array of job career fields, and/or upper-division colleges and universities throughout the United States and possessions. There is a two stage process involved in the 'narrowing' of choices, with the first stage concerned with delimiting the unrelated jobs or colleges, and with the second stage dealing with a narrowing based on data related to a specific job or college. The latter being a much finer basis for discerning:

(1) Computerized Vocational Guidance System (VOCGUYD), and
(2) Computerized Educational Guidance System (EDGUYD).

Computer-Based Tutorials

A whole series of computer-based programs are presently available in this area, all of them have important implications for the Psychological Laboratory; for they deal with critical behaviour of individuals in the usual and characteristic life-space of our Western culture:

(1) Case Study Analysis (CASTY)-where carefully selected and well organized case studies have been computerized, and where Psychology Interns are expected to diagnose crisis states, and describe their degree and nature of development.
(2) School Psychology Practicum (PSYPRAC)- where computer-based modules are provided pertinent to the pre-training and education of the psychologist.
(3) CAI for BASIC Language (BASIC)-where individuals are introduced to BASIC computer programming.
(4) Elementary Tutorial (TUTOR) - where elementary and secondary students are offered a wide range of CAI and other teaching or self learning units.

Research Support

Here we have an entire repertoire of statistical analysis programs all adapted for use on the microcomputer without a disk system. These program are designed to support a major research effort of small number, and to afford the widest array of choice for statistical procedures.

References

1. Boring, E.G. A History of Experimental Psychology. New York: Appleton-Century-Crofts, 1950.
2. Brown, B.B. Biofeedback-New Mind, New Body. New York: Bantam Books, 1974.
3. Emmons, M.L. The Inner Source. San Louis Obispo, California: Impact Publishers, 1978.
4. Fuller, G.D. Biofeedback:Methods and Procedures in Clinical Practice. San Francisco,California: Biofeedback Press, 1977.
5. _____ Current status of biofeedback in clinical practice. American Psychologist, 1978,33(1),39-48.
6. Hilgard, E.R. Divided Consciousness. New York: John Wiley and Sons, 1977.
7. Cassel, R.N. Computer Assist C ounseling for development of cultural compatible ego-ideal imagery. Psychology in the Schools, 1969,VI(3),289-291.
8. _____ Computer Assist Counseling (CASCON)-A basic essential in youth correction programs. Journal of Correctional Psychology, 1969,XXI(2),21-23.
9. _____ Helping relationships and decision making competency. World Journal of Psychosynthesis,1972,4(7),26-30.
10. _____ UNM Computerized School Psychology Program Supplement (PSYPRAC). World Journal of Psychosynthesis,1972,4(10),26-30.
11. _____ The UNM Computerized Decision Development Competency System (DEDEV). Psychology, 1972,9(3),40-45.
12. _____ Psychological aspects of gaming in relation to child play. College Student Journal, 1973,7(1),17-20.
13. _____ Decision counseling. Education,1973,94(1),38-43.
14. _____ Fundamentals of humanistic psychology. World Journal of Psychosynthesis.1973, 5(4),21-24.
15. _____ The UNM Case Study Analysis (CASTY). Psychology, 1973,10(2),6-14
16. _____ Psychological aspects of human freedom. Psychology, 1973,10(4),37-39.
17. _____ Decision dynamics in relation to confrontation (TWORCLE). World Journal of Psychosynthesis,1973,5(1),26-27.
18. _____ Basic fundamentals of mind control and Transcendental Meditation. Psychology, 1974,11(2),26-33.
19. _____ The college penal experience alternative (COPEX). World Journal of Psychosynthesis, 1974,6(1),30-36.
20. _____ Instructional gaming and simulation. Contemporary Education, 1974,XLV(2),100-105.
21. _____ Computer Assist Counseling (CASCON).Psychology, 1975,12(1),3-9.
22. _____ The Computerized Educational Guidance System (EDGUYD). Psychology, 1975,12(4),12-17.
23. _____ Describing the general perimeters for eight different levels of reality in Transpersonal Psychology. World Journal of Psychosynthesis,1976,8(2),23-26.
24. _____ The marketable skills inventory. Psychology,1976, 13(2),29-35.

25. Cassel, R.N. Fostering Transcendental Meditation using biofeedback eliminates the hoax and restores creditability to art. Psychology, 1976,13(2),58-64.
26. _____ A computerized pastoral counseling system. Psychology, 1977,14(2),20-23.
27. _____ and Blum,L.P. Computer Assist Counseling (CASCON) for prevention of delinquent behavior among teenagers and youth. Sociology and Social Research, 1969,54(1),72-79.
28. _____ and Mehail,T. The Milwaukee Computerized Vocational Guidance System (VCCGYD). Vocational Guidance Quarterly,1973, 21(3),206-213.
29. _____ and Stroman, S.D. Evaluation of the Computerized Decision Development System (DEDEV) for use with ROTC students. Journal of Instructional Psychology, 1974,1(1),12-22.
30. Cassel, R.N., and Zander,G. Teach me what I want to know about drugs for junior high school students. International Journal of Addiction, 1973,9(5).
31. Karlins, M.,and Andrews, L.M. Biofeedback:Turning on the Power of Your Mind. New York: Warner Books, 1975.
32. Miller, Neal E. Fact and Fancy About Biofeedback and Its Clinical Application. Washington, D.C.: American Psychological Association, 1976.
33. Minor, E. Gateway to the Unknown. Los Angeles, California: Crescent Publications, 1975.
34. Moberg, D. Medical applications of microcomputers. Current technology with a look at the future. Interface Age,1978,3(7), 76-79.
35. Moskowitz, M.J. Hugo Munsterberg: A study in the history of applied psychology. American Psychologist,1977,32(10),824-42.
36. Murphy, G., and Kovach, J. Historical Introduction to Modern Psychology. New York: Harcourt Brace Jovanovich, 1972.
37. Ruphuy, R.S. Psychology and medicine: a new approach for community health development. American Psychologist,1977,32(11), 910-13.
38. Schwartz, J. Voluntary Controls. New York: E.P.Dulton,1978.
39. White,J.,and Fadiman,J. Relax, How You Can Feel Better, Reduce Stress, and Overcome Tension. New York: The Confucian Press,1976.

MICROCOMPUTER APPLICATIONS FOR BIOMEDICAL INSTRUMENTATION: A Monitor for the Corning M-175 Blood Gas Analyzer

Robert C.A. Goff, M.D. Research Associate
Children's Hospital Medical Center of Northern California
51st and Grove, Oakland, California 94609 (415) 654-5600 ext 276

President, Berkeley Medical Data Associates, Inc.
Box 5279 Berkeley, California 94705 (415) 653-5707

Abstract

New biomedical instruments provide digital output of test, calibration, and machine status data, usually in the form of seven segment LED displays. Many such machines are microprocessor controlled. This paper discusses in general those which the manufacturers can equip with a serial data port for transmitting results and/or receiving control commands. A microcomputer monitor, developed by the author for the Corning M-175 Blood Gas Analyzer, will be used as an example of the considerations which should be addressed when designing software for such a biomedical application. [Key words: **computer monitor, blood gas, biomedical instrumentation, laboratory data storage**]

Introduction

In the practice of medicine, we are just now seeing the firstfruits of microprocessor controlled biomedical instruments and life support devices. Of importance to this discussion is the fact that such devices are capable of providing their data to an external device such as a microcomputer system. While at present most microprocessor controlled devices do not come equipped with any form of external interface, most of them can be modified, either by the manufacturer or by the user, to support a serial or parallel data interface. The essential requirement is that the data displayed by the device be in a digital form, which would allow a supplementary interfacing device to easily decode and transmit the data externally.

If the digital form of the data is available only as a seven segment LED signal, then the interface must tri-state, buffer, and decode the seven segment signals in such a way that each of the data types provided by the instrument may be transmitted in an identifiable format to the interface. While such a hardware task is conceptually simple, it may require significant modifications to the original electronics of the instrument, and such modification will certainly complicate the picture of warranty and service contracts relating to the device. In considering the construction of a "homemade" interface, the biomedical electronics technician (BMET) must first acquire a thorough knowledge of the equipment which is to be modified, and such knowledge is usually available only in the form of a training course provided by the manufacturer or vendor of the device. If the warranty period has already expired for the device to be modified, and the user institution has decided to maintain the device with the service of its own BMETs, then the "homemade" approach may be reasonable. Otherwise, I would strongly recommend that only manufacturer provided interface electronics be installed.

An optimal interfacing device should posess the following capabilities:

1. read the various data types and store them in interface RAM prior to attempting to communicate the data externally;
2. provide a parity bit with each character to be transmitted;
3. provide a longitudinal parity check character with each data string to be transmitted;
4. provide the option of either spontaneous data transmission (to a dumb device or an interrupt driven smart device) or an inquiry/acknowledge protocol (for transmission to a smart device which is not interrupt driven and cannot be dedicated solely to monitoring the data output);

5. if an ACK/NAK protocol is to be used, to retransmit the data in response to the NAK message;
6. provide BAUD rate options for interfacing to the most common external devices;
7. allow independent operation of the device in the event of computer system failure;
8. if more than one identical device is to be operated simultaneously, to provide a reliable means of determining whose data is on which of the devices (some sort of software ID);
9. transmit all data in a commonly available code and in an unambiguous format;
10. posess in ROM the instructions which allow the above capabilities.

The software needed to relate to such an interfacing device will varry to some extent with the equipment being interfaced, however there are a number of considerations which should be evaluated regardless of the specific devices. Most important is the fact that most clinicians and researchers have spent years gleaning the data they require from inadequate or inconvenient sources (such as piles of lab slips or disorganized laboratory printouts) and, as a result, cannot provide the programmer with a complete list of desired output unless they are first provided with an inspired discussion of the many exciting and valuable possibilities. Until these ultimate users of the data have gained an understanding of the possibilities and systhesized a list of their desired output, the programmer must limit his program development to the actual drivers of the I/O hardware. Even the selection of a mass storage device and data file format must await their specifications. In the same line of thought, the final program structure must be capable of manipulating and storing considerably more data types than those originally anticipated, since the expectations of both clinicians and researchers will certainly increase dramatically once they enjoy the benefits of automated data processing. This will be discussed further below.

Utilizing a manufacturer provided interface will circumvent much of the headache of hardware design and, at the same time, not jeopardize the maintenence and warranty provisions so essential in operating laboratory equipment or other biomedical electronic devices. It must be understood, however, that the manufacturer must provide the programmer with absolutely thorough documentation of both the operation of the interface and the exact format in which the data will be supplied. The latter should include:

1. the type of character code to be used;
2. the type of error checking to be implemented;
3. the method of identifying each data type to be provided;
4. the exact communication protocol(s) necessary;
5. the size and character range of each data field;
6. the hardware communication options available (such as BAUD rate, full/half duplex, number of start and stop bits, etc.);
7. the outcome of data transmission errors.

With such complete documentation and with a list of desired final outputs (from the clinicians or researchers) the programmer should be able to construct the required software with relative ease. Probably the most difficult aspect of designing the software is with respect to making it "crash proof" [knock on wood]. It must be implemented in such a way as to allow the essential functions of the device to continue, unimpeded, in the event of computer or software failure (this is mandatory in any health care application). For example, if the program must access floppy disk files, then it is subject to crashing if the disk drive latch is inadvertantly left open. Ordinarily in such an event a "HARD DISK ERROR" message is displayed on the CRT and the program halts. A far more preferable outcome would be the "trapping" of such an error by the program, the display of an appropriate error message, followed by the continuation of the program in a mode which bypasses disk accesses (or the offending disk drive). Programming of this sort requires more thought, sophistication, and patience than most of us are accustomed to devoting to the average programming task.

The remainder of this discussion will concentrate on a specific example of the software needed to utilize a manufacturer provided interface. The Corning M-175 Blood Gas Analyzer was chosen since it is a common laboratory device, and provides a wealth of data and communication options. The majority of the blood gas analyzers currently in use in this country are manufactured by Corning, and are now able to support their well designed

printer/computer interface option. The software was designed by the author for use in the Department of Pulmonary Medicine at Children's Hospital in Oakland. Since the greatest demand on these machines in our hospital is made by the Intensive Care Nursery (the largest in the San Francisco Bay area) special programming consideration was given to the clinical needs of that service.

The Corning M-175 Blood Gas Analyzer

This is a microprocessor driven laboratory instrument which analyzes blood samples for their acidity and content of oxygen and carbon-dioxide. (These are essential in the management of ventilator patients.) In addition to these measurements, made by precision electrodes, the M-175 calculates a number of other blood parameters, such as the saturation of hemoglobin, the content of bicarbonate ions, and the standardized base excess. The machine is capable of automatically calibrating itself at specified intervals and displaying pre- and post-calibration values. It is also able to diagnose and display any of 31 specific malfunction error messages. The standard machine utilizes LED displays of all data values, as well as a small self-contained printer for printing the values onto lab slips. In most applications the printed lab slips are the only permanent copy of the data. At Children's Hospital, our two on-line machines process a total of approximately 250 of these tests each day, seven days a week, representing a tremendous data load.

Corning has now made available a dual interface which allows communication of the M-175 Blood Gas Analyzer with a printer and/or a computer. Its cost is approximately $1,800. It allows transmission of all of the above data, as well as machine status information (with any error conditions), and provides substantial error checking and complex software handshaking options -- all switch selectable. In addition, the computer may command the M-175 to perform either a 1-point or 2-point calibration at any time. The interface will drive either an Rs232C interface or 20ma current loop interface, and has a complete array of switch selectable hardware handshaking options. Their user documentation is among the most complete in my experience, though it regretably does not include hardware documentation suitable for maintenance or modification of the interface electronics. Nonetheless, to a user and programmer, the documentation is comprehensive. A sample from their 31 page interface manual will be found in the appendix to this article.

The Corning interface simply requires that the computer software be capable of interpreting ASCII code, and be able to accept a data transmission within 15 seconds of an ENQuiry message. The complexity of the requisite software rests not on its communication modules, but rather on its methods of storing and manipulating the data, providing several types of printed reports, and allowing maximum standby function in the event of unrecoverable errors. The software may utilize the character parity and longitudinal parity checks to request a retransmission of data, or it may simply ignore the error checks and ACKnowledge all data transmissions. For further hard copy reliability, the M-175 printer interface may simultaneously be enabled to print all output regardless of the status of the computer. Using multiple computer I/O ports allows several M-175 machines to be monitored simultaneously.

Designing The Monitor

The design of the M-175 computer monitor was prompted by the requirement of the State of California that all laboratory test data, along with pertinent clinical and administrative data be continuously logged as a permanent record. In trials of logging the required information by hand it was determined that the necessary logging time would require between $25,000 and $40,000 per year in staff time. The required information numbered 21 separate data for each test performed, plus the frequent logging of all calibration data. With these estimates, it became clear that the cost of performing the bureaucratic paperwork by hand far exceeded the cost of implementing a microcomputer monitor.

The output required by the state represented a motivation for providing other output of greater clinical relevance. The following output is the result of merging the necessities with our imaginations:

1. a continuous log of all test results, accompanied by other data required by the state;
2. a periodic patient-oriented printout of all billing information relating to the blood gas tests;
3. a daily patient-oriented printout of all tests performed;
4. a daily chronologic printout of all quality control and

calibration data;
5. an on-demand CRT display of any stored data indexed by any number of parameters.

The data input to the computer needed to be limited as much as possible to non-keyboard input. To facilitate this goal, any recurring patient information which was required for the state's log (such as attending physician, hospital ward, patient file number, etc.) would be entered only once -- usually at the time of admission or at the time of the patient's first blood gas. This could then be recalled from storage for each consecutive test entry. The time and date of each test is obtained from the onboard clock-calendar. The sequential test number is generated by the computer. While all ventilator settings must, at present, be entered at the keyboard, this data will soon be available to the computer directly from the patient's ventilator. (The new Seacrist infant ventilator is an 8080A driven device which can be equiped with an Rs232C serial interface, which will provide all of the ventilator settings to the computer on demand.) In order to minimize the keyboard input, the patient's previous ventilator settings are displayed, and if there is no change, only one keystroke is needed. Since, for each test entry, the name of the patient must be indicated, the computer maintains a table of all active patients, and displays a numbered menu in preparation for each new test.

Software handshaking was implemented for several purposes. First of all, with two machines operating simultaneously, it is necessary to ACKnowledge only one at a time. Secondly, since the pericdic printed summaries and reports require a significant time to print, the software had to poll each input device between each output line, so as not to miss a data transmission (the M-175 waits 15 seconds for and ENQuiry ACKnowledge), and so that new tests may be entered from the keyboard. This second problem results from the fact that the blood gas machines operate at the same demand 24 hours a day, in contrast to the typical medical accounting computer which can perform its lengthy printouts at the day's end. A third purpose for the software handshaking was to avoid the interruption of disk accesses in the absence of a hardware interrupt controller. (Implementing hardware interrupts within the confines of a single BASIC program seemed to be not worth the considerable effort that the programming would require.)

Data packing is a substantial problem in microcomputer applications which carry such a high data load. In this application, long term data storage was implemented for research purposes yet to be determined. It was assumed that since all the data printed on the log must pass through the computer, it would be a crime not to save it in a machine readable form for later reference or applications. The one concrete application which will utilize all of this stored data is in the Intensive Care Nursery Bedside Patient Monitor which is currently under development by the author at Children's Hospital in Oakland. The bedside computer will from time to time request the blood gas data from the blood gas computer, and then process and store that data on the individual patient's floppy disk. It is certain, however, that with large volumes of correlated blood gas data, patient ages, and ventilator settings, all in machine readable form, significant new knowledge will result concerning pulmonary management.

To accomodate the large volumes of data on the smallest possible number of floppy disks, all numeric data is stored as ASCII strings. An economy results by bypassing the numeric precision of the BASIC used in writing the program. It should be realized that most clinically significant laboratory values fall in the range between 0.01 and 5000, and can be ASCII encoded in one to three bytes. Eight digit precision BCD requires 5 bytes for any number; fourteen digit precision requires 8 bytes. Much of the descriptive data can be encoded as one byte per datum (one byte allows 256 different possibilities). The price to pay here is that of processing speed, which is significantly decreased if the data must always be decoded prior to manipulation, but for this application, the processor will be idle most of the time anyway.

Data access was a major consideration in designing the data file structure. There are several types of data access which are required to provide the desired output. The menu of patients displayed prior to initiating a new test is composed by reading the disk files and accumulating all of the current patients. Since this is a frequent task, and one for which the user impatiently waits prior to each test, it must be very fast. The patient directory is an alphebetized semi-dense list, and can be read and displayed as the closing process of each computer operation. On the other

hand, detailed clinical data must be accessed only upon selecting a particular patient. The directory, since it is ordered and of known length, can be rapidly searched using a modified binary search. The correct entry then contains a link to the detailed clinical information. Since the detailed information is not included in the directory entry itself, the task of keeping the directory ordered and semi-dense during the addition or deletion of patients is a fairly rapid process.

The actual test data, since it is written chronologically rather than by patient, is contained in an unordered file area in which each patient's data is doubly linked. The top and bottom address of each patient's linked list of data entries is contained in the first entry, which is in fact the clinical information entry. Because the usual need to access test data is chronological (either forward or backward), these linked lists provide rapid access to any patient's test data. In addition, the first entry also contains a link to the last test which was billed, and a link to the last test that was included in a summary printout. Thus, for the billing access, each patient's record is searched to find the last test billed, then all subsequent tests in that patient's linked list are billed and flagged. The slowest access occurs when the user requests any data for which there is no dedicated link (for example, a request for all tests on all patients occurring between midnight and eight a.m. for the last two weeks). In such cases, virtually every file entry must be searched and compared to the specified criteria. If, instead, such a data request includes a qualifier for which there is a dedicated link, then the search is dramatically faster.

With the above shortcomings in mind, it becomes obvious why it is a good idea to allow more room for each entry than is actually needed. Initially, 50% of each record was empty. By the time the major program testing was complete only 30% of each record was empty. The increase in record length was directly a result of increasing the number of dedicated links so as to make some newly desired output practical in terms of the file access time.

Permanent storage is a difficult problem. Once the decision has been made to store the data permanently, the task of selecting the medium becomes a major one. In a microcomputer application which requires continuous operation, the only candidates are floppy disks and hard disks. Since our application did not have a cost effective reason for permanent storage, the current price of hard disks eliminated them from consideration. Left with the floppy disk, it was decided that the 5 1/4 inch disk drive was the most desirable in terms of mechanical reliability in a continuous operation application, since the drive motors shut down automaticaly if no disk access is made for several seconds. Ideally, the size limitation is not an issue with the new double sided, dual density drives, but at the time of this writing they are in production only by one manufacturer. With our data load, the single sided, single density drives, when used in a triple drive system, will allow the accumulation of approximately two weeks of data before requiring that the files be thinned. This turns out to be fairly reasonable, since our protocol is to duplicate each disk at midnight each night, and to empty the on line data files once each week. Although our blood gas machines will generate approximately 75 disks of data each year, given this format, we plan to cut this to one quarter that number (about 2 disks per month) when the high density drives are readily available.

Multi-megabyte fixed disks will surely play a significant role in such permanent storage applications, once they are available at a price comparable to floppies. A 27 megabyte disk, for example, would store four years of blood gas data, or in the case of the computerized patient records which we are developing, the complete records of approximately 600 intensive care infants (or ten times that number of typical medical office patient charts). The drawback is that the entire disk must be duplicated for back-up purposes.

The choice of a language was somewhat difficult, that is if we don't group all BASICs as a single language. Since the application does not require high speed real-time processing, there was no need for assembly language. BASIC was an obvious choice because of the extensive use of ASCII string encoding. The program was developed using North Star BASIC (version 6, release 4) rather than CBASIC primarilly because CP/M tends to crash if the disks are shuffled from one drive to another, whereas North Star's DOS doesn't care. In health care applicatons, maximum attention must be paid to avoiding

system crashes. Also, CP/M disk accessing tends to be slow, even though the remainder of the program's execution is faster in the compiled form. Since the slowest portions of the program involve disk access, the difference was felt to be significant. Perhaps as languages such as PASCAL and MUMPS become available for microcomputer applications, they will supercede BASIC.

Selection of the hardware was mostly dictated by the above considerations, plus the constraint of counter space in the blood gas lab. A Z-80 processor was chosen solely for its speed. Since the disk drives would need to be located a some distance from the CRT and keyboard, a CRT peripheral terminal was placed near the M-175 machines, and the remainder of the hardware placed in an adjacent room. The two blood gas machines required two additional serial ports, and a fourth is in anticipation of interfacing a software selectable switching device for direct input from the Seacrist ventilators. If it is expected that the computer will need to communicate with other computers, then an additional serial port will be needed. The printer can be any 128 column printer. Dot matrix is acceptable. North Star Microdisk drives (3) were chosen to be compatible with the North Star BASIC.

Completing The Circle

The day is not far off when systems such as the one described here will, in addition to accumulating and logging data, generate decisions for the control of ventilators. Since the Seacrist ventilator includes two needle valves among its controls, it cannot be used, as is, in such a feedback loop. It can only provide data. But the next generation of microprocessor controlled life support devices will certainly posess the capability for accepting digital control commands. With sufficient safeguards, such a control feedback loop could markedly reduce the burden to the physician of minor routine ventilator adjustments, and in a large intensive care nursery such as that at Children's Hospital, free the physicians to make only those decisions which fall outside the predefined realm of the computer. For the majority of situations involving ventilators, a simple algorithm could reach the proper decision for generating setting changes, and could flag the physician for any unusual situations. The challenge of developing such software lies not in constructing the command algorithm itself, but rather in selecting the limits beyond which the computer's decisional authority should not be permitted.

Summary

A general discussion of interfacing biomedical electronic equipment to computers has been presented using the Corning M-175 Blood Gas Analyzer and a computer monitor developed by the author.

Please Note:
The M-175 Monitor was developed at Children's Hospital in Oakland, California, and is available from Berkeley Medical Data Associates, Box 5279, Berkeley, California 94705. The complete listing in North Star BASIC, can be obtained by institutions for $100 and the entire program on a 5 1/4 inch 10 sector disk with fully documented listing and user's manual for $700. Individuals who are interested in the dynamics of certain portions of the program are invited to write to the author for a partial documented listing at no charge.

Acknowledgements

The author wishes to thank Dr. John McQuitty and Dr. Barry Phillips of Children's Hospital for their assistance in the development of this program. The author also wishes to acknowledge the support of Children's Hospital Medical Center of Northern California in the continuing development of new microcomputer applications for health care. Special thanks is offered to Mr. Loren Lewis. cheif biomedical electronics technicia at Children's Hospital, without whose continued and painstaking assistance this project would not have been possible.

About The Author

The author is a graduate of the University of Missouri -- St. Louis, and of the University of Missouri School of Medicine -- Columbia. He has undertaken specialty training in Pediatrics at the University of Missouri Medical Center -- Columbia, Mt. Zion Hospital -- San Francisco, and Children's Hospital -- Oakland. He was a Fellow in Neonatology at Children's Hospital -- Oakland, and continues there as a Research Associate and practicing pediatrician. He is President of Berkeley Medical Data Associates, Inc., a consulting firm specializing in microcomputer applications for health care.

4.2.4 Baud Rate

The M175 allows the terminal/M175 interface to operate at any of the baud rates listed below. These are switch selectable inside the M175, and are set during installation.

50	134.5	600
75	150	900
110	300	1200

4.3 Data Formats

4.3.1 Data Transmitted by the M175 to the Computer

Data transmitted by the M175 to an external computer will be on a block basis. Each block will be organized as follows:
- The first character in each block will be an STX
- The 2nd, 3rd, and 4th characters will define a message character count (n)[1]
- An n-character message will follow
- An ETX will follow the message
- A check character (longitudinal parity - even) will follow the ETX.[2]
- The last character will be an EOT

There are 7 message types which the M175 can transmit to the computer. These are as follows:
- 1-pt. calibration data
- 2-pt. calibration data
- sample data
- status data
- patient ID cue message
- ACK/NAK message
- Enquiry message

The character string transmitted to the computer for each of the above message types are shown in Figures 4-4 through 4-10 respectively.

---

1. This 3 digit character count is a base 10 number, MS digit is the first of the three characters.

2. The check will be for characters starting at STX and ending at ETX. STX and ETX are included in the calculation. This check character is calculated by computing the "exclusive OR function" for the first two characters in the message, exclusive ORing the result with the third character in the message, exclusive ORing the new result with the fourth character in the message and so on until the last character (ETX) is included.

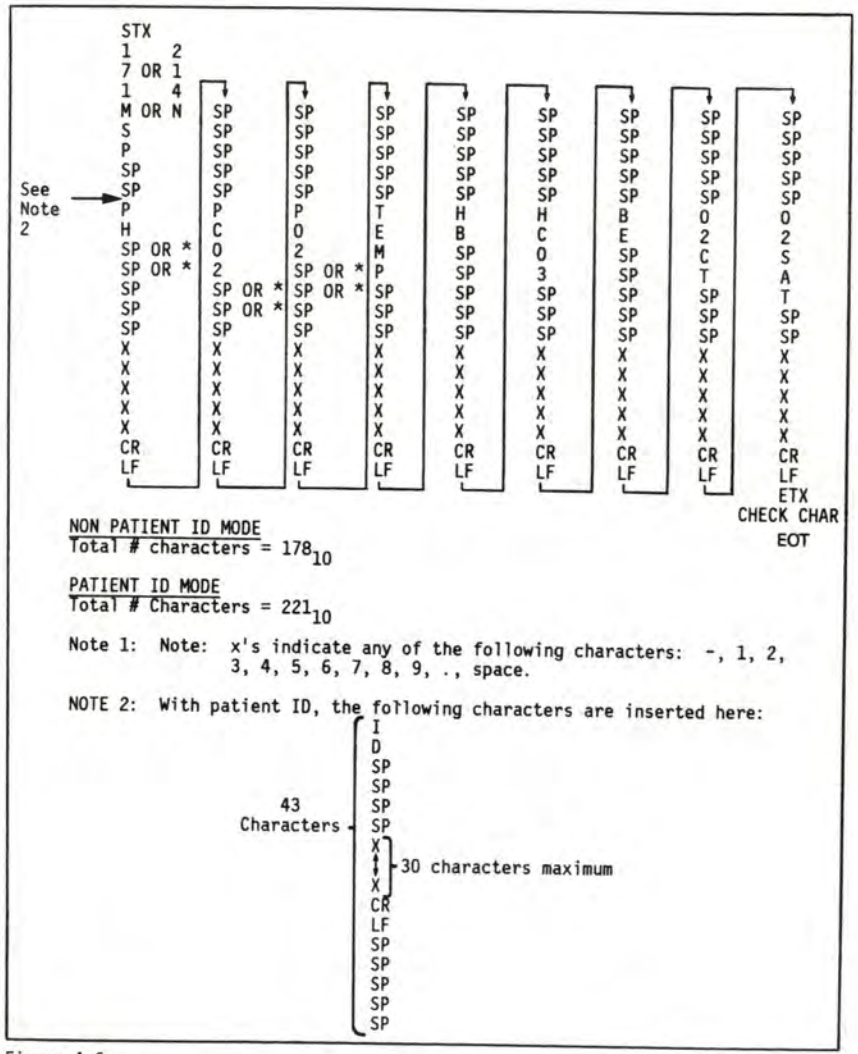


Figure 4-6
Computer Data Transmission for Sample Data

MINNESOTA LOOKS AT MICROCOMPUTERS

Dr. Kenneth E. Brumbaugh
Manager of User Services

Minnesota Educational Computing Consortium
2520 Broadway Drive
St. Paul, Minnesota 55113

INTRODUCTION

The purpose of this paper is to present to microcomputer users from around the country a view of recent developments in Minnesota regarding the potential for microcomputers in educational computing.

BACKGROUND

The utilization of microcomputers is relatively new, and is occurring in a variety of fields throughout the entire country. Interest in microcomputers is high and it is difficult to stay abreast of an exploding knowledge base. In the education area, expectations about microcomputers tend to be ahead of capabilities and the knowledge required to utilize them. A small number of microcomputers have been purchased by schools, colleges, and universities throughout Minnesota and the rate of acquisition is increasing. In fact, an exponential increase in the sale of microcomputers can be expected, particularly in the case of systems costing less than $1000. The inexpensive systems are well suited for introductory programming, and small instructional simulation activities.

Software development and related instructional support for the promotion of classroom computing activities will not keep pace with the initial movement toward microcomputer usage. In the case of many of the systems use will be impeded because of a lack of hardware/software features. In order to meet the expanding need for assistance with microcomputer hardware and software it is necessary for higher educational institutions, and educational service agencies to broaden their support in these areas.

In December, 1977, the Minnesota Educational Computing Consortium (MECC) Board of Directors established the Microcomputer Task Force for the purpose of advising the Executive Director and member systems on matters related to the evaluation and utilization of microcomputers for instructional computing services. The Task Force was composed of individuals with expertise in various areas of instructional computing who met each month to discuss findings of its subcommittees. MECC staff performed the background work and the research necessary to provide information for the full Task Force, whose objectives were:

> To determine the strengths and weaknesses of microcomputer utilization in various instructional computing modes and environments.
>
> To prepare a position statement regarding the potential for large scale acquisition and utilization of microcomputers and the appropriate roles and responsibilities for MECC, member systems, and institutions or school districts.

SUMMARY OF MICROCOMPUTER SOFTWARE

Introduction

The procedure for evaluating microcomputer systems software followed several steps. First, a minimal system configuration was defined. As a result vendors supplying a microcomputer for evaluation knew what the minimum requirements for an educator would be. Table 1 lists the eight required components of a minimal microcomputer system.

Table 1

DEFINITION OF A MINIMAL SYSTEM

1) A microprocessor

2) I/O Device(s). The system must include a keyboard and printer and/or monitor.

3) An ASCII keyboard for input.

4) A permanent file storage device. This can be of any form: disk, hard disk, diskette, etc.

5) A vendor-supplied operating system.

6) The BASIC programming language must be supported

7) At least 12K of user memory must be available. This excludes memory space required for the operating system and the language processor(s).

8) All components, software and hardware must be documented. This must include instructions on the Operating System, a language manual, and setup and maintenance instructions for the system.

The Microcomputer Task Force decided to evaluate only microcomputer systems marketed and supported by Minnesota vendors, who were contacted and asked to either supply demonstration systems for MECC staff to evaluate, or to complete all evaluation steps themselves and submit the results to MECC staff for validation.

The vendors listed in Appendix A (see footnote #1) provided the necessary equipment or information and their products were included in the study. Table 2 lists the microcomputer systems which were analyzed, along with some descriptive information. Several well-know minicomputer and microcomputer manufactures chose not to cooperate and did not supply either hardware for analysis or the necessary information; so their products were not included in this report. In some cases microcomputers which were supplied to MECC were also loaned to area schools and colleges, where the equipment and software were tested in classroom teaching situations. MECC staff comments on the various microcomputers evaluated are summarizied in Appendix B.

Footnote #1: The Apendices referred to in this paper are not included in this article, but are available from the Minnesota Educational Computing Consortium, 2520 Broadway Drive, St. Paul, Mn. 55112 as part of the microcomputer report, which costs $4.00.

Table 2

MICROCOMPUTER SYSTEMS INCLUDED IN MECC SURVEY

CODE/NAME	MEMORY SIZE*	PACKAGE PRICE
A ALTAIR ATTACHE	7	2395.00
B ALTAIR 88-1301	25	3890.00
C ALTAIR MULTI-USER	25	6075.00
D ASTRAL 2000	20	6490.00
E IMSAI VDP-80/1000	16	5770.00
F NCR 7200	4	7000.00
G OLIVETTI P6060	16	7607.00
H PET 2001	7	795.00
I POLYMORPHICS 8813	5	3840.00
J PROCESSOR-TECH SOL	28	3305.00
K RADIO SHACK TRS-80	16	895.00
L APPLE 2 (INTEGER BASIC)	6	1195.00
M TEKTRONIK 4051	8	4800.00
N TERAK RT-11	24	6000.00

* Available for use by user

During the early stages of the study, MECC staff classified educational computing on microcomputers into two areas: applications-oriented and computer science-oriented. The applications programming classification was defined as the use of a microcomputer system to meet the classroom needs for running application programs written in the BASIC language. This classification also included the ability to write simple BASIC programs for problem solving in elementary and secondary schools. This mode of use requires the capability to run programs which are simulations, etc., and are generally more than 8K in core requirements and may go up to 32K. Because these systems will be making heavy use of the library of programs on the timeshare system, downloading capability is highly recommended, if not essential. This system will also be used in a programming mode, probably using BASIC exclusively. In some cases FORTRAN would be helpful. BASIC in ROM is highly desirable. An editor would be desirable to aid in program conversion.

The computer science classification was defined as the use of a microcomputer system to meet the needs of secondary and post secondary computer science instruction involving such topics as advanced programming, operating systems, compilers, and assembly language. This is a system used primarly by those who are interested in studying the computer system itself. Accessibility to assembly language is needed. Ability to modify system software would be desirable. Software flexibility is important. Some high level language should be available but not needed in ROM. The system will also be used in a programming mode. Ability to interface assembly language routines with high level language is important. An editor would be desirable for program conversion and general programming.

A list of fifty features which are useful in educational computing was generated (see Appendix C) and prioritized as essential, desirable, or extra items associated with microcomputer use. Each microcomputer was tested to determine if it had the necessary equipment and/or software capability specified. Table 3 shows the resulting scores obtained by each microcomputer included in the study. Appendix D gives the specific breakdown for each microcomputer evaluated.

Table 3

NUMBER OF SPECIFICATIONS AVAILABLE BY CATEGORY AND INTENDED USE

VS.

MICROCOMPUTER SYSTEM INCLUDED IN MECC SURVEY

INTENDED USE	MICROCOMPUTER CODE REFERENCE													
APPLICATION/PROGRAMMING # OF SPECS. AVAILABLE	A	B	C	D	E	F	G	H	I	J	K	L	M	N
ESSENTIAL (POSSIBLE 29)	28	29	28	29	29	24	29	25	28	26	24	24	28	28
DESIRABLE (POSSIBLE 23)	17	17	14	15	14	14	14	12	16	16	11	8	18	16
EXTRAS (POSSIBLE 8)	3	3	1	5	2	1	2	4	2	1	2	3	4	5
COMPUTER SCIENCE # OF SPECS. AVAILABLE														
ESSENTIAL (POSSIBLE 34)	33	33	29	34	32	28	29	27	31	28	26	26	32	34
DESIRABLE (POSSIBLE 16)	10	12	11	10	11	9	11	9	14	12	8	8	13	11
EXTRAS (POSSIBLE 10)	5	4	3	5	2	2	5	5	1	3	3	1	5	4

BASIC Language Features

Since BASIC is both the most often used language compiler by microcomputers and by MECC timeshare users, specific BASIC features and performance tests were completed. The statement "that you only get what you pay for" does not necessarily hold true in the case of language capabilities on microcomputers. The cost of obtaining microcomputers with similar selected features varies by an order of magnitude of nine. This is documented in Table 4, which shows the BASIC features available by microcomputer, and Table 5, which summarizes results obtained when timing selected BASIC routines. The scripts used to obtain the data in Table 4 and Table 5 are attached to this report as Appendix E.

Table 4

SELECTED BASIC FEATURES SCRIPT TEST RESULTS

SCRIPT NAMES

B2 - Sequential file handling
B3 - Random access file handling
B4 - Chaining
B5 - Time function
B6 - String functions
B8 - Matrix operations
B9 - Formatted output

P = pass
- = fail

	Script						
Feature System	B2	B3	B4	B5	B6	B8	B9
ALTAIR ATTACHE	P	-	P	-	P	-	P
ALTAIR 88-1301	P	P	P	-	P	-	P
ALTAIR MULTIUSER	-	-	-	-	P	-	P
ASTRAL 2000	P	P	P	-	P	-	-
IMSAI VDP-80/1000	P	-	-	-	P	-	P
NCR 7200	P	-	-	P	P	-	-
OLIVETTI P6060	P	P	P	-	P	P	P
PET 2000	-	-	P	P	P	-	-
POLYMORPHICS 8813	P	P	P	P	P	-	P
PROCESSOR TECH	P	P	P	-	P	-	P
RADIO SHACK TRS-80	-	-	-	-	P	-	P
TEAM APPLE 2	-	-	-	-	P	-	-
TEKTRONIK 4051	P	-	P	-	P	-	-
TERAK RT-11	P	P	P	-	P	-	-

Table 5

SELECTED BASIC PERFORMANCE SCRIPT TEST RESULTS

NAME	Time req.* to complete computation script Script B1	Number of math functions available Script B7	Time req.* to gen. and sort numbers Script B10
ALTAIR ATTACHE	15	11	2.0
ALTAIR 88-1301	15	11	2.0
ALTAIR MULTIUSER **	15	11	2.0
ASTRAL 2000	75	11	4.5
IMSAI VDP-80/1000	18	9	1.5
NCR 7200	22	11	2.0
OLIVETTI P6060	16	13	2.5
PET 2001	12	11	1.3
POLYMORPHICS 8813	50	9	2.3
PROCESSOR TECH SOL-20	50	9	2.3
RADIO SHACK TRS-80	18	11	2.0
TEAM APPLE 2	9	11	1.1
TEKTRONIK 4051	26	12	4.0
TERAK RT-11	9	9	2.0

* All timings are in minutes.

** The timings depend on the mix of activities on the various terminals

SUMMARY OF MICROCOMPUTER SYSTEM HARDWARE ANALYSIS

General Information

The microprocessor, a single-chip Large-Scale Integrated (LSI) circuit device is the focus of the microcomputers survey. All make use of a technology called Metal- Oxide-Semiconductor (MOS) wherein the microscopic circuit is processed into the surface of a slice of silicon crystal, which usually is packaged in a 40-pin flat pack. The systems surveyed, with one exception (16-bit), operate with 8-bit bytes and 8-bit memory and generally provide 16 address lines permitting direct addressing of up to 65 thousand (65K) bytes of memory. Usually this means 65K of Read Only Memory (ROM) and Random Access Memory (RAM) combined. The 16-bit unit has greater processing capability at the cost of more expensive memory and other support logic.

All evaluated systems were partially expanded but had additional expansion potential. Most were equipped with one or two mass storage devices, either tape or disk, in order to permit rapid change of application. Disk systems load substantially faster and more reliably, followed in order by tape cartridges and then cassettes. Cassette systems vary in speed from about 120 down to 30 or less characters per second. Disk and cartridge systems provided recovery of files by name or number, whereas most cassette systems did not.

Hardware features for communications between microcomputer and timeshare systems and down-line loading are available on several systems but were demonstrated on only a few. This feature is considered important for educational uses. General hardware information for each microcomputer evaluated is given in Appendix F.

Electrical Considerations

Grounded metallic enclosures provide rigidity and shielding, both of which are requirements for reliability. Grounding via a 3-wire A.C. power cord also protects users from electric shock and helps to protect the hardware from static discharges. Approval by Underwriter Laboratories of the electrical system, not just the line cord, is also desirable from a liability standpoint. Several failures of unshielded and/or ungrounded evaluation systems probably resulted from electrostatic discharges which occur frequently during Minnesota winters with room humidity below 50. An effective solution to the grounding and shielding problem which also immobilizes cables and thus reduces cable maintenance is to mount the system modules in a rack-type cabinet. This arrangement is available from several vendors. The system power supply must have sufficient added capacity to support any planned future expansion modules.

Maintenance

Most vendors provide 90 day parts and labor warranties and are willing to provide maintenance contracts for longer periods. Most vendors have at least one Minnesota service center and a few have more than one. A few systems are provided with software analysis packages to help isolate system failures and indicate the defective module. At a location with several identical systems, it may be cost effective to stock spare modules and perform repairs by replacing modules. This procedure is most appropriate where the system design provides a modular approach.

RECOMMENDATIONS

INTRODUCTION

Minnesota is in the enviable position of having instructional computing capabilities available to public school districts which serve over 90 % of elementary/secondary/vocational students in the state, and 100% of all public higher education institutions. The bulk of the instructional computing service is provided on a large central timeshare system, which has an extensive telecommunications network providing equal timeshare service to all users throughout the state. Additional service is available on smaller systems owned by a member system, a regional center and a school district. Complementing the access to the computer is a user services people network which provides instuctional support related to system use, and use of application programs and programming languages; applications software support in the form of acquisition, development, maintenance, and documentation of programs; and general support in the form of response to problems, questions, and requests which arise on a daily basis.

In the near future, microcomputers will play a major role in instructional computing in Minnesota. Support services staffs view microcomputers as an integral, rather than a supplemental component of instructional computing, and as such, propose to give microcomputer users support similar to that currently provided to users of the timeshare system. When planning for support of microcomputers, one must consider the large number of different systems presently on the market and the relatively small number of support services personnel. It is a physical impossibility, given present staff sizes, to provide a high level of support for all microcomputers. Therefore, it is necessary to set priorities and define the level and type of support microcomputer users can expect to receive. Appropriate resources, including, hardware, software, and staff, must be directed toward providing the support necessary to assure quality use of microcomputers.

RECOMMENDATION 1:
ACQUISITION OF MICROCOMPUTERS

It is recommended that MECC release an Invitation for Bid, included as Appendix H, for the acquisition of microcomputer systems by any non-profit, educational-related organization. The statewide bid will define for manufacturers and vendors the specification of a microcomputer to be used in the educational environment. The bid will also serve as a guide to educators needing assistance in acquiring a system to meet their needs. The resulting statewide contract for purchase of microcomputers will also allow MECC members and others to purchase a general purpose microcomputer system at a reduced cost.

As a part of the MECC Microcomputer Study, an Invitation for Bid was prepared which specifies a complete microcomputer system, including all essential peripherals. Support of one all-purpose microcomputer system will enable instructional computing service staffs to better serve the needs of users who purchase systems under the state contract. The estimated number of microcomputers purchased through this contract will vary from 30 to 100 (or more) depending upon the price of the contracted system.

RECOMMENDATION 2: SUPPORT

It is recommended that MECC provide support services to Minnesota's educational users of microcomputers. The issue of microcomputer user support is multidimensional and includes the source of support, the type of support, and the type of microcomputer. Support can come from three primary sources: 1) staffs of MECC and member system component organizations. At this time it appears that TIES and the University of Minnesota are in a position to provide support services to microcomputer users; 2) microcomputer vendors; 3) users. The types of support fall into four general categories: 1) purchase, installation, maintenance, and documentation of the system; 2) training in system operation, use of applications packages and programming languages; 3) acquisition, conversion, development, maintenance, documentation, and dissemination of applications packages; and 4) response to questions, problems, and requests. For the purpose of supporting microcomputer use in education, two categories of support will be identified. Highest level support for the 1978-79 school year will relate to the system which was awarded the statewide contract (see recommendation #1). Lower level support will relate to all other microcomputer systems.

A major concern of those purchasing microcomputers is the availability of applications software support. Because timesharing users throughout the state have acccess to the MECC Timeshare System, the most feasible and economical solution is to download programs from the central timeshare system to the microcomputers. A subset of the programs in the MECC system library which are appropriate for microcomputers will be converted and stored in an appropriate form for the selected system. Users having these systems will be able to call the timeshare system and have the appropriate program loaded electronically on their microcomputer. They can then save copies of programs on disc. This procedure reduces the need to buy, sell, reproduce, and mail tape cassettes or discs. this support will be available for all MECC timesharing users and would include efforts to facilitate the normal modes of instructional computing. A by-topic comparison of how well microcomputers and all-purpose timeshare systems complement the instructional computing modes is presented in Table 6.

TABLE 6

COMPARISON OF COMPUTER SYSTEMS

MODE OF USAGE	MICROCOMPUTER * (GOOD-FAIR-POOR)	REGULAR TIMESHARE SYSTEM (GOOD-FAIR-POOR)
Computer Programming	G - F	G
Computer Science	G - F	G
Computer Literacy	F	G
Computer Assist. Inst.	P	F
Computer Managed Inst.	P	G - F
Computer Based Inst.		
Problem Solving	G	G
Algorithms	G	G
Simulations(Gaming)	G	G
Information Retrieval	P	F

* As specified in Appendix H

It is interesting to note that the major weaknesses of microcomputer systems currently are in the areas of availability of CAI languages, ability to perform repetitive calculations, and the storage and movement of large data files. However, the majority of the instructional computing modes can be accomplished quite well with microcomputers.

It is recommended that MECC continue to analyze microcomputer hardware and software technology and continue to disseminate the resulting information to the Minnesota educational computing community. As this report is being written, new advances in microcomputer hardware and software are being released which will enable educators to do more computing in ways not possible on existing microcomputers, and at reduced costs. A variety of microcomputers can do some very exciting, innovative instructional activities, such as recognize human speech and respond with human-like "sounds", and plot all types of graphs. At the same time, some of these same microcomputers have difficulty sorting numbers. Few microcomputers have any authoring language facilities. The size of programs and numerical arrays are limited by the size of the machines. Microcomputers cannot do everything for everyone. It is absolutely imperative that people interested in obtaining a microcomputer system first analyze the needs for which they intend to apply a microcomputer. An analysis of instructional computing needs should involve each of the following areas: intended application, resource management, support, hardware, and software. A list of questions in each of these areas is included (see Appendix J) to help a prospective microcomputer purchaser to make the correct decisions and obtain the right microcomputer system.

Until recently it was difficult to obtain information about microcomputers and their educational utilization, but this is no longer the case. Numerous periodicals, professional, commercial, contain an adequate number of articles on the topic. Selected conferences either are devoted to microcomputers, or have strands which focus on them. Shopping centers now sell microcomputers and related items. While it is becoming easy to obtain information about microcomputers, the fact remains that those in need of specific information must look for it. Therefore, MECC can and should serve as the clearinghouse of microcomputer information for Minnesota.

****

On September 21,1978 the MECC Board of Directors authorized MECC staff to proceed to contract execution with APPLE COMPUTER, INC for its 32-K disc-based Applesoft microcomputer system. An estimated 250 units should be sold by APPLE through MECC to educational users in Minnesota.

CAI IN THE HOME MARKETPLACE

Silas S. Warner
Submitted to the Third West Coast Computer Faire, September 18, 1978

Americans tend to believe that the place for instruction and education is in the schools. That includes computer-assisted instruction. Think of CAI, and you are most likely to think of a computer in a school or university.

Actually, the schools in general have proved generally unreceptive to the use of computers. There have been exceptions, to be sure -- schools that have seen and encouraged the power of the computer to capture the student's attention and enrich the learning experience. But the usual emotion found in elementary and secondary schools on mention of computers in education has been fear - fear of obsolescence and replacement.

The same fear can be found in the university environment. It takes a different form. Universities and colleges are expected to experiment with new learning methods, and CAI systems of great utility have been developed primarily in universities. But the CAI market in universities has been greatly hindered by just this experimentation. The developers of a CAI system are almost never willing to abandon their brainchild, even if the alternative is superior in every dimension.

So far, the biggest successes for computer-assisted instruction as a product have come in the areas of business and industrial training. Business firms are not afraid to buy high-technology systems if they can be justified by benefits. And business firms have the money to spend on new, expensive technological tools like computers. But the success of computer-assisted business training has had its price. The narrow, specific goals of business training programs rarely allow CAI designers the opportunities to build enriching educational systems. And the entire business training field is miniscule compared to the total educational system in this country.

Up until recently, those markets have been the only area where CAI could be sold. Computers have been thought of as "Business machines", with card-readers and printers - like mathematical assembly lines, handling "jobs" in sequence. Ten years ago, only premium-priced and experimental computer systems even had video displays!

Well, that was the bad old days. Home computers, once the exclusive property of electronics hobbyists, have exploded onto retail counters everywhere. Most of these systems are designed from the ground up for interactive computing. Card readers and printers are the expensive peripherals these days!

The new emphasis on computers designed to interact directly with people not only makes computer education possible.,but almost dictates an emphasis on educational applications.

Consider, for instance, the Apple II. Like most computers designed for the home market, the Apple II consists of a processor, memory, and keyboard in a single package. It uses a video screen for its output (a TV set in the case of the Apple II) and stores data and programs on audio cassettes. A BASIC interpreter is built into the operating system in ROM. Unlike some home computers, the Apple is also directly programmable in machine code, using a crude assembler.

BASIC is an acceptable language for most small programming applications. But for most instruction, it is inadequate. Apple BASIC is woefully inadequate in its ability to accept and analyze input from the user. Like most BASICs, it can only accept groups of numbers, or store the entire line of text given to it as a "string." This is completely useless when a program must accept more than multiple choice answers.

A solution to this problem would be to forget BASIC entirely and to move to a higher level-language, such as PASCAL. A simpler solution is to provide a language which corrects the deficiencies of BASIC, while using BASIC in the areas where it proves strongest.

A simple CAI language is already in existence, called PILOT. PILOT is not a complete language, but a means of enhancing a simpler language (such as BASIC) with a structure of student input matching and judging. BASIC supplies variable allocation, computation, and (in the case of the Apple) graphics, while PILOT controls branching, text presentation and input analysis.

The PILOT language is a relatively new development, and therefore not fully standardized. The simplest PILOT is called "core PILOT" and consists of just 8 statement types. One of these types, the C statement, allows statements to be executed by the "host language", or BASIC. (See Figure 1A.)

The command field of each statement in PILOT can be followed by a "modifier," which in most cases determines whether or not the statement will be executed. The most common modifiers are "Y" and "N". The Y and N flags are variables reserved for PILOT's use.

When an M command is run, the Y and N flags are set. If the words to be matched are found in the input, the Y flag is set to 1 and the N flag to 0. If the match is not made successfully, the N flag becomes 1 and the Y flag 0. After a successful match, then, statements modified by an "N" are not executed. After an

unsuccessful match, statement modified by a "Y" are skipped. In this way, PILOT can branch the student depending on the content of the last input line.

Core PILOT is of course the simplest "complete" version of PILOT. Currently a PILOT standards committee is a work on a slightly larger PILOT to be called "common PILOT". Common PILOT has several more statement types (see Figure 1B), including statements to specify input and output to a file.

Some features of common PILOT have not yet been standardized. The standards committee is still wrestling with a universal format for the G command - and it's no wonder, with the many types of graphic terminals and computersavailable today. The committee is also trying to decide on extensions to the M command - special characters which allow synonymous words and word choices in the student input. I might add that the standards committee is very interested in input from the CAI community in general, and from PILOT users in particular.

I first became interested in the development of a PILOT for the Apple II almost as soon as I got my machine in February of this year. The first obstacle to my implementation was that the Apple firmware was not capable of creating or modifying source files in any language other than BASIC. This led me down a long detour, and eventually to the APPEN-1 text editor, now published by MUSE. APPILOT is designed to accept source programs written by this editor.

APPILOT commands are similar to the commands found in common PILOT. (See Figure 1C). There are a few special commands not found in common PILOT. One of these is the L command. Since the Apple II displays its text on a video screen, positioning commands are essential. The L command has in its tag two expressions, which determine the vertical and horizontal positions of the (invisible) text cursor.

The C command and the G command are APPILOT's two links to BASIC. The tag fields of these commands are executed as BASIC statements. This shortcut to implementation of gra phic commands was decided upon because the format of the G command has not been fixed, and because Apple BASIC already allows a fine range of graphic commnds. Any BASIC statement can be used as the tag of a C or G command, but certain BASIC statements such as GOTOs and GOSUBs are guaranteed to foul up the system. These problems cannot be corrected without modification to the BASIC built into Apple.

A controversial feature of APPILOT is the I command, which allows the user to set the Y and N flags from PILOT without an answer match. Many PILOT developers I have spoken with feel that this command is superfluous. But in my estimation, the the benefits of easier judging of complex answers outweigh the disadvantages. Consider the follwing situation: the student must type in a number which must be between two limits and even. In common PILOT, this statement might be handled:

```
*ANS A: INDEX%
T(INDEX>MIN AND INDEX<MAX AND INDEX MOD 2=0):
    THAT'S RIGHT!
T(INDEX≤MIN OR INDEX≥MAX OR INDEX MOD 2≠0):
    NOT A VALID INDEX.  TRY AGAIN.
J(INDEX=<MIN OR INDEX>=MAX OR INDEX MOD 2≠0):
    ANS
```

In order to prevent all this expression typing and evaluation in APPILOT, an I command can be used:

```
*ANS A: INDEX%
I: INDEX>MIN AND INDEX<MAX AND INDEX MOD 2=0
TY: THAT'S RIGHT!
TN: NOT A VALID INDEX. TRY AGAIN.
JN: ANS
```

The development of APPILOT as a language is one step toward the home CAI Market, if effective and interesting lessons are made available for it. PILOT is one of the few languages that already boasts a large and expanding body of tested courseware. Most of this courseware is in a version similar to common PILOT, and much of it has been developed by medical and health sciences schools. In cooperation with MUSE, I have been pursuing the rights to translate these lessons into Apple-II tapes. At the same time, a curriculum in CAI and PILOT is being developed for people who wish to create new lessons in APPILOT.

Even with a body of courseware developed for it, APPILOT is only an evolutionary step. One of the advantages of a command-oriented language like PILOT is that it is infinitely expandable to handle the greater capacities of future machines, simply by adding new commands. I would like to speculate on future developments of APPILOT, and describe what I see as the teaching computer to be found in the homes of the future. I will not describe the processor of this machine, but concentrate on the features most important in CAI - the peripherals by which the machine and the user interact. (see figure 2.)

Certainly this future CAI system will have a screen. Probably it will be capable of mixing computer-produced text and graphics with video from other sources such as video disk units. The system will be able to produce a wide variety of sounds. Certainly it will be able to synthesize speech, in order to communicate with pre-reading-age children. Finally some type of hardcopy printer will be used for record keeping, and for producing assignments. The students can't sit at the console constantly, so the system will need to be able to generate "homework!"

The CAI system will also be able to accept information from the student in many ways. For a long time to come, computers will have typewriter keysets. But as software improve s

the keyset will become more and more a vestige used only by technicians. A standard keyset is just not flexible enough to accept all the possible varieties of student responses.

Instead, the primary input to the computer will be through devices whose functions and the labels attached to them, can be redefined by the program. There are many ways to do this. The simplest in theory and the most difficult in practice, is to place small display devices on the keycaps themselves, and to change the labels of the keyset on command. This approach has been investigated by peripheral manufacturers, but I know of no successful product or application of this idea yet.

The simplest method of redefinable input is a light pen which designates areas of the display screen. The light pen is a photocell which can be placed over any area of the screen which is lighted. The cell directs light pulse caused by the passage of the electron beam through the area covered.

There are many other approaches to this same problem, including various devices for registering finger-touches on the screen, Certainly one or another of these devices will be a part of the future home CAI system. There will also probably be a speech recognition device to complement the speech output for early childhood education. Other, more exotic inputs will probably be available as options, including devices simulating real-world situations such as aircraft.

Ten years ago, such a system could not have been predicted before 1990. Today, I believe that everything I have described will b e implemented on an Apple-based system in 1980, at a price of about $3,000. Everything I described either has already been implemented on the Apple, or is now being implemented in experimental form.

Without software, though, such a super-system will be just a collection of experimenter's hardware. The challenge to developers is to create useful lesson packages that enable the computer to really teach. And that means not just generating pretty software -- that means creating complete, integrated and tested curricula. It means that not just computer expertise, but educational knowledge is required. And even the finest and most complete educational methods can fail if misapplied - so educational curricula must be tested.

Software development organizations capable of all this already exist - in large universities and corporations. So far, they and their customers (corporations and governments) are the only entities that have been able to afford such complete development. But that is changing fast. Consider the economics of the home system I have just described. It would retail for $3,000 - less than the cost of an automobile today. It would require new courseware on a continuing basis, to keep up with the educational development of its users. But it would last far longer than a car: there is no theoretical reason why a well-built home computer should not last for five years or more. My trusty computer, running the "Loan Amortization" program, tells me that the capital cost of such a system at present-day interest rates would amount to a monthly payment of $69.65. Even with $50.00 per month for new curricula, a CAI system for the home could be available in the next decade for the price of a car.

To sum up, by 1980 the hardware and the financing should exist to make the educational computer a part of the American home. Only the software will be needed, and that is where all of us here are equipped to help.

A SCHOOL'S NEW STAFF MEMBER

Gerald Hasty
4884 Irene Avenue Las Vegas, Nevada 89110

A modern instructor needs to be a resourceful and creative person. Taking a cue from the business world, school administrators are assuming accountability. Just what are the returns from a given expense in education, or what is the value received? The wells of tax monies that formerly flowed freely are beginning to dry up. The bond voter is reluctant to approve an issue that may not return a student that can meet expected criterion goals. Unfortunately the measurement of student performance is imprecise. The classroom instructor that is unwilling to develop modern skills of evaluation does little to help eliminate that impreciseness. Given the expanded goal to justify student grades and the need to diminish criticism, the competent instructor will be found making the most of what is available. Another way of looking at it is that good instruction is a matter of strategy.

A computer in the classroom can make an excellent teacher's assistant. The reader is well aware of what kinds of tasks a computer is suited for and thereader is knowledgeable of many of the micro-computer's strengths and weaknesses. But in working a strategy for the classroom, of what tactical advantage is a computer assistant? Hopefully a little poe-in-the-sky like this paper will help in critical thinking on that question. Of all the areas for possible discussion, let us focus out attention on student testing.

Evaluation is a process used to develop feedback. Testing, the formal judgment, loops back to the instructor, the student, the parent and the administration. Having this data then makes decision-making less hazardous. The instructor can determine if the instructional objectives are being met. The student learns how faraway from the bullseye they are. And the policy-makers can gauge the effect of their policies. In other words measurement overcomes subjectivity and monitors learning and diagnoses strengths and weaknesses. The greatest benefit of a micro-computer to the faculty is as a means of evaluating their teaching effectiveness and whether or not the goal and objectives are reached and discerned. Is the teacher-made test appropriate? What about test validity? Does it really test what it is supposed to? And finally, a computer can be used so that the teacher can tap the feelings and thoughts of pupils. What is intended is called an objective. What is taught is known as the lesson. But what is measured is the achieved outcome. The outcome that is demonstrated over time. But achievement is what is accomplished in the present. It is the testing of achievement that the broad subject of testing will be narrowed down to. What are the elements of a computerized testing program?

As is true of every endeavor, there must be planning. In the planning stages are steps. Step one, set down the class objectives. Now that the targets are identified, a table of specifications is the next step. This is where you ask, "How will the student demonstrate mastery of the objectives?" With these two steps climbed, the next one is the construction of the questions themselves. Good tests are not made up overnight, but require detailed planning. What is beginning to emerge is a data base. This base consists of potential test questions keyed to the objectives and the table of specifications. Building a test can then be like going to the supermarket. The objective is to feed your family; the table of specifications is the shopping list and you fill your basket with selected test questions. To handle the data base manually there re the card files. But greater versatility can be achieved by setting the data base into a computer. The test builder is now prepared.

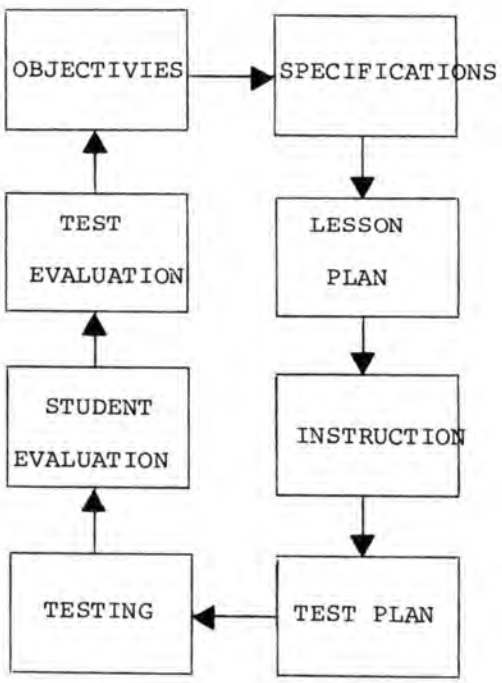
Out of the question pool a test is to be constructed. The easiest way to describe this process is through a scenario. Imagine sitting down to a CRT display and keyboard. Handy to you are your notes outlining the points the questions of this test are to cover What is on the screen is a possible

question to use that was found by a key-sort system. Accompanying the question is analysis information collected from previous uses of this question, and information on where it fits into your lesson plan. You look it over and decide that the item will not discriminate well. The next question is then called up. This item shows a good track record and you decide to use it. Off into another file the question is saved. You decide that just one question on that objective does not represent the importance you place on the objective. The next question is cued and if accepted placed into the file. The process is repeated untill all items are covered in sufficient detail. Then the micro-computer is signaled that you are finished and satisfied with the new test. The items are randomized and printed out to the printer. From the printer you pull the test that has a proper title and instructions writen in. From the output the multiple test copies are made. This has been a description of a phase of micro-computers and word processing being used to assist the teacher. But there is more.

In an advanced stage of development of this idea the test could be scored by an I/O interface to the micro. Failing that the hand scored test is ready for item analysis. It is from this critical step that poor and mediocre test-making is separated from the test-making of an instructor with access to computing power. From the information gained at this time, future tests are made. At this time, in an objective fashion, test items can be justified as being either a good item or a bad item. The good items are kept and are used to justify the grades assigned to the raw scores, whereas bad items can be dismissed thereby helping to improve the amount and quality of learning. A good measuring instrument has validity, reliability, and objectivity. How would your test measure up? Without a micro, that could be a formidable question. One key to any success is consistency. By using a micro-computer to reduce the workload, greater consistency can be had for less labor and quicker results. The micro as an assistant can be called upon to check the consistency of goals and of student performance. Also consistence is gained from the use of micro's, by allowing timely reports to be generated. If an activity takes too much time, then no matter how good it is or how helpful, there is little kikelihood of the procedure being used.

One well-used statistic from test are means. The mean is the arithmetic average jof test scores. It is from this mid-point that the C (average) grade is located. Such a straight mathematical problem as averaging is a snap on the computer. Unfortunately such a number tells very little information. There are fortunately other tools for interpreting an examination score. A series of test scores that are close together tell a different story than scores that are further apart. The measurement of that dispersion is called standard deviation. It is with such information that it is possible to compare a test score by a student with other test scores of their's. Rember that apples and pears cannot be compared but fruit can be. In the same vein an evaluation can be made among several tests if they are set to a standardized score.

As was metioned earlier, ther is a need for consistency. The consistency of an examination is usually determined by an item analysis. Again the computer is capable of taking a lengthy process and distilling it into a usable output. Such an output allows the instructjor to evalutate their test-making and thus we have come full circle.



The tools of an item analysis are such procedures as: Kuder-Richardson Formula 21, standard error, item difficulty, and item discrimination. These procedures are strictly arithmatic. Can the reader think of a way to perform a large amount of calculations flawlessly?

A brief view of those tools will be shown. The KR21 is a process of assessing the degree of reliability of an examination between test items. The standard error of measurement attempts to determine the degree of agreement between an obtained score and the true score. Item discrimination and item difficulty evaluates each test item for its ability to differentiat between high scores and low scores. The question is answered as to whether the item is difficult enough and if it separates those who know the question from those who do not.

While this is not meant to be a detailed treatise on student testing; it is hoped the reader will find the challenge of using a micro as a tool that will allow greater precision in measuring student evaluation. A procedure has been outlined and discribed for the use of a small computer that is commonly available to anyone. To any possible critics, it will be noted that these concepts are already used on main frames. With the intelligent use of testing evaluation it is possible to raise the quality of teaching far beyond any possible hit or miss methods that might be found. In a world of finite resources and infinite demands, today's competent instructor must apply the knowledge of modern evaluation procedures; the goal is reater precision in resource allocations.

RESOURCES:

Instructors and Their Jobs
W. R. Miller & Homer C. Rose
American Technical Society
Chicago, 1975

Human Development and Learning
Hugh V. Perkins
Wadsworth Publishing Co., Inc.
Belmont, CA, 1974

Measuring Educational Outcomes
Bruce W. Tuckman
Harcourt Brace Jovanovich, Inc.
New York, 1975

MICRO COMPUTERS IN THE HIGH SCHOOL - EXPANDING OUR AUDIENCE

William J. Wagner[1]
Mountain View High School
Mountain View, California

Introduction

This talk is a slightly revised version of the one I presented at the Second Computer Faire in San Jose in March, 1978. To those few here who heard me then, I apologize. To all of you I promise to try to have something new to say next time.

The title of this talk is meant to be ambiguous, for I will argue that two audiences are being expanded: the introduction of micro computers into the high school will allow computer education to be offered to more students; and educators like me represent a very different set of users of the formerly hobbyist-oriented micro computer. So if you think my feet are shaking up here, it is merely because I am playing the role of the tip of the iceberg.

Computer programs in schools are not new. What is new is that micro computers open the possibility of bringing computer education to many more students within the strictures of modern school budgets.

This broadened market represents a real challenge for computer educators. In every school there exist some students (about 1%, I think) who will, if you let them, spend all their waking hours at the terminal, making the machine jump through hoops, and occasionally emitting those familiar gales of laughter. They are great people (they are us, right?). Every teacher needs these kids around - they keep the adrenalin circulating. We are proud of their programs and like to take credit for their progress. Indeed, we should take some credit, but not for teaching them much. Our contributions are in suggesting interesting new kinds of problems and in getting the equipment in their hands in the first place.

Important as it is to give these future professionals and/or hackers their first boost, our programs should not be judged or justified according to the progress of these superstars. Schools are full of lots of different people, and it is this other 99% which I intend to discuss here, and to which the program at Mountain View High School is directed. Also, although I am by no means a computer hobbyist and have only limited knowledge of the micro computer field, I will present my views on the special requirements of high school computer facilities, and why we selected micros.

Throughout this talk I will also ask for your help. Computer educators need better ways to find each other and share their ideas, plans, successes, and even failures. After my talk in San Jose a very useful group of "Computer-Using Educators" (the name we have given ourselves) was formed and our fourth meeting will be held in January. Anyone interested in being put on our mailing list should write to Don McKell at Independence H.S. in San Jose.

The Students

Without risking more guesses about percent composition, here are five categories of students, for the purpose of discussion:

A. The computer hotshots. These have been discussed above, but I should add that they are not necessarily the best students in other classes, and may even have trouble getting computer assignments in on time. We can really help them by focusing their energy and enthusiasm toward activities which will continue their growth.

B. The good students in math and science. These are our natural audience, but sometimes it seems surprising that they do not flock to the computer. For one thing, they are very busy with course work, and school courses are rarely set up to reveal how the computer might be of use. And the tightly packed college prep schedule has little room for electives.

C. Other good students. This may be the largest group in any school. They also are busy with various school activities, but unlike group B, computers are not on their list of things to pursue some day. Further,

---

[1] Home address: 127 O'Connor St., Menlo Park, CA 94025

they often show a remarkable disinterest in what we like best - computer games and programs which crunch numbers.

D. Kids with lots of interests, but not particularly good students. Some of these might be interested in our hardware, but not so much in how to manipulate it. Others find programming to be their first interesting "academic" class. Often their math level is so low, however, that the number of problems that can be suggested is limited.

E. Unmotivated and unsuccessful students. This is another large group. They find their way into computer classes because their schedule is not full and because someone, usually not the student himself, thinks that something flashy and new may produce a change. It turns out to be true in some instances.

I will mention in passing another group which has remained an enigma to me - young women. At my school they make up at least half of the enrollment in advanced math classes, but seem impossible to attract to the computer program. Those in the programming classes are every bit as good as the boys, but only one or two out of the twenty or so students who spend extra time programming or on games is female. I look forward to discussing this situation with persons interested in sex differences in math and science.

The Importance of Expanding our Audience

I think that we designers of computer curricula have tended to think in terms of groups A and B only. It is very difficult to find a programming book that does not assume knowledge and interest in higher math. Also, we have been content with the few students from those groups who did flock around us because we couldn't have served many more with our limited facilities anyway.

It is important for us to reach out to this large majority of students who do not naturally find their way to the computer room. It is obvious that computers are actually a part of their lives whether or not they like or understand them. However, few opportunities will exist for them to interact actively with computers after they leave school.

Furthermore, these students tend to be alienated from the computer activities at school - they think it is for smart people only, and they think that demented laughter and secret language means you have to be a little crazy too. (A colleague at a nearby school who brought her Algebra I class to the computer room for a week of introductory BASIC told me a wonderfully poignant story that fits here. As a 10th grade girl sat down at the terminal for the first time, she looked around nervously and inquired, "Can anyone see me in here?")

How can we reach these individuals who do not automatically think of computers as their thing? It is a difficult teaching and strategic problem, but we must continue to look for ways, for several reasons. First, this could be their last chance in a relatively non-threatening environment to have hands on experience with a computer. Once they leave high school their education and experiences begin to narrow toward a vocational or academic or lifestyle choice which will preclude taking a flyer into an apparently unrelated field like programming. Their contacts with computers will be passive: inspiring awe, fear, or anger.

Also,the opportunities that do exist out there are more restricting. The college computer classes I know about are not intended for the liberal arts person, but for the future professional or technician. Thus we must think of some of our teaching as part of a general education. And for those students too busy, nervous, or skeptical to commit themselves to a full course, we must find ways to bring computers into the classes they do select.

Another reason to teach these students about computers is that we can offer valuable experience in problem solving and thinking, as well as a new vocational direction. This area matches so many of the goals expressed by various departments of my school and of the administration, that I am sure that if the slate were wiped clean and a new public school designed, courses about programming and computers would merit permanent status in the school.

Computer instruction of some kind would be a natural part of most students' coursework: either as a regular course like Geometry, the first step onto the college bound math track, or the equivalent of an elective in English or Social Studies, or a vocational offering like Woodshop or Auto Mechanics. I have found that experience with computers and programming can help teach math more effectively, can help students problem solve and organize their thoughts, and of course can lead to jobs at various levels of academic preparation. Revolutionary school reform is not imminent, so it is the responsibility of us educators to work for the gradual increase in use of computers in schools so that these benefits can be made available to more students and apparent to more administrators.

Finally, I think that almost anyone in high school who is willing can be taught programming in BASIC at some level, and we owe it to them to provide this opportunity to be in control of the computer for once. The problem is pedagogical rather than one of prerequisites. What we need is a set of assignments for various levels of student ability and interests. It would be glib for me to say that this is an easy task, but I am busy working on the problem, and believe it can be solved satisfactorily.

The Curricular Offerings

At Mountain View High School we have tried to broaden the audience of our computer offerings by using six different approaches: A two week course in BASIC offered as a unit in certain math classes; simulations and games which have applications in various subjects offered in the school; easy access to the computer at designated times for game playing; classes in programming which offer experiences at three distinct levels of math ability and interest in programming; a well attended adult education class in BASIC at night; and an introduction to assembly language programming for a few students.

First let's talk about what we do not do. There is no CAI (computer assisted instruction) per se because I have not figured out how to accomplish this in a class situation with our present facilities, and I await the software which can provide the organization, record-keeping, and pacing which are required in order for CAI to actually be a help to a teacher. Also, I await the teacher who wants to give it a try. Another thing we slight is instruction about hardware. This is simply because I know almost nothing about the wondrous world inside the machine. I am a teacher, a programmer, and somewhat of a mathematician, but I have had a nervous awe of electronics ever since my younger brother became a ham operator in the sixth grade and began speaking in tongues. I hope that others will know some ways to approach these omissions and will suggest more things we could usefully pursue.

A final comment on these offerings: a school district cannot get the full benefit of computers in the curriculum without a real commitment. As long as no one has released time to coordinate these activities, to encourage other teachers to get involved, and to provide in-service training, we will not be able to realize the full potential of these offerings. My own proposal to be part-time in the classroom and part doing this kind of coordination was a victim of Proposition 13.

The Two Week Course. This has been the backbone of our program. It has brought programming to a large number of students and it has been effective in building enrollment in our programming classes. Furthermore, it was a major factor in gaining permanent status for the computer program.

This course is based upon two premises: (1) A remarkable amount of programming in BASIC can be taught in two weeks, and (2) if programming is introduced as a non-optional part of a course, then some of the barriers against participating are transcended (for example, none of the regulars are around to make people feel dumb, and most students tend to try harder when something is expected of them, compared to a situation in which the activity is optional).

The two week course was the way computer programming got its foot into the door at the school. Using 4 rented teletypes timeshared to a HP 2000F at the Santa Clara County Office of Education, we gave two weeks of instruction in BASIC in 9 math classes, from Geometry to Calculus, while modified versions were given to three lower level classes. I received 40 % released time, and taught most of the classes, with assistance from the regular teacher. This pilot project to prove the value of further investment was an exhilarating experience and a great success. I am confident that the same proportion of students who pass these classes came to understand the essentials of the following statements: INPUT, LET, GOTO, PRINT, and IF..THEN, how to work with strings, and how to use the Hewlett-Packard system commands.

At the minimum each student wrote and ran 6 programs, some related to their coursework, one using strings to produce a conversation with the person at the terminal, and some of general interest like computing a batting average, or a grade point average. I must emphasize that this was the minimum. In each class there also appeared advanced programs such as the one which computes change in correct coin denomination, prime generators, and a number guessing game. My favorite of these was a program in which the computer guesses the person's secret number. Each of the writers of these particular programs had never programmed before, and accomplished these results within two weeks.

It is gratifying that as we enter our third year of computer instruction, there are now three math teachers in the school who have incorporated the two week course into some of their classes.

Educational Games and Simulations. HANGMAN in French and Spanish (and also in Tagalog and Ilocano!), LUNAR in Physics, CIVIL and ELECT in History classes, math drill in remedial classes, NEWTON (getting across the stream against the wind) in Trigonometry, demonstrating limits and areas in Calculus. These are all activities which we have carried out with classes. We reach a lot of students this way, and a lot of teachers. There are more things we can do, and improvements we can make with these. We have not, for example, offered a coordinated unit using, say, the Huntington teaching materials. If others have, I would appreciate hearing from them.

The problem of interacting with other teachers and their curricula is an extremely delicate one. We computer teachers must tred carefully - our programs and equipment must be easy to use (or we supply student assistants), and our offerings should be more than just time fillers. The entire class should be able to get involved, and this may stretch our imaginations, given our limited facilities or space.

Also, we must work to convince our colleagues that we can contribute to their instruction, because don't forget that they are probably not from either of the first two groups described above. I hope that other teachers trying the same things will get in touch with me, for this is an area of great importance to the increased success of our programs, and one in which successful strategies are very critical.

Here is one more comment about these games and simulations. Have you noticed how so many programs reflect their original association with teletypes? Responses to correct answers, a crashed lunar lander, or even exultation over a conquered wumpus can certainly be made more interesting as we make better use of the screen capabilities. Also, we should consider carefully how average kids react to screenfuls of text being thrust upon them. The good old teletype was more their speed, I think.

Game Playing. The computer facilities are open various times during the week for game playing by any student in the school. There are many games, most with little educational value. We do not encourage these games, but they do attract kids. We try to show them the more interesting (from a teacher's point of view!) alternatives to FOOTBALL, BLACKJACK, etc, and there are many on our system. Don't underestimate these games however! I use a gambling game to demonstrate adding and subtracting negatives (kids love it when you bet -$1000000 and lose), and also to show how computers are only as smart as the programmer.

Now that we have changed over completely to our own computers there is more control over the available games, but game playing continues to be an important part of our program. There is a lot of thinking and problem solving required by the right selection of games, and I think such things are a useful addition to a school's extra-curricular offerings. Game playing leads some kids to inquire about programming, gives others ideas for original programs, and at the minimum lets a kid interact with a computer in a non-passive way.

Programming Classes. We offer 2 or 3 sections of a one semester course in which the primary activity is teaching BASIC. In the course description I made Algebra "recommended but not required", which is consistent with my desire to reach more students and with by belief that programming can be taught to a wider range of people than is usually thought possible. Students from all of the groups A through E described above enroll in the class, and it has become apparent that the course could not follow a traditional format of lectures and assignments.

The course now operates on three different levels. Some students just work on programs which they and I agree would be productive, and ignore the regular schedule of assignments. The schedule involves completing four chapters of our text, but almost every book or program assignment is divided into two levels of difficulty.

I would be glad to discuss the course in more detail with anyone interested, but let me close by highly recommending our text - Computer Programming in the Basic Language, by Neal Golden (Harcourt Brace Jovanovich) - it is cheap and full of good assignments at almost every level. It does, however, assume a minimum of Algebra I, which has caused me to generate a lot of supplementary problems.

Adult Education. A night class in BASIC using our facilities is being offered each semester. Right now the format is 6 weekly meetings of three hours each. I consider this an important part of our program, with potential for exciting growth. If you think about a public school as a community resource, many uses come to mind for an accessible public computer facility downtown. Furthermore, the Adult Education Department has been very supportive of the program, and gladly agreed to support purchase of one of our computer systems. We are interested in talking to others about these possibilities.

Assembly Language. We have been very fortunate to have my dismal knowledge in this area supplemented by the weekly visits of a volunteer computer scientist from NASA, who meets weekly with 4 or 5 kids. This arrangement was made through the coordinator of the volunteer programs at the school. I continue to look for more volunteers to supervise the computer room or help students in return for having computer access.

Getting the Program Funded

Of primary importance in the initial stages of our effort to establish a computer curriculum at Mountain View High School was the involvement of several dedicated parents, including one Board member. These parents insisted that the school should have a program comparable to the other schools in the District, and did not let the matter rest until it was a reality. Almost an entire school year was required to produce a proposal, and another six months went by before final approval was given.

In the spring of 1977 we began with four rented teletypes and offered the two week course described above. Its success produced wide spread interest in the computer throughout the student body, and led to a decision by the Administration and Board to make the program permanent. Other computer programs that I am familiar with in schools gradually grew over the years - a teletype here, renting or scrounging time there, with the continual struggle for marginal improvement each year. I admire those individuals with the perseverance and ingenuity needed to build a program this way. At Mountain View High School we are fortunate not to have taken years to establish an adequate computer facility, and our success is due in large part to sustained parent support from the beginning, and to our ability to involve some 300 students during the first four months of the project.

Computer Facilities

The rented teletypes were gradually phased out as we purchased our five micro computers. The school district committed $15000 to the computer program at the school. Although Proposition 13 cut into this amount, the addition of Adult Eduation funds made up for some cuts. We have two Sols from Processor Technology, and two Horizons from North Star; all four run North Star disk operating system and BASIC. (We find it very convenient to have all the computers compatible.) Each of the units has 24K of usable memory, which means that BASIC programs may reach about 10K in size. We use an Okidata printer which may be accessed by each of these computers through a centrally located switch.

We have been very pleased with the Sol-North Star combination, and with North Star software. I feel that disk drives are very much worth the extra expense - the tedium of loading and saving programs on cassette can seriously hurt a program like ours which is trying to reach a lot of students. An important feature of one of the Horizons is a dual disk drive, which allows us to copy all our disks regularly. This is absolutely necessary in a school environment, even with constant supervision.

Recently we have added a TRS-80 from Radio Shack. The disk drive has been delayed, so this machine has not really entered our program. I am quite impressed, however, with Level II BASIC, with the peripheral devices they offer, and with the maintenance promise.

We have been extremely fortunate in obtaining excellent service on our equipment. Any purchase decision should depend upon the ability to get convenient service and help with the countless unpredictable little things which crop up. I feel very fortunate for the help and service provided by the staff of the Byte Shop of Palo Alto and of the Microdoctors nextdoor. Service for the year has amounted to about $600, including full inspection and cleaning this past summer, and major repair to the printer.

The primary reason for choosing micros was cost, for the total purchase price of the four systems will be roughly equal to two years of rental of four time-shared teletypes. I was attracted to the DEC system which time-shares four ports, because of DEC's reputation and all the software which comes with it. But for their price we could purchase six of the Sol systems, and DEC's maintenance is over $2000 per year.

Having become convinced that micro computers had ceased to be strictly a hobbyist domain, I made the plunge and have been delighted. Like the Holiday Inn commercial, the best surprise is no surprise, and for me the best system is one you don't have to think about. We load BASIC in the morning and can continue through the day with students programming, loading, printing, and saving programs. Often at lunch time, or as a reward for my General Math classes, however, the flashy Sol games are brought out.

Computers in schools must be very easy to use. Students should be able to concentrate on the programming and ignore

the machine itself. Teachers should not need special skills or knowledge if the use of computers is to spread to more schools and to more departments within the school.

It is fine if one staff member can fathom the mysteries of machine language or hardware, but the overall program should not depend for its operation on this individual or his/her knowledge. Uncommitted or uneasy teachers will not want to touch a system which seems inaccessible or arcane.

In addition to satisfying these unusual requirements for a micro computer, whose tradition is with the hobbyist, our micros offer nice extras. For those students who wish to pursue assembly language, this is possible. The four systems are totally portable, and can each be dispatched to separate classrooms (or to different homes during vacations!). The FILL and EXAM commands (PEEK and POKE in some languages) bring a fascinating extra dimension to BASIC. And recently students have discovered how to allow data entry without the carriage return, and to make the screen blink ... and so it goes.

Our micros are symbolic of our entire program. Their cost, versatility, and ease of operation make it possible to bring computer education to many individuals who never would have had the experience, and yet they still can provide challenge and excitement for the brightest student. I am very pleased to have joined the ranks of the micro computer, and especially pleased to have been able to join on my terms rather than theirs. There are a lot of teachers and students like me out there, and I predict that next year this place will be too small to hold them all.

SOME EXPERIMENTAL SUPPORT FOR EDUCATIONAL COMPUTER GAMES

Muata Weusi-Puryear, president
Edutek Corporation
415 Cambridge, #14
Palo Alto, California 94306
(415) 325-9965

Abstract

This paper is presented as an argument for the incorporation of games in computerized drills and tutorials. It is based on experiments conducted at the Lawrence Hall of Science, University of California at Berkeley. The experiments sought an answer to the question: "Can the game elements of a computerized tutorial/game motivate student involvement in the tutorial elements to a degree high enough to produce significantly greater achievement than the tutorial elements alone can produce?" The study supports an affirmative response to the question. Students, who had an opportunity to continue their participation in a game if they correctly responded to randomized exercises, achieved significantly more than students who did not have the opportunity to play, even though the game-playing students did fewer exercises.

Introduction

This paper is presented here to provide some experimental support for those of us who are using (or would like to use) computerized games to teach mathematics. We are often confronted with the question: "Ok, I'm convinced that your games motivate children, but wouldn't the children be better off if they spent full time doing the mathematics?" In 1975 I showed (1) under laboratory-like conditions (2) that the motivative effect of games can be great enough to more than compensate for the time lost in playing them. The following is a summary of that study.

Procedure

The study used a computerized tutorial/game called GAMBO. There are two players in a GAMBO tutorial/game. One player is "the student". The other player, who shall be called "Jody", is a computer simulated player. The computer program also exhibits a third personality called "Gambo" who is the umpire and scorekeeper for the game. The output of the GAMBO tutorial/game looks like a script of a play with three characters -- Gambo, the Student, and Jody. Each player in turn is given an arithmetical problem to solve; after he has committed himself to an answer, his opponent is given an opportunity to evaluate the player's answer; the umpire presents the correct answer; if the player's answer was correct, he gets an opportunity to make a move on a three-by-three Tic-Tac-Toe board; and if the player's answer was incorrect he would forfeit his turn on the Tic-Tac-Toe board.

The object of the game is to gain as much score as possible in a limited amount of time. In the GAMBO game a player can receive ten points for correctly answering the problem, five points for correctly evaluating his opponent's answer, and fifteen points for getting three marks in a row on any Tic-Tac-Toe board. A new Tic-Tac- Toe board is presented after either player receives three marks in a row.

Two experiments were designed to be part of one-day-field-trip activities for summer school classes of San Francisco Bay Area school districts. Each field trip started with students arriving at the Lawrence Hall of Science by bus from their school districts in mid-morning. Approximately 30 students (depending upon the number of terminals available that day) were randomly selected from among the students in the 8- to 11-year-old age range. Those students not involved in the experiments were taken to the activity halls where various scientific activities were available for them. The students in the experimental group were placed in every other seat of a small auditorium where they received a battery of tests. These pre-experimental measures were intended to measure the students' skill in the basic arithmetical operations and to inventory their attitudes towards arithmetic and arithmetic stressful situations (such as tests). While they were seated they were, without their knowledge, randomly divided into three sub-groups. The selection was based upon that day's random assignment of the auditorium seats. After the pre-experimental testing, two of the sub-groups were escorted from the auditorium to computer terminal rooms containing enough terminals for each member of the group. During the next forty minutes: one group received a computerized tutorial; one group received the same computerized tutorial interwoven

with a simulated Tic-Tac-Toe game; and the group remaining in the auditorium viewed two films of scientific interest but having no relevance to the tutorial. The computerized tutorial was the GAMBO system's. Students under 10-years-of-age received addition problems and multiplication problems were given to the older students.

The first two groups then returned to the auditorium where a post-test was given to all students on the material covered in the tutorials. Students then left the auditorium and together with the non-experimental group of students had a bag lunch. After lunch, all students had a couple of hours to explore the Lawrence Hall of Science. All students had an opportunity to play games on a computer without regard to their previous groupings. In addition, other scientific activities were available for the students -- physics, biology, chemistry, astronomy, and mathematics. The experimental situation carried no risk to the students -- the entire field trip activity was designed to give all students a positive experience with science.

In each experimental situation there were three groups of participants. The full-treatment group was exposed to the GAMBO system as described above. The partial-treatment group was exposed to the same GAMBO system but they did not see any reference to a Tic-Tac-Toe game and all score--referenced output was surpressed. The no--treatment group watched two National Geograghic films.

The partial-treatment group corresponds to what is commonly called a "control group". As closely as possible we tried to make the difference between the full treatment and partial treatment the presence or absence of the game elements during the experimental phase. The full- and partial-treatments were conducted simultaneously in separate rooms. Through scheduling we controled for possible room effects, sex of laboratory assistants effects, and their possible interactional effects. A teacher from the participants' school was present in each experimental room to aid with disciplinary problems, but they were discouraged from actual participation in the experiments.

The purpose of the simulated opponent, Jody, was to provide a human-like opponent who was "equal" to the student in ability -- the probability that Jody would give a correct answer was equal to the proportion of correct to total answers previously given by the student. When we were in the checkout phase of the GAMBO system we ran sessions with elementary school children similar to those run later during the experimental phase. We observed that students did not give the impression that they were competing with a super-human machine. On several occasions the students remarked that "Jody is stupid," or "Jody is trying to cheat," or attributing other human characteristics to Jody. In addition to the errors Jody made, there was Jody's child-like method of typing. Whereas Gambo's presentations were done at maximum output speed, Jody's presentations were made at a speed of (at best) 1 character per second. Hence, it appeared to someone watching Jody's presentations, that Jody was doing exactly what the students were doing when they typed -- that is, searching the keyboard for the next letter or digit.

The fuction of a no-treatment group is to statistically smooth out the effects of the highly motivating field trip atmosphere. By regressing pre-experimental measures on post--test scores, using the no-treatment group, we are able to obtain formulas for predicting post-test scores of the two experimental groups. These predicted scores used as covariants in an analysis of covariance on the post-test measure provide additional statistical control. Since the no-treatment group is a randomly selected sub-group of the experimental participants, and we can safely assume that they received no training in the concerned algorithm during the experimental period, their performance on the post-test items can serve as a statistical basis for measuring the difficulty of the items.

Summary of Analysis Results

There were 258 students who participated in the two experiments -- 84 in the "addition" experiment, and 174 in the "multiplication" experiment.

The students' attitudes toward arithmetic and their levels of debilitating anxiety did not differ significantly from those of a large national sample of fourth-graders (3) taken in 1962.

Students, who had an opportunity to continue their participation in the game if they correctly responded to randomized exercises, achieved significantly more than students who did not have the opportunity to play -- even though the game-playing students did fewer exercises.

Conclusions

Games can have a positive pedagogical value to mathematics instruction. The motivative effect of games can be great enough to significantly increase students' achievement levels. In particular, games can significantly improve the learning that takes place during a computerized drill-and-practice lesson.

Footnotes:

1. Weusi-Puryear, Muata. An Experiment To Examine The Pedagogical Value Of A Computer Simulated Game Designed To Correct Errors In Arithmetical Computations. Ph.D. Dissertation, Stanford University, 1975. Dissertation Abstracts International, Volume XXXVI, Number 4, 1975. Xerox University Microfilms, Ann Arbor, Michigan, Order No. 75-21.906, 157 pgs.

2. Here "laboratory-like conditions" means a setting and design that closely approximates the type of experimentation done in the physical sciences -- where a quantity is systematically varied and the effects of this manipulation on other dependent variables are measured.

3. The National Longitudinal Study of Mathematical Abilities

A VIDEOGAME MICROPROCESSOR IN THE ELEMENTARY SCHOOL

Al Ahumada and Sam Hersh, Aero/Astro Dept., Stanford U., Stanford, CA 94305
Tel:(415) 497-4061 or (415) 497-4925

This is a report on the successful use of a microprocessor videogame system to help teach basic number theory and computer principles to elementary school children. We will describe the lesson plan and the special hardware and software features of the system which contributed to the success of the lessons. The class was a summer school mathematics class sponsored by the Stanford YWCA for 15 students from 8 to 12 years old. The lessons were judged to be a success because of the high level of enthusiasm and active participation of the students and the positive report of the teacher.

Three one hour lessons were spread over three days. On the first day the class was given a lesson on binary arithmetic in which they learned to convert decimal digits to binary and vice versa. The computers were not present during this preparatory lesson. At the start of class on the following day, two computers were programmed and ready to play a Pong-like game called Wipe-off and each student was allowed to play one ball. A lecture was then given reviewing binary numbers, describing the logical contents of the computer memory and explaining how the display that is seen in the game was just a map of binary numbers in the machine's memory. They were then told that once they learned hexadecimal numbers, they could create their own display on the screen. The lecture ended with a presentation of hexadecimal numbers and for the rest of the hour the students played Hex Reflex--a game in which a binary or decimal number is displayed and the corresponding hex key must be pressed. Scoring is based on both speed and accuracy. The third day's lesson began with the students playing Hex Reflex. Students who were not playing were given graph paper and asked to draw an 8x8 design of black and white squares and code the design in hex. Most students drew their own first initial. The students then keyed their designs into the display memory. For the rest of the hour one computer was used by those who wanted to debug or improve their designs and the other was used for a spirited Hex Reflex competition.

The computer system used was the RCA COSMAC Video Interface Processor (1). The most important features of the VIP for this application are that it is a cheap, completely programmable videogame system. Its interpretive language CHIP-8 is especially designed for writing such games and the 20 games that are provided with the VIP can be used as is or are easily modified for teaching computer principles. The machine leanguage of the 1802 processor is very simple as compared with 8080 or 6500 type processors, so that the advanced student or interested teacher can readily gain complete control of the processor.

Most of the emphasis on computers in education has been on using computers to teach standard subjects. We feel that learning about computers is itself important and that grammar school is not too soon to begin that instruction. Our plans are to add a non-contact touch scanner (2) to our system so that it will be possible to enter information into the computer by merely pointing to various programmable symbols on the display. This will reduce the cognitive requirements made on the children so that next summer we can introduce computers at the kindergarten.

References

1 Hersh, S. & Ahumada, A. Psychological tests with video games. Procedings of the Second West Coast Computer Faire, 1978.

2 Ciarcia, S. Let your fingers do the talking. Byte, 3(8), August 1978.

DISCOVERY LEARNING IN MATHEMATICS

Ludwig Braun, Jo Ann Comito, Philip Reese, Robert Wlezien
National Coordinating Center for Curriculum Development
State University of New York at Stony Brook
Stony Brook, NY 11794
(516) 246-8418

Abstract

Three computer programs specifically designed to enrich the Algebra I and II classroom experience will be demonstrated and described. The programs utilize the discovery approach, self-pacing, and a game format. They are intended to stimulate interest in mathematics and are designed to be used with a wide range of students. These programs use the graphic capabilities of the PET to provide students with mathematics laboratory-type experience.

The first program, POINTS, is self-paced. It allows students to explore the coordinate plane and teach themselves how to locate points in the plane. The second program, HI-CALC, uses a game format, permitting students to manipulate the coefficients of algebraic expressions while attempting to maximize the value of the expression. TICTACPET also uses a game format, whereby students capture boxes on a normal tic-tac-toe board by solving algebraic equations. Students can choose equations from ten levels of difficulty, ranging from the simple to the complex.

THE PET VISITS HUNTINGTON

Abstract

The Huntington Two simulations (along with most of the courseware which exists currently) were designed to run on computers with teletypewriters as I/O devices. This constrained the program designers seriously, permitting only rudimentary graphic output and keyboard input.

The new personal computers have given program designers much more flexibility and have significantly increased the pedagogical impact of these programs.

Two Huntington Two programs (POP and POLUT) have been modified to take advantage of the PET graphics, and one (CHARGE) has been modified to utilize an analog-to-digital converter designed at our laboratory at Stony Brook. These programs are described and demonstrated.

THE PET TEACHES ENGLISH

Abstract

There is an uphill fight to implement usage of computers in English classrooms. English teachers are not convinced that the computer can do anything for them. Furthermore, there would be trouble wresting computer time from the mathematics department.

We offer three programs which in informal studies have been well-accepted by English teachers and, more importantly, by their students. SPELL is a game format vocabulary and spelling program. PUNC provides practice in comma usage. HYPHEN instructs students in the pesty problem of syllabication.

CAI may relieve teachers and students of some of the tedious, time-consuming, and often ineffectual teaching of language fundamentals.

BOULDER COLORADO'S COMMUNITY COMPUTER

Stephan K. Elliott
Project Coordinator
Community Free School
P.O. Box 1724
Boulder, Colorado 80306
(303) 447-8733

The Community Free School in Boulder, Colorado offers cheap and free computer services to the community on a time-sharing S-100 Bus system - and its paying for itself.

About two years ago, the Free School Board of Directors decided to study the idea of having a computer to help with the administrative duties of the school, and to be used as a teaching tool. We talked to too many salesmen, got an armload of help from Lawrence Hall of Science in Berkeley, fast talked some local businesses and individuals out of $2,000 in donations for a downpayment, talked even faster to the local bank, and presto! in November of 1977, our Alpha Micro AM-100 system arrived from the local Byte Shop.

A year later, we're barely able to sneak in a few minutes for the school's administrative jobs since the students have taken over.

The Free School is a non-profit, independant school, offering 2100 courses per year to over 21,000 students. Last year, over 300 students enrolled in 40 computer based classes. The Free School's computer program also includes offering the computer as a free service at open houses, including this last August a month-long stay at the Boulder Public Library.

We offer classes for adults in LISP, BASIC, ADVANCED BASIC, PASCAL, and a survey of computers class called COMPUTER LIB. Each class costs about $50 and includes seven - two hour classes and twelve hours of practice time on the computer. The kid's classes cost $20 and include a book in addition to the practice time. Together the tuitions are making the monthly payments to the bank.

The system consists of an Alpha Micro AM-100 CPU, 64k of ransom access memory, four Persci floppy drives, four CRT terminals, two hard copy terminals, and an auto-dial, auto-answer modem.

The AM-100 is a 16-bit processor that fits the S-100 Bus, and is based on the LSI-11 chip. It timeshares to the six ports via an operating system that partitions memory to users. The operating system allows the use of reentrant higher level languages which means that BASIC, for instance, needs only to be loaded once into memory for several users.

In fact, the operating system of the Alpha Micro is where the system really shines. It features passwords and individually assignable accounts, friendly error messages, a disk filing system that's easy on the user, a handy text editor and formatter, and a BASIC compiler that won't quit.

BASIC programs are loaded in the usual interactive mode, with instant error message feedback, but when programs are run, they are first compiled, and then run. The BASIC also allows CHAINing to other porgrams, calls to assembly language subroutines, and calls to system level commands (the real time clock can be accessed from BASIC, for instance). Subroutines in BASIC can be called by name, variables can be given labels of any length, and the compiled programs run fast. We run all of our CRTs at 1200 baud, and all of the terminals can run jobs with constant printout without the users even noticing.

Alpha Micro supplies us with regular software updates, and one of the recent acquisitions is PASCAL. We started our first PASCAL class this September and the devotees are beginning to come out of the woodwork. There's already enough interest to have an advanced class scheduled for the January term.

The Free School's computer staff and teachers are an energetic crew of professionals (and a couple of grad students from the University of Colorado) who are interested in providing casual, friendly classes for beginners, and more serious help for students with projects being developed and debugged. Projects seem to come together easily in the computer room, with lots of back seat drivers and over the shoulder debugging.

It seems that a big hurdle for the novice is the act of sitting down at a keyboard for the first time and actually typing in something. The Free School system has been geared to make that first experience a fun one, and one which will invite the user to interact and explore. We have a set of introductory programs that include games, a demonstration of how a BASIC program is written, an easy cross referencing system, and several other programs that we hope will lure the beginner into discovering just what computers are all about.

We have been accused of luring people into computers. During the month of August, we didn't have any classes scheduled, so we carried the computer over to the Boulder Public Library where it was available free to anyone who wanted to use it. We had staff members on hand to get people started and to answer questions, and over 350 people came by. Many touched their first computer, many were pleasantly surprised, and some were convinced that we were charlatans.

The thing that is exciting is that people around Boulder are getting exposed to computers, and are beginning to think that: 1. computers are just a tool like any other tool, in that they will work for you once you learn to operate them, and 2. computers are not just for the giant banks, corporations, and government agencies, but are now within reach of many small businesses, schools, libraries, and even homes. More than once, a new user has commented, "I've heard about these things, but I didn't think that they were like this." For the layman, reading about a new technology in Time magazine is one thing, but being able to sit down and really use a computer is a reality that will change the way that person thinks about the world around them.

What's next? Well, we're interested in getting a computing center developed that would be accessable by people all over town via coin operated terminals in city recreation centers, schools, libraries, and shopping centers. A coin in the slot would give you access to upcoming events, medical information, information about government agencies, jobs, bartered goods, or people with interests similar to yours. A public school is investigating using our computer via the phone line to teach classes to fifth and sixth graders, and a personal computing network is underway which will eventually interface to our community computer to allow members to share programs and an on-line newsletter.

We're excited about the capabilities of our system now, but we think that the idea of community computing is just in its infancy and has an even more exciting future. We are interested in sharing our ideas, and hearing what other communities are doing with computer based systems available to the public.

COMPUTER SIMULATION IN THE COLLEGE CLASSROOM:

IMPLEMENTATION AND EVALUATION

Dr. Gene D. Steinhauer
Department of Psychology
California State University
Fresno, CA 93740

The use of computer simulations as a heuristic device in teaching undergraduate courses in Research Methods in Psychology appears to be gaining in popularity. Most individuals who have introduced the technique in the classroom have been very cautious in explicating the difficulties of doing good evaluative research into the efficacy of computer simulation as a teaching aid. Many such projects have been elaborate and incorporate global goals which do attenuate the feasibility of good evaluative research. The use of computer simulation techniques, however, is not necessarily restricted to the kind of model building techniques and more global understanding of the process of scientific inquiry described by Main (5), and Gregory (1). Hartley, Fisher, and Hartley (3) have reported that the use of specific simulations designed to deal with such problems as the strengths and weaknesses of within and between subjects designs resulted in more correct responses on a final examination question which assessed the ability of the students to identify the best design for a research problem. I have recently developed and incorporated into our library of computer programs a few simulations which are intended to illustrate, by example, the concept of an interaction of two independent variables in determining the value of a dependent variable. Since the goals of using these two programs were directed at imparting understanding of a single concept, the opportunity for evaluating the efficacy of the simulations was more obvious and straightforward.

One of the programs simulates a study reported by Grice and Hunter (2). This program, when first called up, outputs a brief description of the question of whether or not stimulus intensity effects classical conditioning of the human eye blink response. The experimental procedure is then described including details of manipulation of the independent variable and measurement of the dependent variable. The computer then asks the student if he/she wants to use a within subjects or between subjects design for studying the relationship between the two variables. The data set which is subsequently outputed is unique for each run of the program. After the data are printed out the student is given instructions by the computer for preparing a graph and instructions for calling up the appropriate statistical program for analyzing the data. If the student choses a within subjects design the data show that as conditioned stimulus intensity increases the percentage of trials with a conditioned response increases. The output for the between subjects option does not show any relationship between conditioned stimulus intensity and percentage of trials with a conditioned response. After the various results are discussed in class (where, hopefully, a student will vocalize the hypothesis that experimental design interacts with the intensity variable) students return to the computer to run a subroutine of the program in which experimental design is a second independent variable. This simulation, of course, yields an interaction of design and conditioned stimulus intensity.

A second program simulates a study of the relationship between anxiety scores and scores on an examination in a research methods course. This simulation also includes an appropriate opening description, instructions for graph preparation and instructions for calling up a data analysis program. In this case the students are asked to make decisions with regard to sampling on three subject variables - sex, age, and IQ. Students can chose to sample randomly with respect to each, or have sex held constant by chosing only one of two categories, age by chosing only one of three categories, or IQ held constant by chosing only one of three categories. Age and sex have no effect on the data generating subroutines. Sampling randomly with respect to IQ yields a relationship such that test scores increase as anxiety increases. Using low IQ subjects

yields no relationship between anxiety and test scores. For high IQ subjects test scores decrease as anxiety increases, while medium IQ subjects show a relationship of the same form, but with a slower rate of acceleration, as the one obtained when sampling randomly with respect to IQ. The data generating subroutines, which give unique data sets for each run, are written so that a weighted combination of the means for the three levels of IQ (1 low : 10 medium : 1 high) yields approximately the same means that are obtained when sampling randomly with respect to IQ. Each student brings his/her output and write up to class for discussion. During the subsequent discussion the discrepant results are discovered and outlined on the blackboard for all to see. Hypothesis are then discussed and students return to the computer to test their hypotheses. The challenge, of course, is for the students to unravel and come to understand the interaction between IQ and anxiety in determining test scores.

A course in research methods is a requirement for all of the undergraduate Psychology majors at SUNY, Oswego (where the following study took place), and is sequenced after elementary statistics and before any core courses in the Experimental Psychology area. There were seven sections of the Research Methods in Psychology course offered in the Spring Semester, 1978. Class size was limited to 24 students due to limited laboratory facilities. Two faculty members, teaching a total of three of the sections, volunteered to use the simulations as instructional aids when covering the topic of interaction effects. Three faculty members, teaching four sections of the research methods course agreed not to use the simulations during the Spring Semester, 1978. These faculty were assured that a package of simulations and a manual for their use would be available for all faculty in the Fall Semester, 1978. All of the faculty involved normally covered interaction effects in their course and gave a final examination question which was designed to assess the extent to which students understood the concept of an interaction of variables. For the present evaluative purpose all sections of the research methods course had a common final examination question designed to assess the student's understanding of interaction. The information given in the examination problem was the cell means for a 2 X 2 factorial design experiment. Students were required to identify which cells to compare to test for main effects for each of the two variables, draw a graph illustrating the results, and provide a written interpretation of the results. A faculty member who was not teaching the research methods course in the Spring Semester prepared a scoring key to be used for purposes of this study. A perfect response was given seven points. Half of the responses for each class were scored by one graduate student and the second half by a second graduate student. These two assistants did not know which of the classes the data they scored was from.

The mean scores for each of the classes was 3.86, 4.29, and 4.00 for the three classes using the simulations and 2.25, 2.97, 4.55, and 2.96 for the four classes not using the simulations. An analysis of variance (unweighted means solutions for unequal sample sizes, Keppel, 4) was conducted with use of simulations or no use as a between subjects factor, classes treated as nested within the between subjects factor and subjects as nested within levels of the simulations and classes factors. This analysis showed a main effect of the use of simulations, $F(1,118) = 14.95$ (the probability of the difference between using simulations and not using simulations being due to chance is less than five in one hundred) and the differences between the classes was on the average so small as to be very likely due to chance alone ($F; 6,118;$ less than one).

The results of the present study suggest that the use of the two computer simulations did help students in giving correct responses to examination questions concerning factorial experiments containing interaction effects. The overall mean score for students using the computer simulations was 4.05, and the mean score for students not using the simulations was 3.18.

We have recently added to our library simulations which are designed to illustrate symmetrical and asymmetrical transfer effects in within subjects designs. In these programs the student is given a description of a problem in a clinical research setting and informed he/she is to serve as a research consultant in solving the problem. The program recognizes

key words which are appropriate or inappropriate in solving the problem. This program allows the student substantial input into the experimental design and provides some students with data showing symmetrical transfer under some conditions while other outputs show asymmetrical transfer under different conditions. The more subjective feedback we have received from students has been overwhelmingly positive when using any of our simulations. The present project was conducted using a PDP 11/34 system shared with the mathematics department. The currently available simulations substantially enhance the variety and levels of projects which we can conduct in the laboratory portion of our research methods course.

All of our simulations have now been adapted to run on a microprocessor. Our success with the present project has substantially enhanced our enthusiasm for the use of computers in many of our experimental psychology courses. We are currently planning to assemble an experimental psychology laboratory with twelve microprocessors and three terminals for the minicomputer. The microprocessors will be used to conduct actual demonstration experiments in human learning without the timesharing problems inherent in the minicomputer system. The various simulations can be run on either the microprocessors or the minicomputer. Data analysis will take place on the minicomputer. The point we find particularly impressive about computer applications in our situation is that we can expose our students to a wider variety of psychological phenomena and methods at a lower cost than purchasing the traditional equipment used in those demonstrations. Microprocessors are certainly well within the budgets limitations of academic institutions and can probably enhance the quality and effectiveness of the educational process.

As the sophistication of psychological research progresses and the budgets for equiping undergraduate psychology laboratories decline it becomes increasingly more difficult to expose students to research areas where interaction effects must be understood. The present study provides evidence that the computing tools frequently available to teachers can be used in ways that are cost effective and can take up part of the gap left by shrinking funds. While computer simulation can hardly be expected to function as a complete substitute for hands on laboratory experiences, simulations can enhance the variety and sophistication of topics which students can come to understand in their studies.

References

1. Gregory, R. J. Introduction to computer data generators. Teaching of Psychology, 1977, 4, 63-67.
2. Grice, G. H. and Hunter, J. J. Stimulus intensity effects depend upon the type of experimental design. Psychological Review, 1964, 71, 247-256.
3. Hartley, A. A., Fisher, L. A., and Hartley, J. T. Teaching the arts of psychological research. Teaching of Psychology, 1977, 4, 202-204.
4. Keppel, G. Design and analysis. Englewood Cliffs: Prentice-Hall, 1973.
5. Main, D. B. EXPER SIM: Experimental simulation. Creative Computing, 1975, 3, 43-46.

Notes

1. The work reported in this paper was supported by a grant from the National Science Foundation, grant number NSF 228-0174A.

2. Copies of this paper, listings of our simulation programs (Basic Plus, and Northstar Basic), and a manual for the simulations are available from the author at no cost.

COMPUTER ASSISTED SELF-EVALUATION AT THE UNIVERSITY OF CALIFORNIA - DAVIS

Eli Cohen and Kathleen M. Fisher
Teaching Resources Center
University of California - Davis
Davis, CA 95616

I. Introduction

This paper describes the implementation of a system for computer-assisted individualized testing in large science classes at the University of California - Davis. The system is designed to bring the advantages of frequent testing with immediate feedback to classes with large enrollments, while using a minimum of hardware. The programs run in Pascal on an LSI-11 based microcomputer.

II. Background of Frequent Testing with Immediate Feedback (FTIF)

The advantages of frequent testing with immediate feedback (FTIF) for school learning have been demonstrated as early as 1919 in the Winnetka (Illinois) Plan (1). The concept was refined by B. F. Skinner, and incorporated in his programmed learning schemes (2).

College level instruction has also benefited from the use of FTIF. Such individualized instructional plans as Postelthwait's audio autotutorial system, and Keller's Personalized System of Instruction have been shown to be effective means of teaching (3, 4).

The University of California - Davis has demonstrated in an earlier project the feasibility and benefit of computer assisted FTIF for large college courses (5). Such frequent testing not only helps students to improve study habits, but also brings the student and teaching assistant in one-to-one contact at those times which are most opportune for learning (i.e., when the student has questions concerning the subject matter).

III. Computer Assisted Test Construction (CATC)

Programs to manage testing were developed for university classes with large enrollments. In 1972, only about 20 institutions constructed tests with the assistance of a computer (6). In the last six years the interest has blossomed. The Educational Testing Service currently sponsors a newsletter dedicated solely to computer assisted test construction (CATC Digest) (7). Many computer programs now exist to generate tests, score tests, or manage the grading of students, although few integrate these functions.

Until recently, CATC (and grade management) efforts have been limited by their dependence on the availability of large computers. A typical item bank for one course may contain some 1000 items (including question specifications) (8). Conservatively, one might estimate the necessity of 1 Megabyte of memory to accomodate the data banks for *each* course using CATC. Only in the past five years has the industry matured to a state where a small machine can physically handle such a large data base at an acceptable level of performance and price. Some of the most recent technological advances which provide such large capabilities for a microcomputer include the advent of Winchester disk technology, and the availability of relatively inexpensive 16 bit CPU's and LSI memory.

IV. Comprehensive Assistance to Undergraduate Science Education (CAUSE)

Funding by the National Science Foundation of a CAUSE grant to the University of California - Davis campus (Professor Kathleen M. Fisher, Principal Investigator) allows the expansion, development, and refinement of existing efforts to automate testing and self-evaluation.

The grant calls for the development and evaluation of a system to generate and score tests, and to maintain student records for six different science courses.

Unlike other computer based learning efforts, the emphasis here is on *economical* use of a small computer. Students receive computer *printed* tests as opposed to *on-line* tests. They solve their problems at a study table instead of at a terminal. The configuration is limited to a printer and a console under a single task environment, instead of multiple terminals under a time-share environment.

Such a configuration allows a relatively inexpensive system to serve many students, but it is limited in that it cannot handle tailored testing or concealed choice testing (9,10).

A typical way in which the system might be used is described below:

1. The instructor enters test items into the

item bank with the help of an interactive data entry program with graphic and text editors. The items are keyed on various parameters, such as their content, cognitive level, and type of item (true/false, matching, fact recall, problem-solving). Other parameters such as item difficulty, discrimination, date of last usage are entered into the item bank automatically by the programs.

2. The instructor enters the names of students and relevant information about them, such as biographical and personal information, into the student information bank.

3. The instructor selects course options, such as how many times students in his class can repeat a test, time lag required between repeat tests and by what method students are to be graded.

4. The instructor selects the specifications of a test, including its title, number of items, and how the items are to be selected.

5. The student enters the testing room, and, types on the console his/her name, course name, and receives a unique individualized test from the printer.

6. The student marks his/her answer on an optically readable card, and enters his/her answers via the reader.

7. The student receives feedback from the printer, including the score and helpful hints on each problem (s)he got wrong. The student is encouraged to discuss his/her problems with the teaching assistant stationed in the testing room.

8. The instructor reviews the progress of the students, both individually and as a class. Individuals who are behind, doing poorly, or overworking can be contacted. For example, the data bank can be queried for the telephone numbers of those students who are two or more weeks behind in passing quizzes.

9. The instructor posts progress reports in various forms (histograms, clustered, listings).

10. Instructor reviews the item statistics to maintain and improve the quality of the item bank. For example, the instructor can query the item bank for the numbers of those items which have poor item difficulty and discrimination.

Instructors may use the system in various ways. Quizzes, midterms, finals, or homework may be generated from the item bank. Tests can be administered individually or in group settings. Grading can be normative, criterion-referenced, mastery or may not be imposed at all, as for practice quizzes.

V. Software Considerations

To borrow a phrase from Hewlett-Packard advertisements, our software should be modular in design, integrated in concept. It must be modular to adapt to the varying requirements of different instructors.

Modularity also will make the system more transportable (within the University of California system) and exportable (outside University of California). Further, modular systems are relatively easier to design and debug.

The software should be integrated in concept. This means that all the programs should work together, with no overlap in routines, and no added expenses due to unnecessary translations.

All of the system's programs can be conceived as sharing four data bases:

1. An item bank, containing all items, item specifications, and item statistics and histories.

2. A test bank, containing information about all existing tests, such as test names, and test item generation specifications.

3. Course option bank, containing information particular to a specific course/instructor, such as grading options, instructor name and the limits imposed on students' generation of tests.

4. Student information bank, containing information relevant for identifying and contacting students and (temporarily) which items were assigned to each student, and what his/her responses was for each item.

We hope to maintain compatibility with some other computer-assisted testing projects. Most influencial in our design considerations are the CAUSE supported projects at the University of California - Irvine (Alfred Bork, Principal Investigator) and University of Utah (Richard Brandt, Principal Investigator) (11, 12), and the UCD School of Medicine testing program (13).

The University of California - San Diego version of the Pascal language has been selected for software development. The data structure capabilities of Pascal equal those of Cobol. The language is very structured and modular, more so than Algol. Also Pascal is nearly machine independent.

There are other software considerations for the system. One is the need for graphics capability and a graphics editor in science instruction particularly, test items often have accompanying graphs and pictures.

The system will be constructed to accept numeric as well as multiple-choice answers. The input of numerics on optically readable cards presents a challenge. We are currently experimenting with a system developed by Professor Douglas McColm (14).

VI. Hardware Considerations

We have selected a stand-along, dedicated microcomputer, with disk memory, graphics printer, and console. Specifically, it is based on the LSI-11 CPU, with 10 megabyte hard disk and dual floppies, Printronics 300 printer, DecWriter II console and Hazeltine Mod I console on the development system.

VII. Future Augmentations

Eventually we may add interactive capabilities for certain types of problems. An intermediate arrangement might be established where keyboard rather than card input is used for item response entry.

Acknowledgements

The system described above is the result of the work of many individuals. They are as follows: Diana Barnes, Kathleen Fisher, Otto Helweg, John Horowitz, Hartley Jensen, Doug McColm, Vernon Singleton, Peter Smietana, Richard Snow.

List of Footnotes

1. Washburne, C.W. and S.P. Marland, Jr., Winnetka, Prentice-Hall, Englewood Cliffs, N.J., 1963.

2. Skinner, B.F. Teaching machines. Science. 1958, 128, 969-977.

3. Postlethwait, S.W. A systems approach to botany. Audiovisual Instruction. 1963, 8, 243-244.

4. Keller, F.S. "Good-bye, teacher...," Journal of Applied Behavior Analysis. 1968, 1, 79-89.

5. Fisher, K.M., Smietana, P., Jensen, H., and Williams, S. Reinforcing the lecture: A simple computer method for frequent, individualized testing in large classes. Working paper, Teaching Resources Center, U. of California - Davis, CA 95616.

6. Lippey, G. (ed) Computer-Assisted Test Construction, Englewood Cliffs, N.J., Educational Technology Publication, 1974. (pg. 16).

7. CATC Digest, Educational Testing Service, Princeton, N.J. 08541.

8. Lippey, G. (ed) op. cit. (p. 16).

9. Wood, R. Response-contingent testing. Review of Educational Research. 43 (4),

10. Concealed Choice testing was developed for Computer Assisted Testing by Stephen D. Franklin, Office of the Director of Computing, University of California - Irvine, CA 92717.

11. Contact Alfred Bork, Physics Department, University of California - Irvine, CA 92717.

12. Contact Richard Brandt, Physics Department, University of Utah, Salt Lake City, Utah, 84112.

13. Contact Richard Walters, Department of Community Health, University of California - Davis, CA 95616.

14. Contact Douglas McColm, Department of Physics, University of California - Davis, CA 95616.

A COMPREHENSIVE PUPIL PERSONNEL ACCOUNTING SYSTEM UTILIZING MICRO COMPUTER SYSTEMS

Melvin L. Zeddies, Ph.D.
Research Consultant/Teacher
1854 Pacific Beach Drive
San Diego, CA 92109

ABSTRACT

Most schools are not taking advantage of the capability of the micro computer system to make the normal school routines more efficient, and humanizing. Time, accuracy, and overall efficiency are improved through the use of small computerized accounting systems. Enlarging the configuration to several schools allows the efficiency to become even more noticable and beneficial. This paper describes one such pupil personnel accounting system, with suggestions for adaptation to an entire district.

TEXT

The broad impact of personal computer systems is finally being realized. The glamour surrounding them is wearing off and serious applications for their use are now being developed in almost all areas of human activity. The field of education is one that lends itself to a wide variety of these micro system applications. To date, most of these applications have centered around the instructional area; however the areas of counseling, administration, and combinations of areas could and should also be deriving a great deal of benefit from the use of these systems.

One of the most common complaints heard from counselors is "I have too much paper work to see many of the kids." The lament is indicative of a program that is failing in its purpose because it is not using the available technology. Rather than being bogged down in paper work the counselor must be freed to fill his/her essential purpose of direct personal, concerned contact with students.

The best source of hope to provide the time needed for the counselor to perform his/her real function may be found in the use of personal computer systems, and their application to the pupil personnel program in today's schools. The need for some relief from the paper work demands made on the pupil personnel services department is well known. The advent of low cost, personalized computer systems seems to be the perfect solution. With that in mind a system was designed which would provide a wide variety of services, streamlining and even eliminating many routine clerical activities.

In developing the particular approach described in this paper, certain considerations were taken into account. The school environment as a whole was considered, as was the relationship of the pupil personnel services department to the school. The size, needs and experience of the counseling and guidance staff were evaluated. The author's past experience as part of the counseling and guidance staff was of real value in assessing the needs of the school. These considerations shaped the direction of the program, and would be important in its implementation.

Any real change must be developed in a non-threatening manner, with the change agent becoming part of the "establishment." The use of a systems approach in the construction of the design was a necessity for it proved to be an effective way of dealing with the many factors involved.

The system described here was developed to meet the needs of the pupil personnel department on several levels and with a variety of programs. One of the most useful and yet most basic programs is the one designed to handle the daily school attendance (see Computerized Attendance diagram). The program provides teachers, counselors, the attendance office and the administration with a daily absence list; weekly, and quarterly absence lists are also available for the attendance office, counselors, and support personnel such as the district counselors, as well as the administrators.

A second program provides a variety of information concerning each individual student (see list of Reprots Generated). Personal information, records of all classes taken with grades received, test scores, and career and vocational information are also included. Present class schedules, also attendance, fines as well as library books. This information is immediately available to the counselor, thus saving a great deal of time and effort.

Within the data base (see Data Base Configuration diagram) there is a great deal of available information about all the students. Thus it is not only possible to retrieve detailed information about the specific student, but it is also possible to gain information about specific groups of students, such as attendance of 11th grade males. This information may be of value to teachers, counselors, the school psychologist, or the school administrators.

It is easy to see that the vast amount of information that by being available provides enormous savings in time and effort. It simplifies many tasks and reduces by almost 20% the amount of paper used by pupil personnel departments. With the cost of the computer equipment coming down and the cost of paper going up, a point will be reached where it is not only easier but also cheaper to utilize the computer. That point in time is now. With the adoption of programs like those described here alleviation of many of the difficulties and frustrations experienced in school record keeping are possible. They free the staff for more creative, and meaningful endeavors; leaving to the computer those retetitive tasks which can best be performed by the computer.

The following diagrams serve to clarify the functions which these different programs perform. The actual content of the programs at different stages will be determined in part by staff interaction (see Implementation Sequence diagram). It is important to allow for as much flexibility as possible so that the greatest use may be made of the system. As the staff becomes familiar with the computer and is kept informed they will be more willing to make suggestions and to experiment with additional applications.

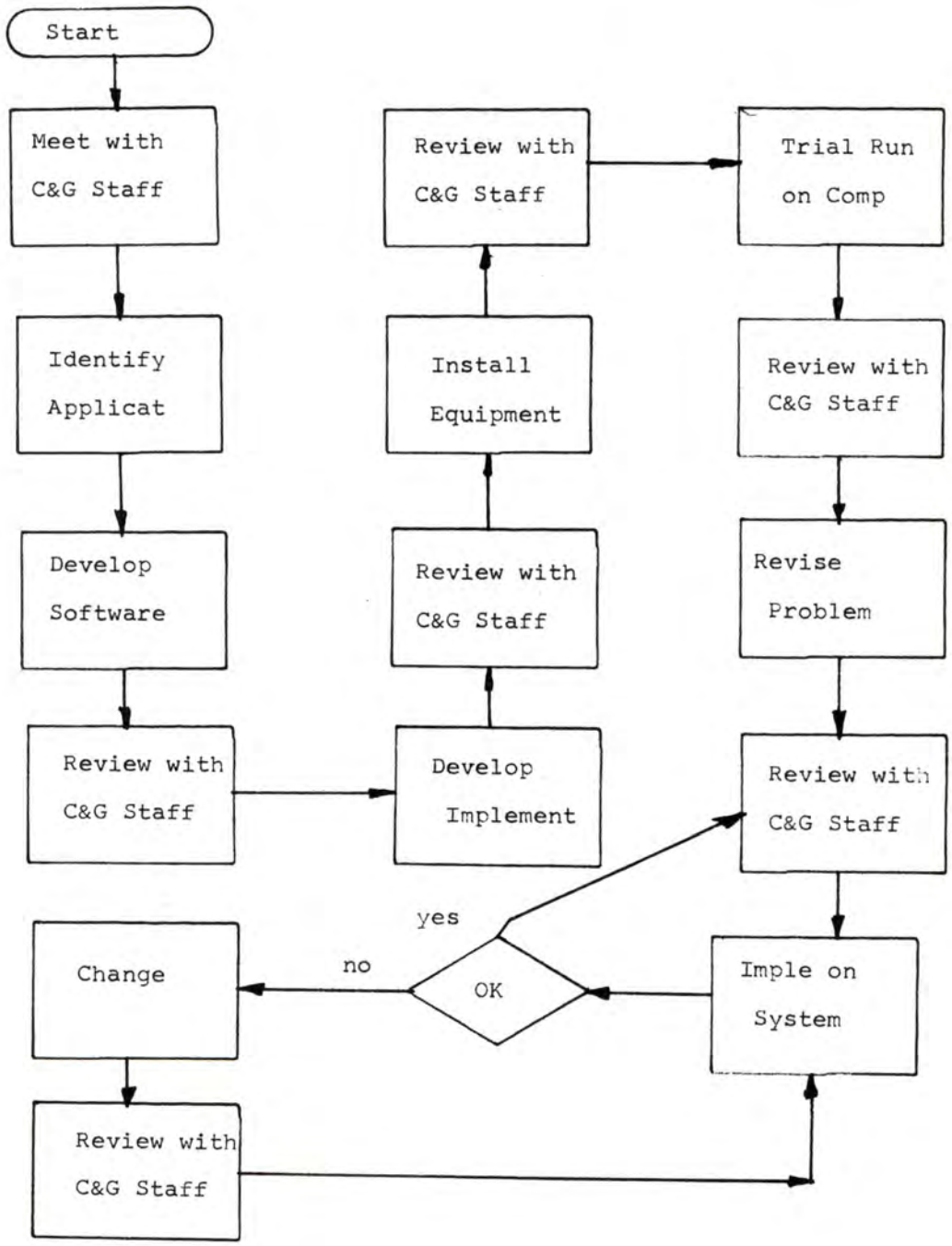
To expand a one school configuration to a multi-school or district wide system, one need only form a small computer network (see Multi-School Network). A very simple quasi-network can be developed with the use of floppy disks that are sent to the requesting party. Any network compounds the security problems, and problems of information leaks. A complete treatment of the security aspect is presented in John Carroll's book, COMPUTER SECURITY.

Field	Contents
1	Student ID (DOB and enrollment number)
2	School number
3	Address
4	Emergency contact person
5	Health record
6	Test records
7	Career information
8	Class schedule (student locator)
9	9th grade transcript
10	10th grade transcript
11	11th grade transcript
12	12th grade transcript
13	Financial record
14	Attendance
15	Special information (coded)

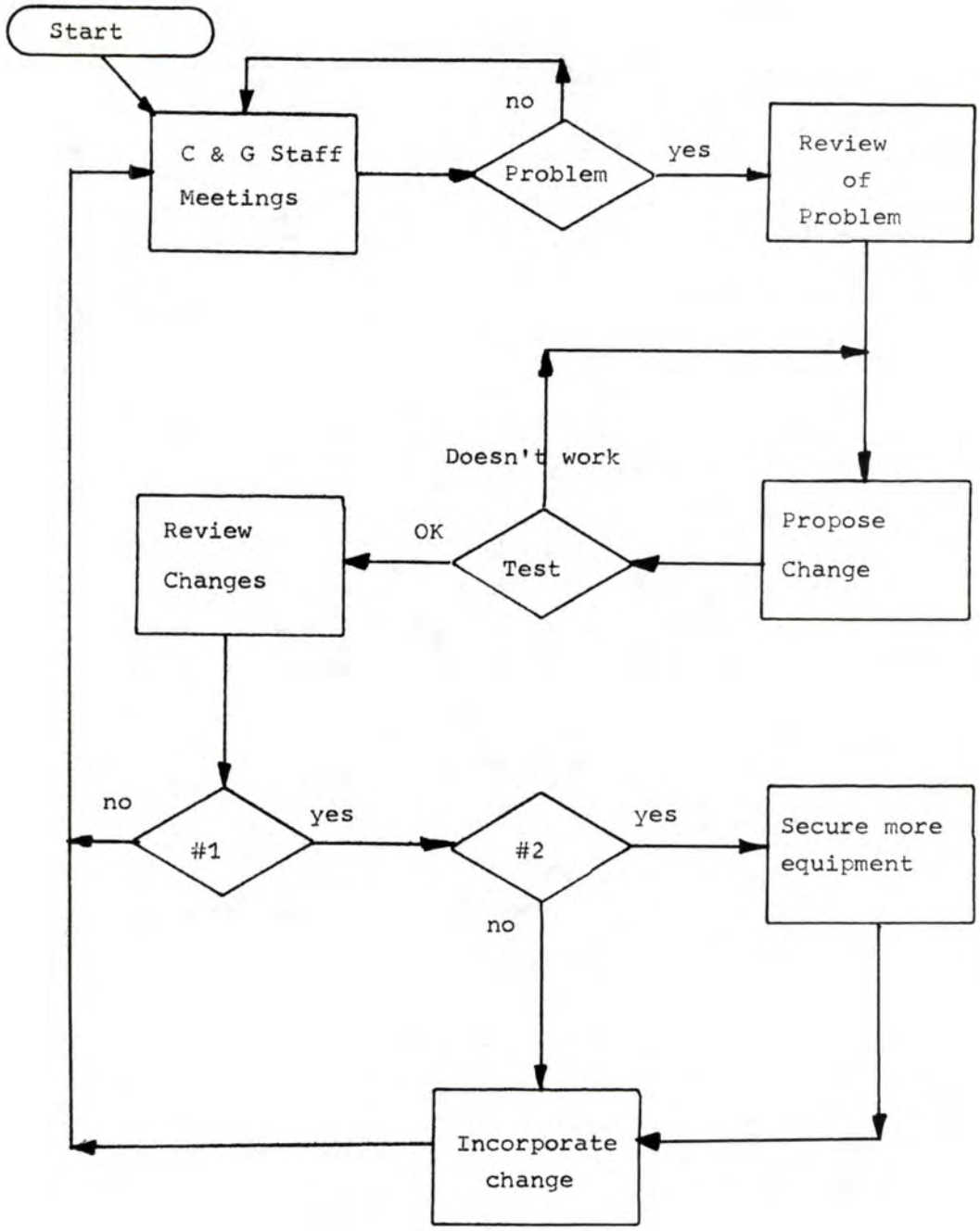
DATA BASE CONFIGURATION

Reports
Student attendance
Career information
Health (special cases)
Financial
Transcripts
Student locator
Other reports are possible depending upon the needs at a particular time and place

REPORTS GENERATED



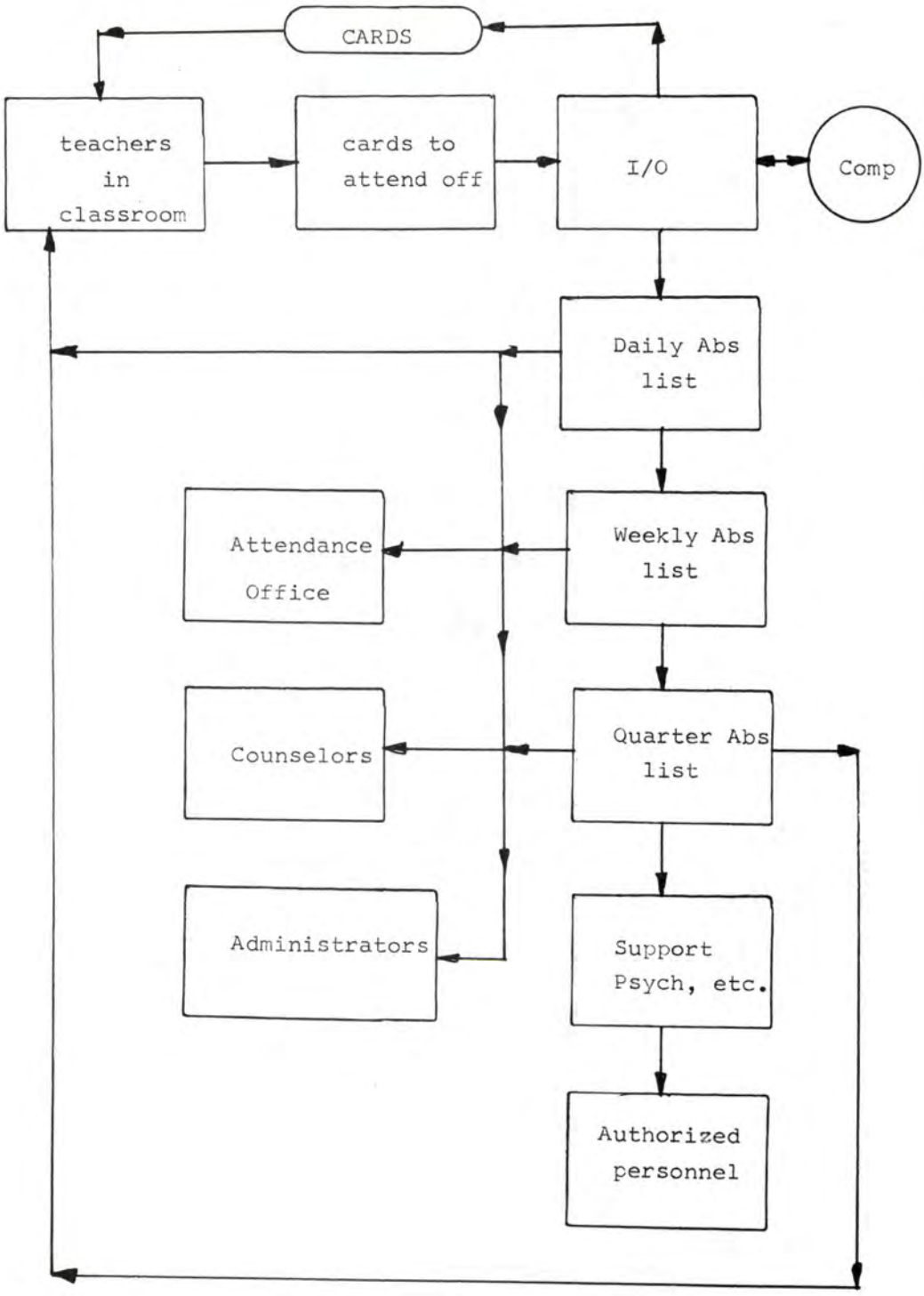
IMPLEMENTATION SEQUENCE



#1 : Implement change?

#2 : Additional equipment needed?

SYSTEM UP-DATING SEQUENCE



COMPUTERIZED ATTENDANCE

indirectly parents and society as a whole. For anything which enables educators to do a better job of educating, helping, and working with individual students will finally be reflected in the way in which these individuals realize their potential and take part in the larger community around them.

SUMMARY

Present day technology provides an excellent opportunity for the incorporation of personalized computer systems into pupil personnel programs. Many of these programs are bogged down in clerical bookkeeping activities, which the computer can and should do. The use of the computer for those tasks frees the counselor and administrator for more human, personal contact with students and provides, as well, for fast, efficient, and accurate information to aid them in their work.

ACKNOLWEDGEMENTS

The following persons contributed to the development of this project in many ways.

Frank B. Thornton, Principal
Patrick Henry High School
San Diego, California

Mrs. Jeanette Kramer
Assistant Principal
Patrick Henry High School
San Diego, California

Leonard King, Head Counselor
(retired)
S.F.B.Morse High School
San Diego, CA

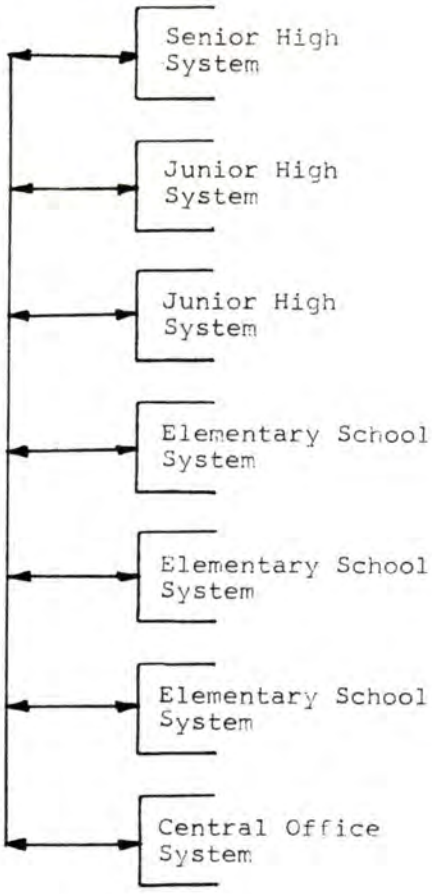
Robert Jenkins
Patrick Henry High School
San Diego, California
(former Attendance Coordinator
S.F.B. Morse High School)

BIBLIOGRAPHY

Carroll, J.M. COMPUTER SECURITY. Los Angeles: Security World Publishing Co., 1977.

Kaufman, R.A. EDUCATIONAL SYSTEM PLANNING. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1972.

Lyon, J.K. THE DATABASE ADMINISTRATOR. New York: John Wiley & Sons, 1976.



MULTI-SCHOOL NETWORK

LET'S GET SERIOUS ABOUT COMPUTER GAMES

Bob Christiansen
Quality Software
10051 Odessa Avenue
Sepulveda, CA 91343

Abstract

The author believes that the quality of games available for the home computer today is generally poor. In most cases, not enough time and thought is being devoted to the development of computer games. Properly designed, these games can be an excellent form of home entertainment. More effort is needed to produce enjoyable games for home computer owners.

A good computer game doesn't blow up, it uses the computer to enhance the game, and it is easy to play, yet challenging. The author discusses the methods by which these attributes can be built into a computer game.

Let's Get Serious

Computers are fun. They can be made to do all sorts of dandy things like display fancy pictures on a TV and print out clever remarks. They can even be programmed to compete against a human opponent in a contest of logic. Put a couple of these neat tricks together and you have a good computer game--right?

One would think it was as easy as that, based on the large number of game programs that have been written and are being sold for home computers. The first purchasers of personal computers needed something for their new equipment to do, and they were quick to recognize the computer's potential for playing games. I recently conducted an informal survey of three small computer journals to see what types of software are available to the home computer owner today. The results indicate that games are by far the most popular type of program.

Program Type	Count	Pct.
Games	139	40%
Financial Programs	59	17%
Personal, non-game	53	15%
Non-Financial Business	40	12%
Operating Systems, Utilities, Program Languages	32	9%
Text Editors, Word Processors	10	3%
Assemblers	7	2%
Music	4	1%
Education	1	<1%
	345	

It's obvious that there is no problem with quantity when it comes to computer games, but there does seem to be a problem with quality. Yes, most of the games do incorporate a neat trick or two, and they are fun at first, but few are really good enough to make me want to buy them. I have heard people who work full-time in the small computer industry complain that currently available games are generally of low quality, and that a large percentage of them are just "junk". This may seem to be harsh judgment, but these people apply the same standards to a game that they would to a word processor or an accounts receivable program. And rightly so.

Customers, too, are beginning to demand better games. At first nearly everyone was fascinated with simple action games--like Pong. But I think that people are now expecting computer programs to be more sophisticated. Some people are even beginning to criticize or belittle the use of home computers for game playing. They seem to be saying that game playing is frivolous. What we need to change these critics' minds are better designed game programs--good games that people will want to play again and again and again--indeed, games that people will want to pay for!

The general low quality of games may be convincing the potential home computer buyer that a computer is nothing more than an expensive toy. To counteract this thought, one computer system manufacturer proclaims in its ad that its computer can "do more than just play games." As a game afficionado, this language offends me. We shouldn't be downplaying games, we should be upgrading them.

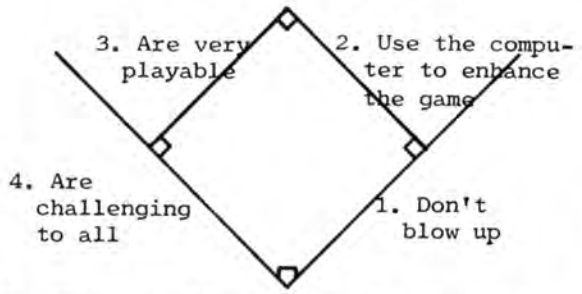
As I see them, the facts are these: first, games have long been an excellent form of home and family entertainment; second, computers can be used to make many games even more enjoyable; and third, computer games will remain a major use of

home computers. The third fact will not be realized, however, unless computer games are recognized as high quality entertainment. So, LET'S GET SERIOUS ABOUT COMPUTER GAMES!

Four Attributes of a Good Computer Game

I will present what I think a good computer game should be like by discussing what I consider to be the four important attributes of a computer game. I like to think of these four attributes in terms of a baseball diamond.

GOOD COMPUTER GAMES . . .



The first attribute, that the software doesn't blow up, is what gets the game to first base. Second base is reached if the computer is used to enhance the game. The game has made it to third base if it is easy to play--that is, interacting with the computer is simple and the rules are not overly complicated. And the circuit is complete if the game is challenging to the seasoned player as well as the beginner.

Games Should Not Blow Up

Those who have done some programming know what blowing up means. A program that has a bug (a programming error) in it will sooner or later cause the computer to do something you didn't want it to do. This is blowing up, and it occurs in all types of computer software. Where games are concerned, blowing up also can mean that the computer fails to enforce the rules of the game. For example, if a chess program allows a rook to move along a diagonal, and this occurs in actual play, then the game has been destroyed. In other words, the software has blown up.

One would hope that computer software being offered for sale would not blow up. Apparently this is not the case--especially when it comes to games. Many for-sale game programs don't even make it to first base. Speaking at the National Computer Conference held in June of 1978, Jef Raskin of Apple Computer related an incident where he was sent some programs that were being sold to run on the Apple Computer. The software company now wanted to sell the programs through Apple, but when Jef tried to run them they blew up. He asked them to bring their own tapes to him, which they did. As it turned out, even these tapes, supposedly the finished product, had bugs in them.

The difficult question is, how can programs be made bug-free? The only answer is careful testing. It is best if the designer/programmer is not the only one to test the program. If you have a friend who doesn't program at all, but loves games, he or she would be an ideal candidate to to test your software simply by playing your game over and over again. This type of testing not only turns up bugs and eliminates blow-ups, allowing the program to get to first base, but it also tests the program's ability to reach the other bases (particularly the ability to get from second to third, as we will see later).

The Power of the Computer

A good computer game must use the power of the computer--that is, it must provide something that a non-computer game can't provide. Remember that if you take an existing game and put it on a computer, you may be eliminating some enjoyable aspects of the game. Game players like to handle the pieces, they like to turn over cards, they like to roll the dice. If the computer takes these things away from the player, then it must make amends by adding something to the game.

There are many ways that the computer can be used to enhance a game, but before discussing them I would like to give an example of a computer game that fails to make it to second base. The Radio Shack people, who have a great product in the TRS-80, currently distribute with their basic system a couple of games, one of which is backgammon. I was interested in this game since I have written a backgammon game called FASTGAMMON that we are now selling for several systems, including the TRS-80. Briefly, Radio Shack's backgammon game is a two-player game that displays a crude version of a backgammon board and keeps track of the men as the players move them around the board. The only way the powerful Z-80 processor is used to enhance the game is to disallow illegal moves. Other than that, this game simply turns the TRS-80 into a rather expensive backgammon board. Based on conversations I have had with Radio Shack owners, I would guess that this program, with thousands of copies in existence, is used very little.

Some ways that the computer's power can be used for games are:

(1) Interaction
(2) Graphics
(3) Data storage/presentation
(4) Number crunching
(5) The computer is the opponent
(6) Randomness and timing

Interaction is probably the most outstanding feature of personal computers, and it can be used to good advantage in computer games, especially one-player games. Star Trek is an example of a game that uses a lot of interaction--the player is required to supply inputs frequently and his selections have a major effect on what happens in the game.

Every computer system on the market today has some video graphics capability, and a good game should make use of this capability. This sometimes requires a great deal of programming effort, but it will be appreciated by the player, who is used to seeing pictures on a video screen. An example of an effective use of video graphics is the game of Life, and I have seen an Othello game on a Compucolor system that made good use of color graphics.

With just the touch of a button, the player can call up on a video screen a great deal of information. This is a distinct advantage of computer games over conventional games. Star Trek is an example of a game that stores a lot of data, and the better versions allow the player easy access to this data.

When a large number of calculations have to be made, computers are at their best. For centuries people have been trying to develop a chess-playing machine, but until the arrival of digital computers their efforts were largely in vain.

The computer also makes a good non-human opponent. It can be programmed to react to the player's inputs and use its own logic to compete against the player. This is what I did with FASTGAMMON, what my partner is doing with his chess game, VIKTOR, and what others have done with Blackjack and many other games.

Finally, randomness and timing can be used to enhance a game. These are not unique to computer games. Conventional games use dice and spinners to determine probabalistic outcomes and clocks for timing. But computers can handle probability functions much more complicated than a simple dice throw, and automatic timing is relatively easy to program into a computer game. A computer football game should put these features to good use.

Make the Game Playable

A game does not have to be simple, but playing the game should be. The popular game of Monopoly, for instance, has many aspects to it and many rules. Yet most anyone can sit down and begin playing Monopoly immediately, learning the rules as he goes along. This is the way it should be with a good computer game, too. Unfortunately, I have loaded in several computer games where you type "RUN" and the only thing that appears on the screen is a question mark. The player has to divine what to do next or search through a list of instructions before he can even get started.

If the rules of a game are complicated, the player should be prompted along the way using alphanumeric prompts. Prompts should be avoided, however, if they only state the obvious and give no really helpful information. An instruction booklet should be included, of course, but its function should be merely to get the player started and for use as a reference or to settle arguments.

Another important way in which games can be made easy to play is to eliminate unnecessary button pushing as much as possible. If the player is faced with one of three choices, he should be able to press 1, 2, or 3 and get an immediate response. He should not have to hit "return" when making a simple menu selection. Also, keys not applicable to the input the player is about to make should be disabled. For example, if he is asked to select either 1, 2, or 3, and he selects 4 or E, nothing should happen, not even an error statement, and especially not an abort.

I have what I consider an excellent method of testing my programs for blow-ups and for unnecessary key pressing. I merely sit my one-year-old son down at the keyboard and let him bang away at it. I try to design the game so that it doesn't hang up even when he "plays" it.

Timing is also important in making a game playable. The pace of the game should be neither so fast that the player can't keep up with the action nor so slow that he gets bored waiting for the computer. The latter requirement, coupled with the lethargic nature of BASIC interpreters, makes it difficult for some BASIC games to reach third base.

In his book, The Home Computer Revolution, Theodor Nelson sums up very nicely what I have been trying to say here. "In this new era, simplicity and clarity and ease of use will become important as never before. . . . There is a myth that things which are simple and clear are not powerful. This is ridiculous: combining simplicity and power is the problem that now confronts us."

Games Should Be Challenging

Given that your game is well-designed and makes it to third base, there is still a final question--will people want to play it again and again? They will if it continues to present them with a challenge. Regardless of the player's ability, he must feel he has both a chance to win and a chance to lose.

I am convinced that the public will begin demanding more sophisticated computer games. They will soon discover they

cannot beat the computer at simple tic-tac-toe. And finding the Wumpus will not remain exciting for very long. People are aware of what a computer can do, and they expect a computer game to present a challenge that will entertain them every time they play it.

There are two popular ways to design a computer game so that it will be fun for players of various skills. The first is to provide some method by which the player(s) can set the desired skill level before the game begins. Some pong-type games, for example, allow the player to set the size of the paddle and/or the speed of the ball. Chess and checker games often allow the player to select the depth of the tree search used by the computer's algorithm. Another way to minimize differences in skill levels is to introduce chance. The roll of the dice or the deal of the cards is sometimes enough of an equalizer, and handicapping becomes unnecessary.

Some Concluding Remarks

Now we have seen what it takes to make a good computer game. How many programs around today touch all the bases? Those who are familiar with computer games will probably agree with my assessment that very few currently available games possess all of the four attributes discussed here. However, there is good reason to expect this situation to change.

An improvement in the quality of computer games is inevitable. The poorly designed games that are now on the market will be discarded and forgotten, replaced by well-designed, entertaining games. And the sooner this happens, the better it will be for everyone concerned. Getting serious about computer games will not only bring more enjoyment to the computer owner, it will make the computer a more respectable household item. Computer games will no longer be considered frivolous, and the image of the home computer as a toy will fade.

SOLVING SOMA & POLYOMINOES PUZZLES BY COMPUTER

David M. Collison, M.A. 2215 W. Broadway, #F226
Anaheim, Ca. 92804. Tel: (714) 956 - 2581

The usual way to solve dissection puzzles is by staring at them or shuffling the pieces until the solution appears. This is a highly intuitive approach. By contrast, a computer tries every case in a systematic way. This sort of program is ideal for a personal computer. It requires little input/output or storage for data -- although the code tends to be bulky -- and a great deal of computation. The hobbyist can leave the machine running for days on end, which is usually impossible at a college or university.

The Soma puzzle (1) has seven pieces which form a cube of side three, while thirteen Polyominoes (2) form a square of side eight. Both use the unit square or cube as the smallest unit of measurement -- important for programming as dissection puzzles are usually irrational (3). These particular puzzles have considerable depth and have not been exhausted yet. To avoid obsolescence, the program accepts input-specified pieces and figures of several colors in two or three dimensions.

The input phase reads in the XYZ coordinates of the cubes forming the figure and the pieces. A previous run finds all the positions of a piece by rotating it in three dimensions, up to 24 for a piece with no symmetry, and stores them in a table. Users with small computers can save space by generating each position as needed, but the runs will be longer. Each position, which consists of a set of coordinates, is sorted in ascending order and there are some error checks.

The Search Phase

We are now ready for the search. The program tries the first available piece in the first vacant cube of the figure and continues until no more pieces will fit. The user can shorten the search time considerably by specifying the ordering of the pieces and the cubes. The most awkward piece should be first and the most restricted cube in the figure. It is easier for the user to determine them than the program.

The tricky part of the search is the backtracking. Usually the program will have some pieces left over which cannot fit into the remaining space. It has to remove the last piece and try the next position. It is fatally easy to forget to reset a marker and I spent most time debugging in this area.

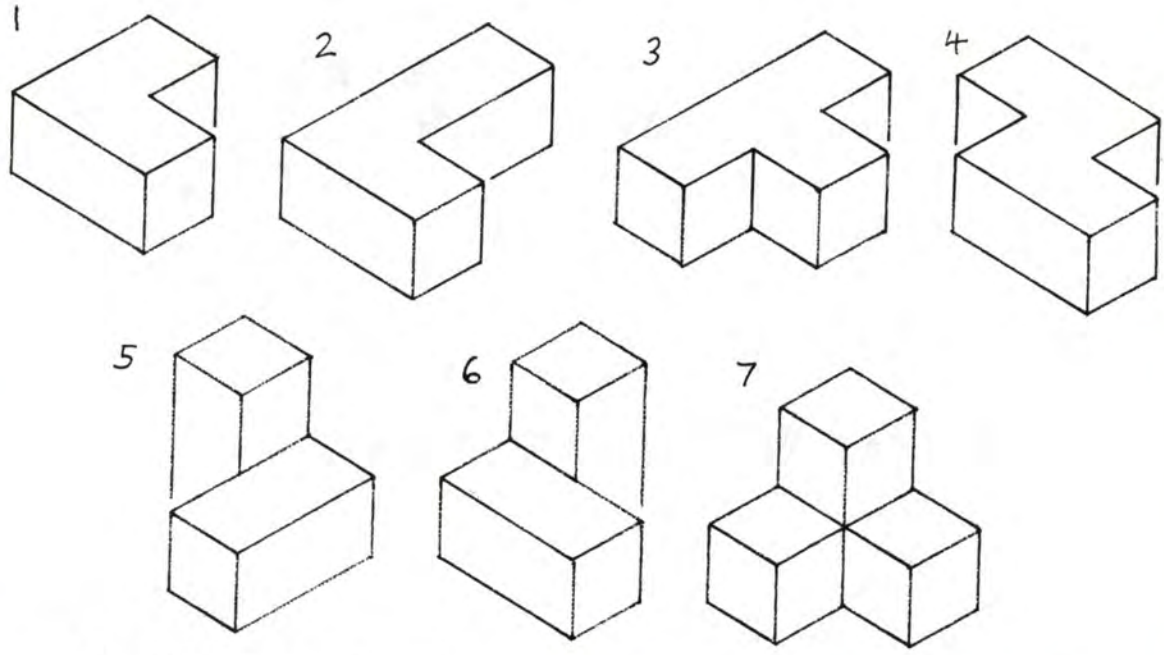
One way to simplify the backtracking is to have a series of nested loops, one for each piece. This is fine for a fixed number of pieces, but I do not know of any compilers which can generate a variable number of loops. Setting the number to the maximum anticpated may exceed the limit for the compiler -- typically this is 8 or 16.

Another way is to have a stack and put all the data for the current piece on the stack -- certain computers can handle this efficiently. Languages which permit recursion, such as ALGOL, do this automatically. The tradeoff is speed -- they are usually slow. The alternative is to write the stacking mechanism in assembly language.

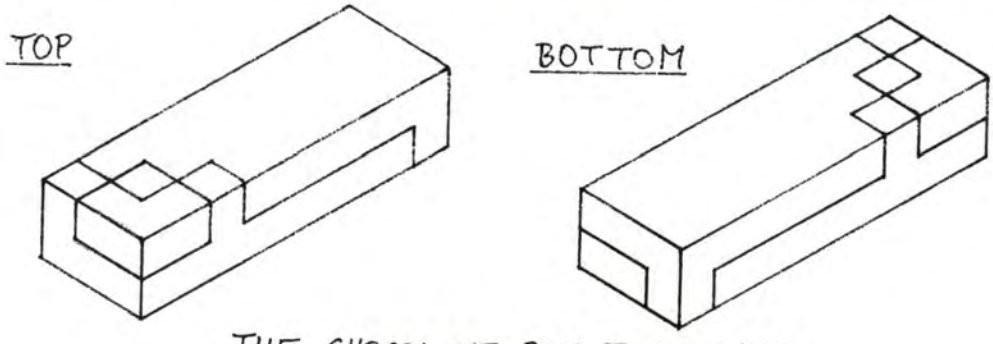
Test Data is Important

Neither way suited the facilities available, so the test data became important. One useful figure is the Chocolate Box, shown on the next page. It uses two differently-colored sets of Soma pieces and is a 3x2x9 block with a diagonal ribbon of the other color across top and bottom -- hence the name. It has a unique solution. This is easy to prove. It may well be the only interesting figure with a unique solution and it makes a good test case.

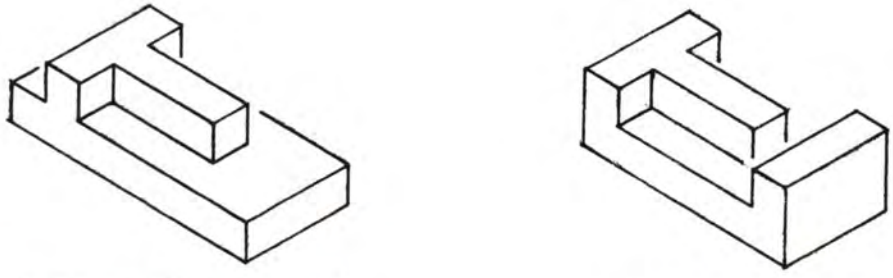
Most figures either have no solution or a large number. Another test case, shown on the following page is the 3x20 rectangle for Polyominoes. This is taken from (2) and has eight solutions including reflections and



THE SOMA PUZZLE. SEVEN PIECES FORM A CUBE OF SIDE 3



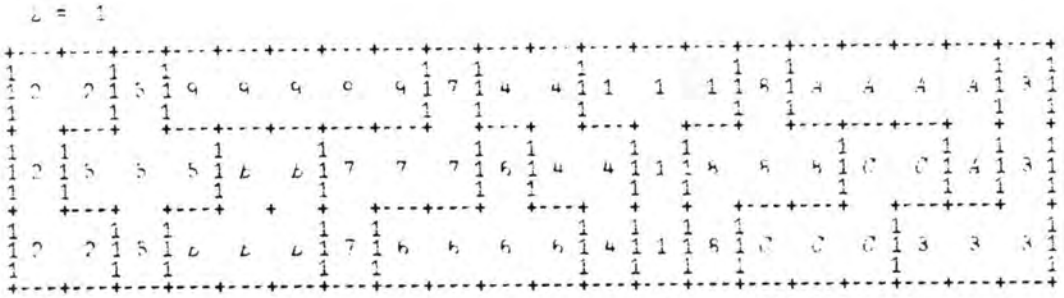
THE CHOCOLATE BOX TEST CASE



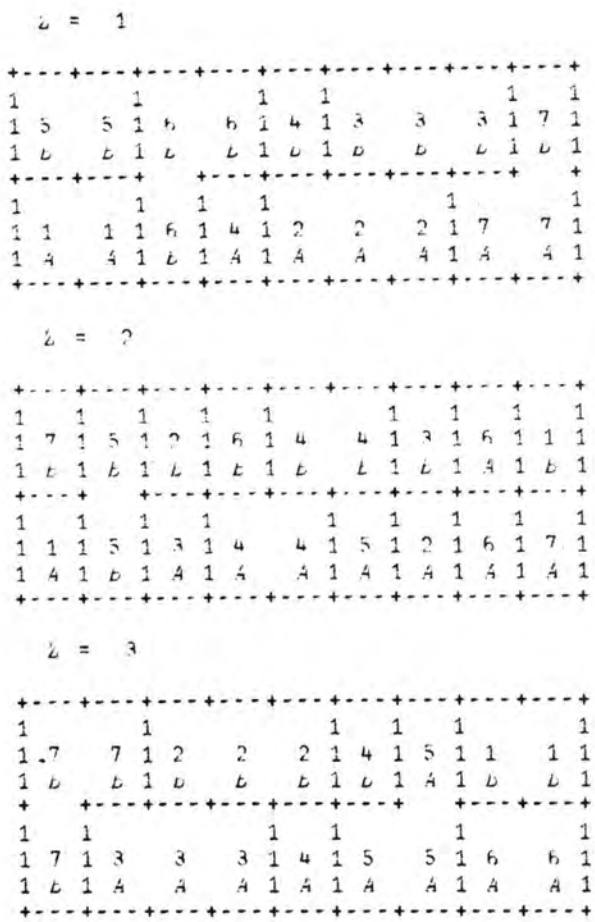
TYPEFACE T (IMPOSSIBLE)

T BAR (POSSIBLE)

TWO SOMA PROBLEMS



Solution to 3x20 Polyominoes
The twelve pieces are numbered 1-9, A-C



Solution to Chocolate Box, Soma Cubes
Pieces are numbered 1-7, colors are A
and B. Sections are vertical by length

rotations. There is something wrong with the backtracking if all the solutions do not appear -- luckily mine worked the first time I tried it.

While they improve the article, it is unnecessary to have the elaborate outputs shown here -- a list of the pieces occupying the cubes is enough for research purposes. Removing the lines between squares or cubes of the same piece is a refinement -- I found the programming tricky. It is easy to adapt this kind of diagram for a CRT display and it only needs the standard character set. An isometric diagram does not show all the cubes and needs a plotter for the diagonal lines.

Impossible Figures

The computer has a clear advantage for proving figures impossible. The hand method uses a decision tree. Start by trying to fit a given piece in a particular position. There are usually several alternatives: the main branches of the tree. Try the next piece for each alternative, and so on. Imitating the computer method is tedious and it takes skill to keep the number of branches manageable. A proof may take several pages (2). Elegant proofs are rare.

There are a number of borderline figures which can be constructed if the pieces are varied. The previous page shows Typeface T, designed by the Rev. V. G. Feser, which is impossible with the standard set. There is no simple proof. Using optimized FORTRAN on an IBM 370/155, the computer took 8½ minutes to check the alternatives. The other design, T Bar, has at least two solutions. The computer took 2½ minutes to find the first -- it is on the next page.

One way to cut the search time is to reduce the number of positions for a given piece. The figure must be symmetric for this. The cube has over a million solutions. Allowing one position for the L-shaped piece, the only one without symmetry, cuts the number to about 50,000.

Another technique is checkerboarding, coloring alternate squares or cubes black and white like a checkerboard. This is a useful hand method and will sometimes prove a figure impossible at once. It is tricky for a computer program -- I did not use it.

There is a challenging problem for people who find the average puzzle too easy. Take two sets of pieces and two impossible figures. Exchange a few pieces between the sets and then try and construct both figures. There are a few solutions, but proving a given pair of figures impossible needs a computer.

Extension to Tangrams

Tangrams (4), the Chinese puzzle, has seven pieces which form a square. The pieces are composed of 1-4 isosceles right-angled triangles and it seems natural to extend the program to handle them. Instead of specifying a square as empty or full, there are another four alternatives for half full. This is no problem. The print routine had to be altered to handle the triangles and this was less successful.

The program does solve Tangram puzzles -- provided that the proportions of the sides are accurately known. Pictures of puzzles do not usually give this (4). Part of the puzzle is guessing the relative lengths and the program is not designed to do this. It does search satisfactorily, but the user has to do most of the work beforehand.

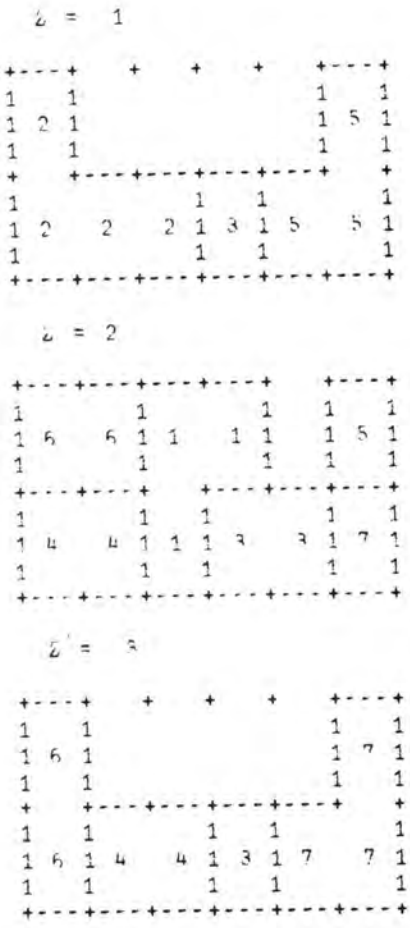
Your Own Program

The constraints on this program are speed and convenience to the other users of a shared computer. It is written only in FORTRAN, no assembly language, to make it compatible with other systems.

For the home computer, extra core is expensive and there are several ways to cut the space requirements. Since many languages have large input/output packages, segmenting the program will help. There are three phases: input and data checking, search for solutions and output. The middle one is the largest and has no I/O.

It is worthwhile to pack the coordinate data -- besides core, it saves two conditional statements in testing. The maximum size of the array for positions is currently 712 words (Soma, two sets). Polyominoes only needs 315 words because it is two-dimensional. Some compilers permit half-word definitions but these do not help much.

I never seriously considered assembly language in spite of the obvious advantages. The FORTRAN linkages are inconvenient and it is tedious to debug core dumps. The home user may find this attractive -- I would prefer to buy more memory.



Solution to T Bar, Soma Cubes
Pieces are numbered 1-7, sections
are vertical by length

Acknowledgments

My hearty thanks go to Mike McNamee, Paul Robinson and the staff of the Computer Center at Orange Coast Community College for their help and forbearance.

References

1. Gardner, Martin. Mathematical Puzzles & Diversions. Chapter 6, The Soma Cube. Simon & Schuster, New York 1961.
2. Golomb, Solomon W. Polyominoes. Scribners, New York 1965.
3. Lindgren, Harry. Recreational Problems in Geometric Dissections. Rev. Greg Frederickson. Dover, New York 1972.
4. Loyd, Sam. The Eighth Book of Tan. Dover, New York 1968.
5. Collison, David M. Rational Geometric Dissections of Convex Figures. Journal of Recreational Mathematics. Scheduled for 1979.

THE RIBICOFF BILL

John S. James, Cetus Corporation, 600 Bancroft Way, Berkeley, Ca. 94710.

The views expressed are the author's and do not necessarily reflect those of Cetus Corporation.

Abstract:

The Ribicoff Bill, now in the U. S. Senate, provides up to 15 years imprisonment for any use "without authorization" of any computer, network, or software operated by any organization affecting interstate commerce. It includes non-profit personal projects by employees, such as unsponsored research or education, club mailing lists, or game playing.

The author draws on his 14 years professional computer programmer and systems analyst experience to discuss the impact of such a law on industry working conditions and the development of professional skills.

Computer people are usually not political, which may be just as well. But today Federal decisions are being made which could utterly change the working atmosphere of one of the nation's most successful industries. We have had no input into these decisions, because we have not known about them.

The prime example is S1766, the Ribicoff Bill against computer crime, co-sponsored by Senators Ribicoff, Dominici, Griffin, Heinz, Jackson, Kennedy, Metcalf, Percy, and Thurmond. Although the current draft has existed unchanged for over a year, very few in the computer industry, even among leaders of professional organizations, have had any idea of what it contains. Perhaps we assumed that a computer crime bill wouldn't affect us if we didn't commit crimes.

The Ribicoff Bill would apply to any computer owned or operated by "any entity operating in or affecting interstate commerce", or connected in any way to such a computer by a network. Essentially it would cover all except personal computers operated stand-alone. Calculators are also covered, and technically the bill's definition of "computer, computer system, and computer network" is phrased broadly enough to cover the telephone, whether used in conjunction with data processing or otherwise.

One section of the bill states that anyone who "intentionally and without authorization" accesses or alters any such computer, or any program or data contained in it, shall be punished by a fine of up to fifty thousand dollars and/or by up to fifteen years imprisonment. "Authorization" is not defined, so "without authorization" is the only definition for this section (1028-b) of what is or is not criminal. However "access" is defined, as

> "to approach, instruct, communicate with, store data in, retrieve data from, or otherwise make use of any resources of, a computer, computer system, or computer network."

The Ribicoff Bill is short and written in non-legal language, so you don't need to be a lawyer to understand it. But you may need to read it three times, as I needed to, because it is full of language about financial fraud and white-collar crime, making the bill appear to be targeted against obvious criminality. But those sections are not the important ones, because they are already well covered by existing law.

Some of the consequences of this are more serious than the unlikely (but possible) case of a computer user prosecuted for playing Adventure on an employer's computer time without advance permission. For the industry depends more than generally realized upon some flexibility of the informal work rules which develop in actual experience, and often embody more wisdom than the official rules of legislation or management. There are large gray

areas of action not officially permitted (which sometimes could not or should not be), and yet are not criminal either, short of gross excess. Some would consider this ambiguity a problem to be eliminated by more formal procedures. We should look carefully first.

A few examples raise questions about how the Ribicoff Bill might be interpreted. Keep in mind that the public has little understanding of computers, some hostility, and recently an image of the clever computer criminal. We cannot be sure what may or may not come out of a court.

(A) Software Modification

Would the law cover contract violations in use of software, elevating civil matters to major felonies? It apparently would cover unlicensed use of software, but what about minor violations of terms? If an installation purchased a license only for object code, and then used a disassembler or other means to modify the software for in-house use, without paying the extra fee for source code, would this be a violation ("alter ... without authorization")?

Software developers could be affected because sometimes the line between copyright infringement and fair use can only be determined by litigation, with no way to be sure in advance. If the penalty for being wrong becomes a felony conviction instead of a civil damage judgment, would product development be unduly slowed by excess caution? Nor is asking permission the answer, for publishers do not ask permission for what they determine is fair use, or rights they otherwise would have had could be lost. An additional complication in software and especially in firmware is that sometimes there is only one correct way to write a short code sequence.

(B) Research Grants

If a scientist receives a Federal or other research grant or contract, and during the study finds another approach which seems more promising, and uses some of the funded computer time to try it, without permission for changing research strategy, could this be a felony under the bill? (The Ribicoff Bill has no gradation of penalties for the size or character of the offense.) Often it isn't feasible to ask permission, because of the difficulty of getting the funding agency to evaluate the request and produce any decision.

(C) Unsponsored Research

Industrial science has a tradition of the unauthorized back-door project, usually a personal passion carried on with a wink from the first level of management. Sometimes this is the only flexibility that exists in either Federal or industrial science, as it is hard for institutions to authorize projects outside their predetermined aims.

In one case I know personally, a technician started work for a large computer manufacturer, and made a series of improvements conservatively estimated to have saved the company more than a million dollars. Afterwards he received the company's highest honor, and went back to school and returned as an engineer. But his early successes were made while the boss was away, and sometimes by coming in at night to complete a project after he had been told not to.

When he returned as an engineer, the workplace was so much more rigid than it had been, that despite his new degree he could not find a creative working environment. He left the company and the field. This case illustrates several things. For us it suggests that a sharp, clear division between authorized and not would threaten the little flexibility which most research workers (other than superstars) now have. Computers, of course, are becoming involved in most scientific research. No one know the cumulative impact of the loss of the many little projects that would never have the chance for the start needed to prove themselves.

(D) Personal Use by Employees

Some computer crimes should perhaps be less than felonies. A programmer who used his employer's computer to handicap the horse races, without paying for the service, is considered a computer criminal. Ironically a collegue who did the same work on his own time but threw the results away without using them might be commended for initiative in professional development. Probably nothing improves programming skill more than a project which turns one on.

Using company computer time for personal projects is a more complicated issue than it might seem. Seldom is there a clear line between professional development and personal use. The

Ribicoff Bill makes no provision for size of the offense, and would cover even the listing of office wall decorations as felonies. Probably most computer users have done things which would be felonies under this bill, and probably they, their employers, and the industry are substantially better off for their having done so. In our field there is no end to the learning curve, and it would be odd if minor abuses of e.g. a copying machine (which has little or no training value) could be handled informally, whereas personal use of the same monitary value in computer time without permission would constitute a felony. If it is not considered criminal to use a typewriter at work for personal purposes, and abuses are well controlled through normal work discipline, why should use of a text editor be a Federal offense? Text editor time is more valuable than typewriter time, but not by much, and increasingly by less. Eventually typewriters may disappear from offices, and only text editors would be left.

It would be unfortunate if an employee who developed software on his employer's computer, then quit, leaving the software behind, and re-wrote it on his own computer and sold it, would be considered enterprising, whereas another employee who did the same thing, only didn't quit, would be criminal.

A complicating factor is that usually an employee cannot possibly pay the employer for computer time, because the installation has no arrangements for retail sale. Why force an employee to go outside and use a different computer system, when using the employer's would have training value benefitting everyone?

(E) Who Can Authorize?

Asking permission before starting a personal project may not be the answer, for who but the owner, who usually isn't present and sometimes doesn't exist, is clearly qualified to give it? If a supervisor could give legal permission, then we have the anomalous situation of the supervisor being able to authorize action which he himself could not take, without (recurive) permission from his supervisor. In practice this means that a supervisor takes a personal risk in giving permission, the risk of being considered a collaborator in crime.

(F) Software Distribution by Professional Associations

The same problem could inhibit distribution of software in professional organizations like DECUS or SHARE. Exactly who can authorize the giving away of valuable programs developed with the employer's resources? An organization could formally designate officials to do so, but in practice how many will get around to it? While it seems inconceivable that Congress would intend to inhibit such professional exchange, how many managers would feel competent to predict what might come out of a court of laymen unfamiliar with the computer industry, and with little or no precedent to guide them? When there is a personal risk, how many managers would play it safe by saying no?

Large organizations don't like to give blanket official permissions, and staying tooled up to make specialized case-by-case decisions is costly. The best way to handle this kind of thing is what's done in actual practice: within a broad range, nobody cares so long as no problems develop, and the rare abuses can be handled on a "management by exception" basis. Under the Ribicoff Bill, the fifteen-year penalty for being wrong would interfere with this flexibility.

(G) Computer Conferences and Scholarly Communication

What happens if a scientist wants to introduce somebody to his collegues on a computer conference, but the new person does not have the status or credentials necessary to qualify for using the conference? This kind of question will become increasingly important.

****

These examples are just the beginning. There could be dozens of others, such as ambiguous "authorization" being re-defined after the fact when higher-ups are desperate to cover for their own misjudgments. Other problems go beyond the computer field, such as whether the Ribicoff Bill could be used as an official secrets act, to allow any information to be kept from the public by being computerized. E.g. could a reporter be prosecuted for asking a source for "unauthorized" information known to be computerized ("directly or indirectly, access"). Another problem is that as computers become involved in almost everything, the FBI would have its pick of ordinary (non-computer) criminal cases normally handled by state or local authorities, increasing the Federal police role at the expense of the

local.

Industry had better raise questions. For there is talk in the Senate of amending the bill to include having knowledge of an offense and not reporting it.

The Ribicoff Bill is intended to scare people, yet it is hard to understand why anyone would want to inject a large dose of fear into the computer industry. In a time of relative decline of U. S. industrial strength, computers has been a shining success. In a time of inflation, computer prices continue their drastic drop. And it's hard to see where computer crime has been so great a problem or threat. Typically computer people are dedicated to their work, and probably less likely than the general public to engage in crime.

The statistics we have back this up. Computer-abuse expert Donn B. Parker of SRI, in recent Senate testimony on the Ribicoff Bill (testimony based on his book Crime By Computer), estimated that in the year 1975, there was one known case of computer crime for each two thousand computers worldwide. The most common route of crime was manual changes to data before it reached the computer; program changes or other acts internal to the computer ranked sixth in frequency. Contrary to the newspaper images, few of the computer crimes were technically sophisticated. Parker also quoted 1974 U. S. Chamber of Commerce estimates of the total dollar loss from all computer crimes being one four hundredths of the total loss from all white-collar crime.

These figures should be reviewed before legislators make it a felony to engage in unsponsored research or professional information exchange without cumbersome permissions. Changing the workplace toward rigidity and fear is more than a matter of crime control. It has inestimable consequences for the future health of the industry.

The crucial work now is to inform the computer industry of the Ribicoff Bill and other policymaking that we should know about, so that we can have intelligent input. That is why this panel was organized.

MY EXPERIENCE ON CAPITOL HILL

By John Draper (c/o Ron Barkin, P.O Box 4399, Berkeley, Ca. 94704)

Early this year, I was invited to Washington to speak to my Congressman about a potential threat to the privacy of every American using the telephone. This threat involves a highly computerized telephone system now being installed, and how it is used or could be used to invade privacy. Early in January I was subpoenaed to testify before a grand jury in Iowa on the existence of certain software within the Bell System which could be used to evesdrop on conversations.

I wrote to Congressman McCloskey detailing this knowledge, which I acquired by pure accident. After doing more research in Washington, I found that most Senators and Congressmen are quite uninformed on how far computer and electronic technology has advanced. About half of them didn't even know what "software" means. With these high-power lawmakers so ignorant of technology, and with so few competent engineers on Capitol Hill, it is no surprise how certain laws get passed.

My observations were as follows. Executives and attorneys representing large corporations like Bell or oil companies are lobbying for the passage of laws that help large corporations, but have little or no concern for citizens. Here is the dilemma: large corporations hire attorneys and executives, and pay their costs for travel to Washington so they can sit in or hearings, talk to Congressmen, and get them to vote for laws that benefit their corporation. But the corporations' employees such as engineers, programmers, and technicians, the people with first-hand knowledge of technology, are not given a chance to look over these laws, at least not as much as the lobbyists. For concerned engineers to take an active part on Capitol Hill would mean loss of wages, even possibly loss of a job. Engineers are too busy with overtime, deadlines, government contracts, and long working hours, to actively take part in politics or government affairs, and are discouraged from doing so. Imagine what your boss would say if you wanted to fly to Washington to speak or submit testimony for or against a law you believed was inadequate or incomplete. I doubt if he would go for it - and I also doubt if many engineers even know <u>how</u> to find out about such new legislation being set before Congress.

As a result, laws like the Ribicoff Bill and others will get passed without any close look into what abuse it could lead to. Computers and software are hard for politicians to understand; and without this knowledge, how could fair laws be passed?

Computers are very versatile, as we know. They can be used for illegal purposes. Some states have laws against possession of a device which could be used to commit a crime. Does this mean that thousands of personal-computer users are breaking the law? Some laws are specific about what kinds of devices, and what they can be used to do that makes them illegal. Others simply state "device". A computer is a "device" that can be used illegally. So is a program. So let's change these old laws to fit new technology.

Once a law is passed, it is much harder to get it changed, and costs much more money. But for personal computer owners and others these laws must be changed, re-worded to cover more specific details of what is or is not illegal.

In summary, I encourage everyone in the scientific community to become more aware of new legislation forming on Capitol Hill and to study these new bills to try to change them <u>before</u> they get passed into law, not after. Also, let's form a group of volunteers who want to take the responsibility of representing engineers, and attend these Congressional hearings in person, and ask to be put on the speakers guest list and speak out and add suggestions and ideas to modify these bills before they get passed into law. So, all you programmers and engineers out there should put those coding sheets and calculators down and think for a few minutes on how <u>you</u> can get involved. It certainly wouldn't hurt.

MICROCOMPUTERS IN LOCAL GOVERNMENT: APPLICATIONS AND IMPLICATIONS

Charles E. Barb, Jr., Ph.D.
Associate Professor
School of Civil Engineering and
Environmental Science and
Department of Geography
University of Oklahoma
Norman, OK 73019

James R. Carter
Assistant City Manager/Technology Agent
City of Oklahoma City
200 N. Walker
Oklahoma City, OK 73102

Abstract

One outgrowth of "personal computing" in that some data processing professionals are becoming interested in the microcomputer for practical local government applications. This paper outlines three types of potential microcomputer applications in local government, sketches illustrative case study examples of each, and summarizes the broader implications of the technology's adaptation. A principal implication foreseen is an expanded role for data processing professionals in systems procurement, development and documentation.

Introduction

The purpose of this paper is to report a preliminary examination of the potential application of semiconductor microprocessor technology in local government and its implications particularly from the local government perspective. The paper is intended for both the data processing professional and those prospectively interested in developing applications for local government. The paper is divided into three parts including a typology of evolving microprocessor and microcomputer applications in local government, sketch case studies illustrating each type of application and a summary of principal conclusions drawn from the inquiry.

Levels of Microcomputer Application

Digital microprocessor technology is fast invading local government data processing as well as most other aspects of local government. Much of it is obscure, however, hidden in a variety of equipment and often used in controls and interfaces. This paper explores the more explicit utilization of microprocessor technologies in local government. Three levels or types of explicit applications are recognizable:

1. Acquisition and use of packaged hardware-software application systems,
2. In-house application software development or adaptation upon low-cost or hobby-type hardware, and
3. In-house hardware-software systems design and development for process control.

In examining microprocessor and microcomputer applications in local government, it is very useful to recognize the different types of applications in order to lend precision to what often becomes an unbounded discussion. Each of the three types of applications have taken place as illustrated in the case studies below.

Application Case Studies

Six application case studies are presented. The first three studies illustrate acquisition and use of packaged microcomputer-based systems. The three case study experiences underscore the potential difference between the adaptation of mass market systems and custom special purpose systems. The next two case studies describe city staff attempts to develop applications software on low-cost hobby-type hardware. A single case case study illustrates the third type of application, in-house hardware-software system design and development for process control.

Oklahoma City Adaptation of Word Processing [1]. Oklahoma City's adaptation of word processing represents a likely common experience in acceptance of what is now fairly fully packaged, tested, yet fast evolving microprocessor technology.

The City of Oklahoma City, since 1973, has acquired eleven stand-alone word processing systems representing a variety of makes, models and levels of technology which are now in use in eight different city departments. Training secretaries to use the equipment has not represented a problem but institutionalization of the equipment has. Initially, word processing was promoted as the basis for organizing secretaries in typing pools for greater productivity. Secretaries and managers have not accepted such reorganization and the equipment subsequently has been promoted and justified on the basis of specific, limited and often time critical applications where text material is repetitively modified or tailored to changing situations. The staffing concept now is to acquire a machine that is shared by a number of secretaries for their separate special applications. Full time machine utilization or rigorous cost-benefit justification of the equipment is not now emphasized.

Example key applications of the equipment by various user departments are as follows. The City Manager's Office has two machines. The key application is developing the weekly City Council agenda which is constantly modified and

updated until its final publication late Friday afternoon. The Legal Department uses their machine for the drafting of ordinances. The Community Development Department uses three machines in part to support plan and report drafting. Personnel uses a machine for producing the many variations of standard job classification descriptions and job announcements. The Water Department maintains standard letters and report forms and a mailing list on their machine. Management and Budget uses their machine during the annual budget drafting process. A City owned convention center uses theirs for contract drafting. And, as separately reported, the Police Department uniquely applies a version of a word processing system for crime data storage and retrieval.

The equipment has been purchased piecemeal on a competitive bid basis by the city as a "solution" to a limited application. With the rapid evolution of word processing technology, this purchase policy has constrained subsequent system upgrading and has led to retaining obsolescent equipment. City staff envisioned that the word processing equipment eventually will be networked into an interdepartmental digital communication network.

Oklahoma City Police Department Crime Analysis System [2]. The Oklahoma City Police Department Crime Analysis System represents a prototype application of a commercially available low-cost microcomputer-based information retrieval package available within a word processing system.

The microcomputer system used in the crime analysis application is an Olivetti 501 text editing system which includes a typewriter, microprocessor with 8K RAM, and double floppy disks. The unit currently sells for $8,324.00 and has been on the market about two years. Reportedly its closest competitor is the IBM System 6. The system can be programmed for either standard word processing or information retrieval. The latter, only, is used by the police department. System software for both the word processing and information retrieval applications are stored on specially prerecorded 10" floppy disks also used in the system for data storage. The prerecorded disks cost $8.00 apiece from Olivetti.

In the information retrieval mode, the system can store 1408 lines of data, 128 characters per line in any number of pre-tabbed user specified fields on each disk. (It can also be configured to store twice the number of lines, half as long.) The data is entered at the keyboard; visual verification of input data is available through a 15 character, scrolling LED display. The system can perform a boolean search of any number of fields and up to six search requests can be stacked. The system can report records meeting the search specification or report the number of records found or the sum of a numeric column. A search of two disks will take less than one minute, however, speed depends on the number of lines of printing produced (the printer operates at 30 characters per second).

The Oklahoma City Police Department Crime Analysis Unit, which has been organized around the Olivetti 501, was modeled after a similar unit in the Phoenix, Arizona Police Department. The Oklahoma City unit was organized in August, 1977, using five civilian CETA employees under the supervision of a police lieutenant, and a leased machine. The hardware was subsequently purchased. The Unit's function is limited to crime analysis in direct support of police operations; it is not a general purpose statistical unit.

Unit staff code and keyboard enter data from standard police reports previously in use in the police department. They maintain nine files: burglary, robbery, sex offense incidents, stolen vehicles and property files, and adult and juvenile suspect files by white and non-white racial classification. Where appropriate, files include the geographic codes for square mile section of the city and recognized resident-organized neighborhoods. Different files are stored on separate disks by month or year depending upon the anticipated size of the file. Files are commonly searched by incident type and/or characteristics, person name and/or person characteristics, property serial number and/or characteristic, or geographic area. The unit also maintains wall pin maps of the incidents.

The unit responds to individual search requests from officers in the field, and also generates special incident alerts based upon the staff's sense of the data they are processing. The unit's service to field officers has grown to about six search requests a day. Only a minority of officers have come to use the service but those that do use it frequently. Special incident alert reports generated by the unit have been instrumental in several apprehensions over the last nine months.

Introduction and use of the equipment has been relatively problem free. Initially Olivetti did not offer local maintenance support which was a problem when a board required repair and problems occurred with bad disks. The system is now down about once a month but repair is prompt.

It is foreseen that with increased demand for searching of expanding data files, file development will likely require a second machine or conversion of the system to a larger computer. The unit has received budget approval for a second machine.

Oklahoma City Fuel Monitor System [1]. The Oklahoma City Fuel Monitor System represents the prototype application of a packaged small computer process control system to a fundamental local government function, and attendant problems.

In the fall of 1974, Oklahoma City staff recognized that approximately 200,000 gallons of gasoline dispensed through the City's motor pool fuel stations were unaccounted for. Alternative types of improved control were considered and the most cost-effective solution found was a central computer controlled system. In 1975 a turn-key system was purchased through open-bidding; installation was completed in early 1977. The purchased price of the system was $136,000 which had an estimated payback period of three years in terms of fuel savings and personnel cost avoidance.

Currently two million gallons of gasoline are dispensed annually through ten geographically dispersed fueling stations which are under control of a Nova II mini-computer. (This mini-computer application is included in the series of case studies because the task is within the realm of micro-computer applications today.) In addition to the Nova, the system includes a Calcomp floppy disk, an interface and telephone communication lines to the fuel stations, and special terminals located on each fuel island. The computer is located at the Public Services Department headquarters. The terminals have a type of numeric keyboard and two slots for the insertion of plastic perforated cards, one issued to the driver and one issued to each car.

The system operates in the following fashion. The fuel islands are self-service. The driver inserts his driver card and the vehicle card in the terminal and enters the odometer reading for the vehicle. The vehicle card indicates the capacity and type of fuel authorized for the vehicle. Before fuel is dispensed the computer verifies both cards and checks the entered mileage against the vehicles preventive maintenance schedule and the previous refueling odometer reading (a logic check to catch gross errors). After the fuel is dispensed, the transaction, with time and date added, is stored on disk. Once a night the disk file is dumped to the City's central computer where it becomes part of the City's equipment management information system. A paper log of all transactions and exception report is routinely generated.

The system was installed on a turn-key basis by a company which previously had sold similar systems for wholesale fuel dispensing at refineries. The City application differed in that it involved a much larger number of small transactions (the City has 3,000 vehicles) at geographically dispersed cites. The system installed also was the company's first application of a floppy disk storage system. These uniquenesses led to some system development difficulties. Other development difficulties were encountered in coordinating construction of fuel islands and cable connections. (The rehabilitation of one fuel island was delayed for a year after the computer's installation.)

Operationally, the dependability of the system has been termed "low." Telephone communications between dispensing sites and between computers has been a problem with the usual attendant "finger pointing" by vendors. Securing maintenance has been a problem. The system vendor will not maintain the fuel island terminals and a proposed service arrangement with the mainframe computer company never materialized. Finally, at the City's insistence, City traffic signal technicians have been trained by the vendor to troubleshoot problems and isolate defective boards or parts for repair by the vendor. Generally, system documentation has been termed "bad." Efforts to train City data processing staff to support the system has been frustrated by successive staff departures. One fuel island terminal is inoperative due to reported software problems. And, overall, there were several changes in project management staff which resulted in inadequate contract administration by the City.

Knoxville Commuter Pool Ride-Sharing Information System [3]. The Knoxville Commuter Pool Ride-sharing Information System represents the potential for development of relatively large-scale applications software on hobby-type computer hardware by city staff.

In transportation circles, Knoxville Commuter Pool is a leading paratransit service development program that has demonstrated the potential for municipally-sponsored and promoted ride-sharing, principally vanpooling, for commuters. The forming of vanpools is basically an information processing problem involving the matching of eight to ten commuters who live near each other and who have a common destination and work schedule. After earlier use of batch matching systems on large computers, Commuter Pool is now implementing a prototype, multi-functional, interactive ride-sharing information system on a consumer-advertised personal microcomputer. System development is being supported by the Urban Mass Transportation Administration of the U.S. Department of Transportation.

The ride-sharing system has four basic functional components

1. Capture and store information on people interested in ride-sharing programs including their name, address, phone, work location and work schedule;
2. Match people to various ride-sharing options based upon residence, work location and schedule;
3. Report alternative transportation available - express or regular bus, vanpooling, or people potentially interested in carpooling - for a given origin, destination and schedule; and
4. Report volunteer agencies available to serve the elderly and handicapped for specific trips.

The system is being designed primarily for use by clerical level staff who directly serve

an applicant by phone in real time. Using the system, staff will record applicant information, search the appropriate bus, van, social service agency, or commuter file to identify and arrange a transportation service for the caller. As a follow-up, the system will produce a personalized letter documenting the call and transportation service options reported.

The system is being implemented on a Technical Design Lab XITAN computer with 64K, four Digital Systems double density floppy disks representing 2.5M characters of on-line storage, a Beehive 100 CRT terminal, and Okidata Printer (which will be replaced with a Centronix 703). The computer was purchased as a kit and assembled in two weeks by three city staff without difficulty or problems. The system operates with an extended FORTRAN IV; TDL was one of the first to implement a microcomputer FORTRAN. The entire system costs in the neighborhood of $13,000.

At this writing phase 1 of system development just has been completed and is being operationally tested and documented. Development took six months and included initial hardware construction, testing and familiarization; development of basic match logics, file structures and file add, delete and change functions; development of a place name-to-zone geocode file and an origin and destination zone matrix key file; development of basic transportation service files for up to 300 regular and express bus routes and 500 vanpools and vanpool drivers; development of a local employer file including their survey history; and development of system access security and user logging systems. Phase 2, which is under disign, will involve development of a file of up to 25,000 commuters by name and commuting characteristics. The commuter file will support carpool matching. It is anticipated that the phase 2 capability will be operational during the Fall of this year.

To date, several significant and lesser problems have been encountered in system development. First, hardware delivery was delayed six months, reportedly due to a subcontractor strike. The FORTRAN package has worked well but error messages are very limited. One bug in the FORTRAN package was found and corrected by TDL. The disk operating system, however, apparently does not allow application system access to disks two through four, only disk one. Since phase 2 system development requires these additional disks, this problem will have to be corrected or the system will have to be updated in the immediate future. Staff feel it is likely that the system implemented on the TDL Computer will eventually require a second duplicate backup system for operational dependability. The Commuter Pool staff at this point are reconsidering their hardware requirements not only due to the limitations above but due to expanded applications being envisioned and the availability of new multi-user time share microcomputer systems that are coming on the market.

A second problem occurred with the consultant producing the application software. Due to the consultant's loss of their project programmer, Commuter Pool staff had to pick up and program some of the system. It should be noted that the Commuter Pool computer staff is one person with a computer science, engineering and transportation background who also developed the basic system specifications.

A third problem encountered is staff acceptance of the new system. It is reported some staff are resisting being weaned away from old manual procedures. More training and familiarity with the system and completion of all system capabilities is viewed as the solution to this problem.

San Jose City Agency Microcomputer Application Program [4]. The San Jose microcomputer application program represents the potential for direct user agency staff application of small microcomputers in local government.

The City of San Jose, CA, has leased and installed seven 8 Bit, 8080-based DTC microcomputer systems with floppy disks. The DTC system is expandable to 64 K memory, offers an option of two or four 8" floppy disks with a potential of 1M bytes on line. The system comes with an extended DOS BASIC and 8080 assembler, basic text editing and form letter writing programs, and a tutor and standard games package. The system sells for $4900.00 for a 16 K, 2 disk configuration minus terminal. The City has leased a variety of configurations and uses a variety of terminals including DECwriters, graphic and alpha-numeric CRT's and daisy wheel printers. Most of the systems have been in use in the city for about a year.

At present five city departments are using the computer which represents their first application of computers to their operations. The Public Works Department has two computers. One is used for traffic signal development (see below) and an inventory control system, and the second is being used in a variety of applications including equipment replacement scheduling, street crew performance analysis and subdivision application processing including the generation of necessary checklists, information letters and even the City Council resolution accepting the subdivision. The Property and Codes Department also has two systems and uses them for monitoring building permit application processing, delinquent garbage collection and contractor liability insurance among other applications. The Planning Department has one computer and uses it, among other applications, to generate statistical summaries of active construction using a copy of the Property and Code building permit disk file. The Economic Development Department has one computer and uses it to maintain an industrial vacant land inventory to support industrial development

programs. The seventh computer is maintained by the City's Science and Technology Office as a backup and applications development facility.

Adaptation of microcomputers in San Jose has been coordinated by the City's Science and Technology office. It began with staff training programs in BASIC obtained through local junior colleges and the Lawrence Hall of Science. All of the above applications have been developed in BASIC. About forty staff have taken the programming courses; about six are currently involved in active software development, one per department. The Science and Technology office also has a highly proficient BASIC programmer who aids the departmental programmers. The individual machines were individually cost justified relative to specific applications and acquired. Hardware dependability has been good; most problems have been in the disk drive units and I-0 equipment. Equipment needing repair is taken to local commercial service facilities. Temporary sharing of other department systems has reduced user inconvenience when equipment is down. Data exchange occurs between the microcomputers systems through disk copying. The machines also exchange data with the City's Burroughs 3700 main frame computer over a telephone modem link.

The application of microcomputers was forced by constraints in the city's central data processing center. The data processing staff are reportedly antagonistic towards the micros; in part they have a COBOL programmer's prejudice toward BASIC. A monthly meeting is held between the microsystem operators and the central data processing staff for coordination. Application programs developed by departmental staff have tended to be relatively short, straight forward programs in the order of 50 lines of code and taking less than a month of effort. Software development has not been controlled or standardized to date; central data processing staff have expressed some interest in assuming this function.

System development to date has been dependent upon the personal initiative of the agency staff "turned on" to programming. These staff tended to be lower or mid-management level who recognize the computer as a practical tool in specific aspects of their job.

San Jose Microprocessor Traffic Signal Controller [4]. The San Jose microprocessor traffic signal controller project suggests the possibility that city staff can design and produce microprocessor-based process control systems at a savings.

The City of San Jose has in the order of 500 signaled intersections. It has been in the process of upgrading electro-mechanical controller units and signaling new intersections at the rate of about 25 per year. City electrical shop staff have traditionally designed, fabricated and installed all loops, cabling and control boxes and installed purchased microprocessor controller units. The microprocessor controllers have cost in the neighborhood of $4500 apiece. It is projected that city staff can design and fabricate comparable microprocessor controllers for $1500 apiece - a savings of $3000 per unit and $75,000 per year. This would also solve another problem the city has in equipment standardization; currently controllers are purchased on a low bid basis which has resulted in a potpourri of makes and models.

The controller that has been designed is a standard 8 phase unit that will be software modified for other phase configurations. The design has been breadboarded and successfully field tested. Staff are currently drawing the printed circuit board artwork; the boards and all components will be commercially procured. It is anticipated a prototype unit will be completed and field tested in about six months. The contrcller will likely include two boards, one with an 8080 CPU, EPROM or masked ROM memory, RAM memory, and an I/O board. The critical design factor is temperature. The breadboard prototype operated satisfactorily with commercial spec components; production controllers will use now-available military spec components. A city decision on use of the controller is expected within the year.

A program toward application of microprocessor technology was initiated by the City's Science and Technology staff three years ago through a digital electronics course developed by the Lawrence Livermore Laboratory of the University of California. The course included a series of videotape lectures and microprocessor breadboard laboratory experiments. Ten city staff took the course and subsequently two of the staff from the electrical shop developed the traffic signal controller application. Both staff were electricians: one with a college degree in mathematics, the other, one year in a junior college. Both were enthusiastic about the project. City traffic engineers, however, have been noncommital and aloof of the project and they generally don't foresee potential follow-on applications of the controllers such as traffic counting or controller networking. It is reported that the traffic engineering staff has a conservative view toward signal control development in part due to the failure of a central computer controlled 60 signal system some years ago.

San Jose is a high technology area with generally competent city staff as a result of a competitive pay scale. It anticipated keeping the electrical staff it trained in digital electronic design, however, one of the two electricians has left the City for other employment.

Initial Conclusions

The above anecdotes broadly suggest the type of problems and implications encountered in

adapting microprocessor technology in local government. The following discussion highlights potential data processing management concerns, introduces idiosyncrasies of local government which will influence the technology's adaptation, and underscores the broader information systems implications of the technology.

Management Concerns. There are three principal areas of concern professional data processing managers will likely have regarding local agency adaptation of microcomputer and microprocessor technology. The first is the viability of the equipment to reliably support day-to-day user application. Experience to date suggests mechanical periphereals, particularly floppy disk drives, are the principal problems today. While equipment engineering is improving, it remains prudent to plan for hardware redundancy or interchangeability as illustrated in the San Jose case study. A related problem is the availability of equipment maintenance services over time. With such a fast developing technology, equipment obsolescence is particularly severe and the availability of repair parts and service staff familiar with and willing to work on old equipment may be limited. Another dimension to the equipment concern is replacing or upgrading equipment. The limited interchangeability between today's hardware underscores this problem. Small computer manufacturers - often short-lived - often do not offer effective means to transfer developed systems and files to new equipment.

A secondary concern is system software. At present, higher level languages - beyond BASIC - are just not appearing on the market. Most of the packages are relatively primitive in error message aids. Applications software development isn't as easy as on established commercial systems. A common way around this is to develop software on larger commercial systems and then transfer completed packages to the end resident microcomputer.

A third, and possibly primary concern is the need for local management of system procurement and applications development. Generally greater care will need to be exercised in drafting system specifications in the system procurement process. Small tailored hardware systems are implicitly less flexible than large scale general purpose equipment for system development or modification. Another management concern is the supervision of application software development, often by inexperienced programmers in remote user departments. While BASIC programming is relatively straightforward and easy to master, application programs developed by inexperienced staff will likely be poorly conceived, executed and documented which will present problems for subsequent maintenance. Professional supervision of software design, development and documentation will be locally needed.

Local Government Idiosyncrasies. Local government has several unique characteristics, distinct from money-making organizations, that will likely influence microcomputer adaption. It will be useful for those looking to local government as a potential market to keep these factors in mind. A primary characteristic of local government is that it generally has not been organized to be innovative, per se, but to react to and solve a limited range of community problems. The problems are rarely of a high technology nature and solutions are commonly copied from the successful experience of other governments. Local governments also tend to want to buy a permanent technological solution to a problem -- to be done with it. They do not acquire technology to obtain a competitive edge as is commonly the motive in business.

Second, local government has a broader range of concerns and functions than business and there is also a lack of standard procedures and processes in these functions across local government even within the same region. Subsequently, most local government systems, beyond basic accounting and billing functions, must be customed designed or extensively tailored during installation. And third, local government is frequently a poor paying employer. Consequently, staff technical competence and experience is often low and staff turnover high. Local government information system development should be approached from the perspective of "appropriate technology" in the same sense it is often applied to the transfer of technology to third world nations. All too often local systems designs are not appropriate to the technology level of the local government environment and they fail.

System Implications. In conclusion, it is appropriate to recall the broader definition of information systems [5]:

> "... a collection of people, procedures, computer hardware, computer software, and a data base organized to develop the information required to support a particular mission."

From this perspective, microcomputer technology merely represents a continued decline in the cost of the hardware component of information systems. Experience suggests this shift results in pressure on the people component of systems. Microcomputer technology will likely require that data processing staff, if they are to participate in the microcomputer revolution, acquire broader language skills, a broader knowledge of hardware and communication systems, and more skills in managing overall system procurement, development and integration, and documentation.

Summary

In summary, the paper has suggested three approaches that local government may take in

adapting microcomputer technology: purchase of packaged systems, in-house software development on hobby-type equipment, and in-house design and development of hardware-software systems particularly for process control applications. Six studies suggested the types of prerequisites to, problems and implications of adapting the technology. And finally, the paper briefly discussed the principal technical concerns of data processing professionals in regard to the technology, idiosyncrasies of the local government environment influencing its' adaptation, and the broader information system implications of the adaptation process.

References

1. For further information, contact James R. Carter, Office of the City Manager, City of Oklahoma City, City Hall, 200 North Walker, Oklahoma City, OK 73102. (Phone: 405-231-2345)
2. For further information, contact Lt. Russell Rigsby, Crime Analysis Unit, Oklahoma City Police Department, 201 Colcord Drive, Oklahoma City, OK 73102. (Phone: 405-231-2201)
3. For further information, contact Thomas Bennett, Knoxville Commuter Pool, City Hall Park, Knoxville, TN 37902. (Phone: 615-637-7433)
4. For further information, contact Monroe Postman, Science and Technology Office, City of San Jose, City Hall, 801 North First Street, San Jose, CA 95110. (Phone: 408-277-4220)
5. U.S. Department of Housing and Urban Development. Urban and Regional Information Systems: Support for Planning in Metropolitan Areas, Washington, DC: U.S. Government Printing Office, (1968), p. 18.

MICROCOMPUTERS IN CITY GOVERNMENT

Monroe H. Postman, Technology Agent, City of San Jose, 801 N. First Street
San Jose, CA 95110 (408) 277-5171

Abstract

There are presently seven microcomputers in the San Jose City Offices all programmed and operated by departmental personnel. The tasks include business and computational applications, word processing and control system development. This last project involves the in-house circuit and programming design of a microprocessor traffic signal controller.

The introduction of microcomputers into the San Jose City offices was a logical follow-on to a do-it-yourself BASIC programming project. The central EDP shop was overloaded and could not supply software services. The City computer was not configured initially to supply timesharing capability and when that did occur, the response time was too slow. The DTC "Microfile" microcomputers which were acquired provide up to 64K of RAM and up to 1.2M bytes of floppy disc storage. Two of the units are provided with answer modems for remote access.

The programs which have been written are usually less than 50 lines of BASIC code and include communication programs to link the microcomputers to the central City computer, a Burroughs 3700. Files are transferred between departments using floppy discs. A library of universal programs is available to all departments in addition to the editing programs which were supplied with the computers.

Machine reliability has been excellent. Estimated total downtime is five days out of seven machine years. When a machine has been down, work can usually be done on a neighboring unit, thus providing an order of magnitude greater availability than most large systems.

The only apparent drawback is that motivation to program the computers is a requisite to success. This has slowed down adoption in only one case but should be considered before embarking on a program of this type. Estimated annual savings at $200,000 have been attributed to this project by the involved departments.

ABSTRACT
Copyright and Software:
Some Philosophical and Practical Considerations

By Kenneth S. Widelitz, Attorney-at-Law
10960 Wilshire Blvd., Suite 1504
Los Angeles, CA 90024
Telephone (213) 477-3067

The New Alchemy

What's all this brouhaha about whether computer programs should be covered by copyright. One reason for all of the concern is because microcomputers now allow us to make gold from lead. That is, into one disk drive you insert a disk with a program on it which sells for $1000. Into a second disk drive you insert a blank diskette. You type "filecopy" and seconds later you hold in your hand 2 diskettes worth $1000! That's alchemy in my book.

Of course, copying along similar lines goes on virtually every day when a person tape records a friend's record album. But in that situation you are dealing with a $5 item and it takes 45 minutes to reproduce it. For the first time in man's history an item with a large economic value can be duplicated in an amazingly short period of time. That's incentive. Put incentive together with alchemy and you get a large scale problem.

Protection of Software

The National Commission on New Technological Uses of Copyrighted Works (CONTU) was charged by Congress with examining the problems involved in the protection of computer software. CONTU has just submitted its final report to Congress. In that report CONTU recommends that the Copyright Act of 1976 be amended to make it explicit that computer programs are the proper subject matter of copyright.

CONTU's earlier report, issued in June, 1977, discussed the issues involved in reaching such a determination. That report discussed the three possible vehicles for protection of computer programs, namely, copyright, patent and trade secrecy. As the report stated, "...copyright is designed to protect the expression of ideas while patent's purpose is to protect what are generally understood to be inventions--in a sense the ideas themselves." Patents protect inventions which must be useful, novel and not obvious to those familiar with the related technology. Trade secrecy involves the dissemination of information pursuant to contractual agreements by the terms of which the party receiving the information promises not to reveal it to anyone else.

Philosophically speaking, issues raised in discussing whether software should be protected by copyright involve the use of analogies in which the statement "computer programs are more or less like..." are made. The question becomes, is a computer program a writing and/or a mechanical device. Other "philosophical" issues involve the question of when a copy of a computer program is made. Is it made when it is loaded into memory or when it is dumped onto tape or hardcopy. Another issue involved is when is a program based on a previous program a derivative work. That is, an author of a book written in English has the exclusive right to translate into French (read BASIC for English, COBOL for French?)

The Copyright Act of 1976

The Copyright Act of 1976 became effective January 1, 1978. That copyright law makes the term of copyright equal to the life of the author plus 50 years. The law also makes the concept of "publication" less important. The law also defines when a work is "created," when a "copy" is made and what constitutes "a work made for hire."

Under the Copyright Act an author has the right to do any of the following: (1) Reproduce the copyrighted work and copies; (2) Prepare derivative works based upon the copyrighted works; (3) Distribute copies of the copyrighted work to the public by sale or other transfer of ownership or by rental, lease or lending. Each of these rights are

independent of one another so that any one may be retained while any other is sold or transferred.

The question as to what constitutes a derivative work is one of the toughest that computer programmers will face. It must first be understood that copyright protection does not extend to any idea, procedure, process, method of operation or concept. Copyright protection extends only to the expression of ideas, not the ideas themselves. Of course, some ideas, concepts, procedures or processes are so basically fundamental that copyright does not protect an explanation of them. For example, some sorting routines are so basic that they cannot be copyrighted. However, if a series of basic routines are put together so as to accomplish a specific task, that program is subject to copyright.

In order to obtain copyright protection, there is certain procedural technicalities. These are notice, deposit and registration. Notice requirements are simple. They consist of a C in a circle, the word "copyright" or the abbreviation "copr.;" the year of first publication and the name of the copyright owner (i.e. Copyright 1978 by Kenneth S. Widelitz.) There are rules relating to where the notice must appear depending upon the type of work to be copyrighted. The copyright law also requires that materials that are copyrighted be deposited in the Library of Congress. There are exceptions if the Library of Congress does not desire specific material. Registration with the Register of Copyrights is a prerequisite to the bringing of a law suit for the infringement of the copyright.

Another consideration is the Doctrine of Fair Use. Fair use embodies that notion that a reasonable portion of a copyrighted work may be reproduced without the permission of an author for a legitimate purpose. The doctrine is most often applied through teachers who xerox materials for distribution to students for educational purposes.

CONTU's final report recommends that the Register of Copyrights should adopt appropriate regulations regarding the notice, deposit and registration of computer programs.

COPYRIGHT AND COMPUTERS

Neil Boorstyn
Phillips, Moore, Weissenberger,
Lempio & Majestic
Three Embarcadero Center
San Francisco, CA
(415) 421-2674

1. COPYRIGHTING COMPUTER PROGRAMS

Since 1964 the Copyright Office has accepted registration of computer programs, despite lack of any then existing statutory authority. As a condition of registration, the Copyright Office required that the program constitute sufficient original authorship, that it be "published" with copyright notice (i.e., copies of the program capable of being eye-readable were sold to the public), and an eye-readable print-out of the program was deposited with the Copyright Office.

The new Copyright Act, which became effective on January 1, 1978, includes in its definition of "literary works" all works expressed in words, numbers or other verbal or numerical symbols or indicia (Sec. 101). Therefore, computer programs and computer bases are now statutorily copyrightable to the extent that they are original works of authorship. That is, the program must be the programmer's expression of his original ideas, as opposed to the ideas themselves.

Copyright protection does not extend to the idea, procedure, process, system or method of operation (Sec. 102b). Only the written expression of the programmer's ideas are protectable.

Copyright protection exists from the time an original work of authorship is fixed in any tangible medium of expression from which the work can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. The fixation must be sufficiently permanent or stable to permit perception, reproduction, or communication for a period of more than transitory duration.

If the program is prepared over a period of time, that portion that is fixed at any particular time is a copyrightable work as of that time, and if the program has been prepared in different versions, each version constitutes a separate protectable work (Sec. 101).

A copyright notice should be affixed to all copies of the program that are published, i.e., sold or distributed. The notice may be placed on the medium embodying the program ((punch cards, magnetic tape, diskette, etc.) or in the program itself so that it is readable on all print-outs. The program should be registered with the Copyright Office and copies of the program deposited. Publication is no longer a condition of registration and unpublished programs may now be registered.

As the copyright owner of a computer program, the programmer owns a "bundle" of exclusive rights: for example, the right to make and sell copies, the right to translate the program into other machine languages, and the right to display the program publicly (Sec. 106).

The copyright protects against unauthorized uses of the program. Copying of the program without permission is prohibited. A purchaser or lessee of a program however, is free to copy the program in order to use it. The ideas, system, or process embodied in the program are also free. The programmer's expression of his ideas may not be copied nor may his efforts in selecting, sorting, arranging, designing, and constructing the program be copied.

If an idea or procedure is reduced to written form, the method of expression of the idea will be protectable, even though the idea itself is not.

2. COMPUTER DATA BASES

Section 101 of the new Copyright Act defines a "compilation" as a work formed by the collection and assembly of data that are selected or arranged in such a way that the result constitutes an original work of authorship. Examples of copyrightable compilations include catalogs, directories, and computer data bases. The data or materials comprising a compilation need not be copyrightable. If the underlying data or materials are copyrightable, i.e., original works of authorship, then the resulting assemblage is a collective work.

Therefore, data bases comprising the selection, organization, and arrangement of pre-existing materials are copyrightable and, if copyrighted, would be protected against unauthorized copying and use. Copyright protection extends only to the particular compilation of material (selection, arrangement, etc.) contributed by the author, and not to the underlying materials. If the pre-existing materials are protected by copyright their unauthorized inclusion and use as part of a data base may constitute an infringement.

The new law (Section 117) leaves open, for the time being, the question of whether the unauthorized in-put of a copyrighted work into a computer constitutes an infringement. But, the retrieval of the work in a print-out would be an infringement unless the doctrine of fair use were applicable, e.g., a non-profit educational use. The Act does not deal with the use of copyrighted works in conjunction with automatic storage or retrieval systems. Legislation in this area may be enacted at a later date. However, the copying of a copyrighted computer program on tape or disk for subsequent in-putting into a computer will be an infringement because such copying is not a "use of the work in conjunction with automatic systems capable of storing, processing, retrieving, or transferring information" within the contemplation of Section 117.

PATENTABILITY OF COMPUTER SOFTWARE

Martin C. Fliesler, Attorney at Law
Phillips, Moore, Weissenberger,
Lempio & Majestic
Three Embarcadero Center, Suite 2900
San Francisco, California 94111

Tel: (415) 421-2674

I. Basic Principles of Patent Law for Understanding the Issue of Software Patentability

A. The Patent

1. Physical Aspects - purposes and differences between the specification and the claims.

2. Legal Aspects - a patent is in the nature of a contract between two parties, i.e., the inventor and the Government.

B. The Invention

1. An invention is needed before there can be a patentable invention.

2. Requirements for an invention
   a) Conception - the mental act
   b) Reduction to practice - the physical act

C. The Patentable Invention - Requirements

1. Patentable subject matter - the key issue in the area of the Software invention (35 USC 100 and 101)

2. Utility (35 USC 101)

3. Novelty (35 USC 102)

4. Unobviousness (35 USC 103)

D. Procedure for Obtaining a Patent

1. Prosecution before the Patent Office Examiner

2. Appeal to the Patent Office Board of Appeals.

3. Appeal from the Board to the Court of Customs and Patent Appeals (CCPA)

4. Appeal from the CCPA to the United States Supreme Court.

II. The Patentability of Software Inventions

A. The present state of the law

1. The Supreme Court majority view - the Benson and Flook cases.
   a) Algorithms though conceived by the inventor, are presumed to be in the prior art.

b) Claims which essentially preempt an algorithm are not patentable.

c) Patentability must lie in some inventive concept in the application of the algorithm.

2. The Supreme Court minority view - clear, crisp and in conformity with heretofore recognized patent principles, but unfortunately not the law.

B. The post-Benson and Flook era.

1. The Patent Office view - will it read the Supreme Court decisions broadly and continue to reject most software related inventions.

2. The CCPA view - will it read the Supreme Court decisions narrowly and continue to allow some software related inventions.

3. Possible areas of patentable subject matter.

a) Specific programs developed from generalized formulations for programs to solve mathematical problems.

b) Generalized programs combined with an otherwise novel process

c) Broadly claimed apparatus performing the algorithm

PROTECTING SOFTWARE WITHOUT PATENTS-- WHAT ALTERNATIVES

© 1978 David B. Harrison, Esq.
Owen, Wickersham & Erickson
433 California St., San Francisco, CA 94104
(415) 781-6361

Abstract

Recent developments in the law concerning the nature and extent of software protection are discussed, including some limitations and drawbacks of copyright. A new law is needed lying somewhat between copyright and patent, to provide reasonable protection to novel innovative programming and concepts.

A lot of water of legal development and thinking has spilled over the races of the intellectual property law dam since last we spoke with you at the Second West Coast Computer Faire held in March in San Jose. At that conference, I presented a paper which surveyed the patent system from a vantage of personal computing. The paper is reproduced in the Proceedings of the Second Faire (beginning at p. 105), and its my intention to minimize repetition of that material today.

In that paper I mentioned a case which was then pending before the United States Supreme Court concerning the patentability of software. As many of you probably already know, on June 22, 1978, in a 6-3 decision, the Supreme Court has once again struck down an application for a patent upon an invention which centrally involved a computer program. In that case, known as Parker v. Flook,[1] the patent application described a method for updating alarm limits. Such alarm limits were said by the inventor to be very important in monitoring and thereby ensuring proper operation of oil refineries involved in catalytic chemical conversion of hydrocarbons. The Supreme Court began its analysis by defining an alarm limit as a number, and then went on to characterize the patent application as nothing more than a combination of old steps except for a new formula or algorithm. Here's the language used by Mr. Justice Stevens, the author of the opinion:

> "All that it [the patent application] provides is a formula for computing an updated alarm limit. Although the computations can be made by pencil and paper calculations, the abstract of disclosure makes it clear that the formula is primarily useful for computerized calculations producing automatic adjustments in alarm settings."[2]

So, at the outset the Court clearly recognized that in broad terms it was struggling with the patentability of computer programs, and even more broadly, the patentability of all software. As recognized by the Court, the value of software in the United States today is staggering. In fact the Court,[3] in a footnote, acknowledges an industry estimate that the value of computer programs in use in the United States in 1976 was 43.1 billion dollars, which figures out to be over 2 1/2 percent of the total Gross National Product for the United States that year.[4] I'm sure that all of you will agree with me that the value of useful software in this country has increased dramatically even since 1976. So there is no doubt that we're dealing with a subject that cries out for effective, appropriate legal protection. But what kind of protection is apt to be effective and appropriate, and what social values or standards does one use to determine what is effective and appropriate protection in the first place.

Let's start by reviewing some of the important characteristics of the patent law as it bears upon software protection.

A patent is intended to give an inventor a short-term seventeen year powerful monopoly in and to the invention and in and to all articles, machines, processes (i.e. methods), or compositions of matter which are physical embodiments of the invention. A patent protects the inventive concept

or idea itself from exploitation by others during its term. Thus, if a patent were granted upon a software concept or idea, the patent would block everyone but the inventor from use of not only the particular software code set forth in the patent but also any other software expression which achieves the same functional result in a different but equivalent way.

It becomes clear that if patents were available to protect innovative software, whole fields of application of general purpose computers, minicomputers and microcomputers might be entirely preempted by one or several patents. Such patents would severely restrict and limit manufacturers and users of computer hardware and thus inhibit the the widespread sale and use of computer equipment of our society. From this point of view, it is not surprising that the Computer and Business Equipment Manufacturers Association argued against patent protection for software before the Supreme Court in the Parker v. Flook case.

Moreover, the task of examination of software patent applications by the Patent Office to rule upon patentability is a most difficult one at least. The Patent Office is not well equipped to analyze code, except perhaps programs written in very high level languages and fully documented and explained in the patent application. Classification of software patents poses another significant problem to the Patent Office. These problems may explain some of the reluctance of the Patent Office to entertain software patent applications, but these problems are no excuse for developing new ways of examining and classifying patentable innovations which happen to be embodied in software.

If the patent system exists to promote the useful arts by providing inventors with incentives for making their inventions public through published patents, why shouldn't there be patent protection for truly remarkable and innovative software? After all, the creation of such software may involve very substantial commitments of manpower and financial resources of a company, resources which will be rendered valueless unless some form of effective protection of the innovative programming strategy or concept is available. Without patent protection, it was argued by the Software Generation industry representatives at the Supreme Court, there would be no sufficient incentive or hope for reward for truly innovative inventions which are embodied in software.

This was the dilemma faced by the Supreme Court in the Parker v. Flook case. The answer handed down is not a startling one, even though it may be a disappointing one to programmers and software manufacturers and vendors. The Court discussed a long line of its prior decisions which have consistently held over the years that " . . . phenomena of nature, though just discovered, mental processes, and abstract intellectual concepts are not patentable as they are the basic tools of scientific and technological work."[5] The Court tried to draw the line between scientific truth or principle on the one hand which denies patentability to software and novel and useful structure embodying the truth or principle on the other hand granting patentability to hardware. The Court analogized to an earlier decision[6] which upheld a patent for a directional antenna having a physical geometry dictated by an algorithm which related desired performance characteristics to the antenna's geometry. In that case, the patented combination of the new antenna and the new algorithm was held to be not infringed, i.e.,a patent which was valid but not enforced under the facts.

In the case just decided, the Court assumed that the only thing new was the algorithm or concept embodied in the program--everything else was said to be old and well known. The Court finally decided that Flook's concept was not patentable under existing law because it constituted the mere discovery of a phenomenon of nature embodied in a mathematical formula, notwithstanding the fact that Flook's formula was empirically developed rather than stated as a derivative of pure mathematical reasoning.

As in its earlier cases involving patentability of software, the Court limited its decision to the particular facts before it, and acknowledged that a patent could be obtained for inventive applications and physical uses of a newly discovered algorithm. In simpler language this means that if you have new and patentably inventive hardware along with the novel computer program, you should be entitled to a patent on the combination, as was true in the antenna case. If you have old hardware and new software only, then under present law, you will not be able to get a patent.

As the Court reached the end of its decision, it apologized to the public that its conclusion was based on reasoning taken from its older decisions which were written in a pre-computer, pre-software era. It then tossed the gauntlet to Congress to provide solutions to what it characterized as the "difficult questions of policy concerning the kinds of programs that may be appropriate for patent protection and the form and duration of such protection . . . ."[7]

On July 31, 1978, just 39 days after the Parker v. Flook decision was handed down by the Supreme Court, the National Commission on New Technological Uses of Copyrighted Works (CONTU) presented its Final Report[8] to the President and Congress. As you are probably aware for the last three years CONTU has been studying and holding public hearings on how best to protect software, including both computer programs and mechanized data bases. Since the Commission was formed under provision of the recently enacted Copyright Act of 1976,[9] it is not surprising to find that CONTU has concluded that copyright protection as defined and governed by the 1976 Copyright Act, is the most appropriate form of legal protection for software and recommended minor legislative changes to assure application of the 1976 Copyright Act to software.

Personally, I disagree. The basis of my disagreement with the CONTU conclusion is not that copyright protection is inappropriate: copyright protection is highly appropriate at the minimum. My disagreement is that, to my view, copyright protection does not go far enough. But, before we reach the pros and cons, let's briefly consider the nature of the copyright.

The purpose of copyright is to grant to authors of original creations a property right which is limited to the precise form of the author's expression of the idea or concept. In accordance with the plain meaning of the word "copyright", the copyright laws protect only against copying the author's expression of the underlying idea or concept. Without copying, there is and can be no infringement of a copyright. What this means in practical terms is that two authors can write identical books or two programmers can write identical computer programs with neither one of them infringing the copyright of the other, provided that each author or programmer comes up with an original creation without copying the other's particular expression of the idea or concept.

CONTU recommended copyright protection for software for a number of reasons:

First, since 1964 the Copyright Office has granted copyright registrations on programs, and this practice has never been challenged in the courts Thus, there is a historical precedent supporting copyright for software.

Second, if present computer industry trends continue, programs written by non-hardware manufacturers, i.e., the so-called software houses, will gain an increasing share of the computer market not only because programming and hardware construction are vastly different skills but also because programming requires little capital investment. (As used by CONTU, "capital investment" may be misleading--the cost of machine time and programmer's salaries often become very significant investments, which might properly be characterized as "capital investments" in the resulting program itself.)

Third, (and here I agree with the CONTU report) the cost of copying a program is far less than the cost of originally developing it.

So, on these three grounds, the CONTU Final Report concludes: ". . . the continued availability of copyright protection for computer programs is desirable." I certainly agree, but as I've said, in my opinion copyright doesn't go far enough.

Consider this situation: a programmer struggling with a systems problem discovers a radically new strategy for programming the system or subsystem, a strategy that has not been known before and would have immediate widespread application in similar systems. The programmer writes a program which implements his new strategy. The program is copyrighted from the moment of its creation (as are all works of original creation), and it is also registered in the Copyright Office and sold to users with the copyright notice printed on the container, the media reel and contained in software at the head and tail of the program which causes it to print out when the program is both listed and run. Another programmer obtains a copy of the program by purchase, and through a careful study of the program discovers the new strategy. This other programmer then writes his own programs using the newly discovered strategy. Under the copyright law, there is no infringement. Anyone is free to use the strategy, which means that while it is valuable,

it is not protectible. While the strategy may be copied, the original program which gave the strategy away, may not be copied. It seems to me that there ought to be some form of protection for that radically new and valuable strategy, protection which the copyright law simply does not provide.

Copyright has two other limitations which are probably advantages, rather than drawbacks, in connection with software protection. The first limitation is that copyright will not protect works which do not qualify as "the fruits of intellectual labor." Thus, you cannot secure copyright upon, e.g., unrecorded magnetic media, or an instruction printed thereon which simply reads "Insert into disc drive this end up." Similarly, a "program" consisting of a very few, obvious steps is not a proper subject of copyright.

The second exception to copyright protection occurs where the concept or idea and the expression thereof are identical. In the computer context (and this has particular relevance in microprogramming), this exception means that when specific instructions, even though previously copyrighted, are the only and essential means to accomplish a given task, later use of the same instructions for the same task will not amount to copyright infringement.

As a practical matter, these exceptions will be raised as defenses in court proceedings for infringement; and even though they may be complete defenses, they do not guarantee that you won't be sued for copyright infringement and put to the expense of defending the lawsuit.

Now that we've identified what I view to be a weakness in copyright protection for software which is the total lack of protection of novel, innovative programming concepts, strategies and ideas, we reach a more fundamental inquiry: how should one plan to recover software development costs and realize a fair profit or return on the investment? Put another way, what contour should the law take to encourage development and widespread usage of not only programs but also programming strategies? Turning once again to the CONTU Final Report, we find four possible bases which encourage dissemination of software:

> "(1) The creator can recover all of its costs plus a fair profit on the first sale of the work, thus leaving it unconcerned about the later publication of the work; or
> (2) The creator can spread its costs over multiple copies of the work with some form of protection against unauthorized duplication of the work; or
> (3) The creator's costs are borne by another, as, for example, when the government or a foundation offers prizes or awards; or
> (4) The creator is indifferent to cost and donates the work to the public."[10]

Now, as to the altruism of the fourth approach, while each of us strives toward benevolence in greater or lesser degrees, it is simply not realistic to expect people to make innovations in software or in any other branch of technology without some form of incentive which requires some form of legal protection.

The first possibility has a fundamental flaw and that is even if the original creator of the software is paid in full by a purchaser, what kind of protection can the purchaser expect for his investment. Moreover, the first possibility, full recovery of investment on first sale, would make the cost of programs too high for widespread purchase and use.

The third possibility, patronage by government or foundations, is valid in only a very limited and narrow sense, and would not tend to encourage software development in the commercial sector.

So this leaves only the second possibility: recovery of investment by spreading costs over multiple copies of the work. If this approach is feasible, i.e., there is a market for multiple copies of the same program, then the copyright laws should provide ample protection. On the other hand, if the real value lies in the programming strategy, rather than in any particular program or expression of the strategy, and the market is for multiple adaptations and widespread and varying implementations of the programming strategy, copyright protection breaks down, and without a patent on the strategy, there is no protection except to keep the strategy a secret. And, a secret inherently inhibits progress in the useful arts, which most certainly includes computer technology.

CONTU noted other forms of legal protection for software that ought to be mentioned.[11] Besides patents,

and as just suggested the doctrine of trade secrecy may have applicability to software. The doctrine of trade secrecy protects business operators who use a formula (i.e., computer program),pattern, device or compilation of information (i.e. a computer data base) in their business which gives them an opportunity to obtain an advantage over competitors who do not know it or use it. This is a state law theory, and such laws vary from state to state, the main drawback of trade secrecy is the requirement of maintaining secrecy over the trade secret, although in some jurisdictions, such as in California, the courts have adopted a test of so-called "relative" secrecy rather than absolute secrecy. Nevertheless, the costs for security precautions to protect trade secrets is staggering and perhaps too often futile. Once the secret becomes known in the industry, trade secrecy protection evaporates.

Usually, trade secrecy cases have arisen in two types of situations both growing out of contractual or confidential relationships. In fact, the existence of a confidential relationship is one of the strongest factors considered in cases involving the protection of trade secrets. In the first typical situation, a former employee has appropriated a trade secret for his own benefit, or he has disclosed it to a third party such as a new employer, in breach of a provision of an employment agreement prohibiting disclosure of trade secrets. Some cases have held that employment in and of itself establishes a confidential relationship which prohibits disclosure of trade secrets even where there is no written employment agreement. In the second situation, the trade secret is disclosed in confidence and for evaluation only by one company to another during unsuccessful negotiations over a contract which is never made and the company receiving the trade secret later makes unauthorized use of it. In both situations the courts have often entered injunctions barring wrongful use of disclosure of the trade secret.

In the personal computing field, we're beginning to see software vendors try to bind users to non-disclosure by restrictive language appearing in advertising and sales invoices, purporting to limit the user's freedom of action with respect to the purchased software. While I am unaware of any cases construing such provisions in the personal/small business computing field, I think the provisions should be respected at least to bar commercial exploitation of the software purchased under those terms. Whether a non-commercial user's group would be held liable is a close question, although I think it probably would be, perhaps more likely on copyright theories than on contract theories.

CONTU also mentioned the theory of unfair competition as it is defined under both federal and state law.[12] Broadly, this form of protection prohibits false advertising, false designation of the origin of goods or services and the "passing off" of anothers work as one's own. This area is a branch of commercial law, akin to trademark law, and is not particularly well suited to software protection except perhaps in those cases of flagrant plagerism and distribution of pirated works.

There are stern laws today denouncing the theft of computer programs and the like which impose rather severe criminal as well as civil penalties.[13] I think we'll be seeing more criminal prosecutions for the theft of software and the misuse of computers in the future as this technology becomes more widespread.

In conclusion, there is a need today for legal protection of software which is a step beyond that afforded by copyright and a step short of that available through patents. If you agree with me, I suggest you write to your Congressmen and let them know your views. I have. The need for an appropriate form of legal protection for software is recognized by everyone. It is only the precise contours of that protection upon which we disagree and which remain unresolved.

Footnotes

[1] Parker Acting Commissioner v. Flook, ____ U.S. ____, 198 U.S.P.Q. 193 (1978).

[2] Id., 198 U.S.P.Q. at 195-196.

[3] Id., 198 U.S.P.Q. at 196, Footnote 7: "The term 'software' is used in the industry to describe computer programs. The value of computer programs in use in the United States in 1976 was placed at $43.1 billion,

and projected at $70.7 billion by 1980 according to one industry estimate . . . ."

[4]Statistical Abstract of the United States for 1977, p. 428.

[5]Parker, supra, 198 U.S.P.Q. at 197 quoting from Gottschalk v. Benson, 409 U.S. 63, 67, 175 U.S.P.Q. 673, 674-675 (1972).

[6]Mackay Radio and Telegraph Co. v. Radio Corporation of America, 306 U.S. 86, 40 U.S.P.Q. 198, 205 (1939): "Carter [the patentee] avoided prior art by defining his angle for antennae with wires of particular wavelengths with mathematical precision, cannot discard that precision to establish infringement."

[7]Parker, supra, 198 U.S.P.Q. at 199.

[8]Final Report of the National Commission on New Technological Uses of Copyrighted Works, July 31, 1978, Copyright Law Reports, Extra Edition No. 2, August 31, 1978, Commerce Clearing House, Inc., 4025 W. Peterson Avenue, Chicago, Illinois 60646 (hereinafter "Final Report").

[9]P.L. No. 94-553 (1976) Codified as 17 U.S.C. §101 et.seq.

[10]Final Report, pp. 26-27, CCH edition.

[11]Id., pp. 42-44.

[12]Id., pp. 44-45.

[13]17 U.S.C. §506(a) "Criminal Infringement.--Any person who infringes a copyright willfully and for purposes of commercial advantage or private financial gain shall be fined not more than $10,000 or imprisoned for not more than one year, or both . . . ." California Penal Code §499c(b) (a person is guilty of theft for appropriating another's trade secret under stated circumstances).

INFRINGEMENT AND LICENSING OF PROPRIETARY PROPERTY

Sheldon R. Meyer, Attorney at Law
Phillips, Moore, Weissenberger, Lempio & Majestic
Three Embarcadero Center, Suite 2900
San Francisco, California 94111
Tel: (415) 421-2674

I. General Patent Licensing and Assignment Considerations
- A. The patentee has the right to exclude others from making, using or selling the invention throughout the United States (35 USC 154).
- B. A license to practice an invention may be of little value as the patentee himself may not be able to practice his own invention without infringing the invention of another.
  1. Pioneer and improvement patents and examples.
  2. Cross-licenses.
- C. What policy consideration should a company make before acquiring a license to a patent?
- D. Investigation by the licensee into the strength of a patent to be licensed.
  1. File wrapper history.
  2. Validity and right-to-use opinions.
- E. The general format of a license and an assignment.
  1. Rights and duties of the licensor/owner and the licensee.
  2. Exclusive versus non-exclusive licenses.
- F. Royalty Structure.
  1. Royalty rate
  2. Royalty Base including unpatented components.
  3. Package Licenses.
  4. Post-expiration royalties.
  5. Royalty structure when a combination of patents, know-how and trade secrets are licensed.
- G. Patent misuse and anti-trust consideration.
- H. Territorial and use restrictions.
- I. Arbitration causes.
- J. The rights of joint owners of patent (35 USC 262)

II. Patent Infringement Considerations
- A. Protecting a marketing position and which infringer to sue.
- B. Types of infringement (35 USC 271)
  1. Direct Infringement
  2. Inducing Infringement
  3. Contributory Infringement
- C. Where can an infringer be sued?
  1. Federal Court Jurisdiction (28 USC 1338)
  2. Commencement of a suit where the infringer resides or where the infringer has committed acts of infringement and has a regular and established place of business (28 USC 1400)
- D. Suing the patentee for invalidity.
  1. Declaratory Judgments (28 USC 2201 and 2201, 28 USC 1391)
  2. What not to say at trade shows and to your competitors customers so as to avoid being sued for patent invalidity.
- E. Patent marking and mis-marking (35 USC 287 and 292)
- F. Injunctions and damages (35 USC 281, 283, 284, and 285)
- G. Statute of Limitations (35 USC 286); estopped and latches.

TRADEMARKS AND SERVICE MARKS AS MODERN GOODWILL AND AS FRANCHISABLE PROPERTIES

HUBERT E. DUBB, PATENT AND TRADEMARK ATTORNEY
PHILLIPS, MOORE, WEISSENBERGER, LEMPIO & MAJESTIC
Three Embarcadero Center, Twenty-Ninth Floor
San Francisco, California 94111

Abstract

Computer products and services are often offered on a national or even international basis. Such concepts as asking a neighbor, e.g., where the best meat market is located, are not applicable in such a situation. Instead, the intelligent buyer seeks out a name, called a trademark or service mark, which identifies a product or service as being of an expected quality, and then seeks an outlet which sells the trademarked goods or offers the service marked services. Trademarks and service marks thus constitute a form of national or international goodwill. How important are such marks? Well, what kind of film do you buy and who has the best reputation for film processing?

The requirements for selecting a good mark, properly controlling products and/or services supplied under the mark, and enforcing the mark against infringers, are discussed. Uses of the mark as leverage in expanding existing and prospective businesses through licensing and franchising is also discussed, with particular emphasis on the degree of control of the mark which must be maintained by licensors/franchisors. Some possible problems which might arise under the anti-trust laws are briefly highlighted.

Introduction

Trademarks are the names that merchants attach to their goods while service marks are the names that merchants attach to their services. For example, Kodak is a trademark for cameras, films, photographic chemicals, and photographic supplies generally. Kodak is also a service mark for the service of processing films into photographs, slides and the like. Moving closer to the computer field, IBM, besides being a company name (synonymous with tradename), is used as a trademark on computers and computer related products. IBM is also used as a service mark for repair services for computers, programming services, educational services and the like. In the home computer field the term Byte Shop is both a trademark for computers and computer products and a service mark for retail store service, repair services and the like. To shorten matters, I will simply use the term "mark" to refer to both trademarks and service marks.

The mark of a merchant is perhaps his most valuable asset. By turning out products of a particular quality or providing service of a particular quality, a merchant acquires a reputation for such quality. It should be noted that this quality is not necessarily excellent or even good - but it is expectable. You may like McDonald's hamburgers or hate them, but you know that they will taste the same whether you buy them in Bangor, Maine or Lahaina, Maui. This reputation leads the buying public to return to the merchant if they are happy with him and possibly to recommend this merchant to other members of the buying public. This can

significantly affect the amount of business done by the merchant. Such a business reputation, and the accompanying tendency for the buying public to frequent the merchant, and to recommend that others frequent him, are known as goodwill.

Registration

Provisions have been made by each of the states, by the Federal Government, and by almost all foreign countries to allow a merchant to protect his goodwill through registration of his mark or marks. The reason that such laws have been enacted has not been principally to aid the merchant, but rather, to protect the consumer. That is, the theory behind such laws is that when a consumer sees a particular mark associated with a product or service, he has a right to identify that product or service with a particular merchant in whom he may have developed some trust. Thus, there are some very strict regulations governing the licensing out of marks, which regulations are designed to assure that the owner of the mark maintains sufficient control over the quality of the goods or services, that the consumer is not deceived as to the nature of such quality.

Because marks are protectable mainly to aid the consumer, one may only acquire right in these marks through use of the marks in conjunction with particular goods or services. In other words, the trademark statute (the Lanham Act in the United States) provides merely a registration of existing rights, which rights are created by the sale of goods or services with the mark used in conjunction therewith. Even though the Lanham Act registration is only a registration of a pre-existing right, such registration provides very substantial procedural advantages to the owner of the registered mark. For For example, in the case of an unregistered mark, in a suit against an infringer, the first user must show that the mark has acquired "secondary meaning". Secondary meaning is an understanding, in the mind of the buying public, that the mark when used in conjunction with the goods or services, conotes a particular source (the first user) for the goods or services. Secondary meaning does not, normally, have to be proved for a registered mark. In both cases (registered and unregistered mark), the first user must prove that there is a likelihood of confusion between his mark and that of the purported infringer.

There are certain bookkeeping procedures that should be followed if one wants to obtain a trademark (or service mark) registration. One thing which must form a part of each trademark (or service mark) application for registration is a recitation of the date of first use of the mark with the product or services and the date of first use of the mark with the product or services in commerce. The term, in commerce, as used in the Lanham Act, refers to interstate commerce or international commerce. Thus, good records should be kept of such dates. Further, it is advisable to keep proof of these dates, since such proof may at times become necessary during litigation or during opposition proceedings in the United States Patent and Trademark Office.

Choosing Marks

Different marks have different strengths. That is, some marks are given broader protection by the Courts, than other marks. As a general rule, one cannot register a mark for a merely descriptive or generic phrase or word. Thus, the word "cup" is not registrable for cups. Further, a Court could certainly not enforce such a mark against any other party. This is because everyone has

the right to use descriptive terms or generic terms to describe their product. Generic terms, by the way, are terms which may have originally been usable as trademarks, i.e., may have originally been strong and non-descriptive, but which have, through extensive uses by a number of persons, become recognized as being descriptive of a particular product. Aspirin, which was originally a trademark of Bayer, is a prime example of a mark which has become generic for a product. A strong mark is a mark which is in no way descriptive of the product or services which it covers and which is not similar to any other mark in use. Kodak, for example, is a quite strong mark. It is completely non-descriptive, arbitrary and coined. Other marks fall some place between the coined and the descriptive. These marks may be somewhat suggestive of the goods, or the quality thereof, but are not so clearly descriptive as to be unregisterable.

Since a mark is very important to the merchant, it is highly desirable that a mark be chosen at the outset, which is not descriptive and not similar to any mark used by another party, particularly in the same field of endeavor. A trademark search helps to minimize the chance of choosing a mark which is confusingly similar to the mark of another.

Protecting Marks

Once a mark has been selected, it must be carefully protected by being prominently utilized in conjunction with the goods or services, by making sure that no one else adopts and uses a similar mark through promptly demanding that such infringement stops and promptly instituting suit if the demand is ignored, and by being sure that whenever the mark is used by the merchant or his licensees, it is used in a trademark sense and not in a generic sense. That is, the mark when used should be capitalized, used in distinctive type, properly marked as a trademark, for example with an ® to show that the mark is registered if it has been registered or with a small ™ if it has not been registered. It is also advisable to have descriptive material with the product or services so that the mark does not begin to be used descriptively and thereby degenerate into a generic term.

Foreign Registrations

If a merchant contemplates selling his products or services outside of the United States, consideration should be given to registering the mark in the particular foreign countries in which the merchant reasonably contemplates operating. The laws of the various foreign countries differ greatly from one another, and, there is a distinct split between those countries which require use of the mark in the country prior to registration, and those countries which grant registration to whomever is first to file on the mark in the country. In the case of the latter countries, particularly, it is important to file as soon as possible so as to make sure that someone else doesn't rush into the foreign Trademark Office and register your mark. This is a relatively common practice in many of these countries. There is generally an advantage in filing in foreign countries within six months of your filing date in the United States so as to obtain some very substantial advantages as to your effective filing date in the foreign countries. Under present law, if one files within six months of one's United States filing date in a foreign country, and claims priority of the United States application, then the foreign country treats the application filed in that country as though it had been

filed in that country on the same date that it was actually filed in the United States. Sometimes this can help you get behind the filing date of someone trying to pirate your mark.

Infringing Imports

It is not necessary to have registered your trademark in all foreign countries to keep your product or services from being sold in the United States under that mark. In fact, it is quite possible to get the United States Customs Department to stop import of a product which has your mark, or a mark confusingly similar thereto, on it and is being shipped into the United States from a foreign country. This is a relatively quick procedure but requires the posting of a bond.

Licensing and Franchising

Marks can be used as leverage in expanding existing businesses through licensing or franchising. In other words, the merchant may develop a valuable mark within a relatively limited area, for example, two or three states. He may well not have sufficient funds, or may simply not have the desire, to expand his business to the remaining forty-seven states. It is quite possible for the merchant to become a franchisor and to give exclusive or non-exclusive territories to various franchisees, wherein the franchisees are allowed to utilize the mark with their goods and/or services. Or, the merchant may simply license use of the mark to others for a fee. In such an instance, it is imperative that the franchisor/licensor maintain full control over the quality of the goods and services. If such control is not maintained, the franchisor's right therein can be lost. For example, the franchisor must maintain the right to inspect the goods and services for quality. He must maintain the right to have any particular goods or services pulled off the market if they do not satisfy his quality standards. He should insist that the franchisees agree that any use of the mark accrues only to the benefit of the franchisor. He must have the franchisees agree to desist from any use of the mark at the end of the franchise agreement. Basically, the franchisor must exercise such complete control over the use of the mark, that the public can be sure that any product or service sold under the mark, will have uniform quality throughout the United States.

Anti-Trust Considerations

There are certain things that the franchisor should not and cannot do. First, he cannot require his franchisee to buy supplies from him, if such supplies are available elsewhere and are not subject to coverage by any patent which the franchisor may hold. Second, the franchisor may not set the sales price to be charged by the franchisee to the public. Third, the franchisor must be very careful to maintain fair dealings with the franchisee, especially since the franchisor will probably be in direct competition with his franchisees at the retail sale level. That is, in our example, the franchisor will generally have his original stores in his three states, and, these stores will be in direct competition with the stores owned by his franchisees in the other forty-seven states. This puts the parties on an equal or horizontal basis as such is referred by the Courts in interpreting under the anti-trust statutes. It should also be mentioned that many states now require that franchise agreements be approved by them prior to the sale of any franchises within those states. California is one of these states. Thus, the prospective franchisor must carefully determine the government regulations, both federal and state, under which he must operate and must follow these regulations with care.

Summary

To summarize, trademarks and service marks

are extremely powerful and useful marketing tools and, thus, constitute valuable property rights. The acquisition of these rights is governed by both statute and common law rules. Like all property rights, trademark and service mark rights may be lost if they are improperly managed and, hence, require careful and constant control and policing.

BUSINESS MICROCOMPUTERS: FRAUD OR REALITY?

Rodnay Zaks
Sybex, Inc.
2020 Milvia St.
Berkeley, CA 94704

Business Microcomputers: fraud or reality?

Microcomputers have been widely advertised as being applicable to small businesses. "For the first time, hardware is available for less than 10,000 dollars which is fully capable of providing the functions required for the actual automation of a small business." Is this statement true or false? Literally, it is correct. Hardware systems are available which could provide this capability at this cost. The essential question is: does it really provide the capability? It is the purpose of this article to address this essential question.

From Home to Business Computers

Just like microprocessors found a market by accident in 1972, microcomputers, which had been originally designed for the industrial market, found their next market by accident: the home computing market. An article by Leslie Solomon, in the January 1975 issue of Popular Electronics revealed the existence of a low cost microcomputer available to hobbyists. The name, selected by the editor's daughter during a Star-Trek episode was the "Altair." This microcomputer is said to have been residing at the time in the trunk of the designer's car at the small Albequerque company called MITS. This journalistic fluke (or foresight) suddenly revealed the unforeseen existence of a huge new market which would become the home computing market. Within a few weeks, MITS was deluged with orders for this microcomputer, and a new standard was born. It is now known as the S100 bus, the set of conventions adopted for the internal mother-board bus inside the microcomputer. The huge demand created a sellers market, where no delay was too long for the enthusiasm of the early hobbyist. The next three years were marked by the emergence of a new race of human beings: the computer hobbyists. As this race is now believed to be quickly approaching extinction, the microcomputer industry is cheerfully predicting the emergence of a huge new untapped market: business applications. The newly found power of low cost microcomputer systems could now be applied to solve the ills and management problems of all small businesses in America. Myth or reality?

Let us first address the real needs of a business and then examine the available resources before we reach a conclusion.

The requirements of business computing. At a minimum, automating any small business requires the availability of specialized files and file management programs for accounts receivable, accounts payable, payroll, general ledger, inventory, tax, bank accounts, sales reports, and other journals and ledgers. As a first step, the computerized automation of any of these files or ledgers is a welcome benefit. The results are improved accuracy, availability of reports and statistics, as well as (sometimes) reduced manpower. However, the real benefits of a computerized business system are accrued only if file management is completely automated. This means that whenever a transaction is performed, all necessary programs or sub-systems must be activated automatically in the proper sequence without having to re-enter data in succession. As an example, in a simple system, whenever a sales order is recorded, all data will be typed in, such as the name of the customer, the nature of the order and address and delivery instructions. This data can already be used in a first step to build a mailing list. In the second step, an invoice should automatically be generated, and the accounts receivable journal should be updated.

In addition, the inventory file should be checked for the availability of the items ordered. Whenever the stock for an item reaches a predetermined threshold, an automatic re-order should be activated. Whenever an item is not available in stock, a back order should be generated. A shipping list or packing list should automatically be generated together with a shipping label.

As a result of a single order, a number of files will have to be updated automatically, and several programs will have to be activated in sequence. The true benefit of a computerized system will accrue only if such automation is implemented.

Second, every small business tends to have different requirements. As an example, an inventory program may offer a large number of attributes per item such as size, weight, dimensions, item number, items in stock, minimum quantity for re-order, purchase price, sales price, date entered in stock, other items necessary with this one, etc. In order to be truly useful or rather usable to a large number of potential users, this inventory structure will specify a much larger number of entries than the number realistically usable by any single business. As a result, a relatively small number of inventory items will reside on any diskette. In addition, the data entry phase for any item in the inventory might be much longer than might be otherwise required. This simplified example shows the disadvantages of a standard program. There is unfortunately no such thing as standard business computerization techniques. In order to derive the true benefits of a computerized business system, every manager or executive shall have to implement a truly customized version of these programs. Flexibility is also essential in a business environment. Initially the needs of the business might be served by a number of simple software packages performing the traditional functions. However, it might quickly become desirable to add other customized routines to this set. Unless the competence exists in-house and all packages being utilized are fully documented, the task necessary to add the necessary additional facilities might become prohibitive.

In summary, the requirements of the small business are technically best served by a highly complex set of programs customized for the specific business. Clearly, this approach is not realistic in the case of microcomputers yet, in view of the general unavailability of sophisticated software, and of the very high cost of programming relative to the cost of the hardware. Limitations in the value of the business programs will therefore exist and will be evaluated throughout the remainder of this discussion.

In addition specific hardware requirements exist to meet the needs of such file systems and will be examined now.

The hardware. Every microcomputer system requires first a box containing the microcomputer itself, i.e. the microprocessor board, the memory boards, any required interface boards, plus a power supply. In addition, the system requires a business quality printer, a CRT terminal, and a disk system for mass storage. The requirements of each of these elements will now be examined. The microcomputer itself often appears as the crucial choice in the selection of a business sytem. It is probably the least important one. The speed of the microprocessor itself is almost irrelevant. Because nearly all business systems are implemented in a high level language, the efficiency of the software interpreter or compiler which is used to execute this high level language is the item of crucial importance for the efficiency of the system. Although it can be said in a simplified manner that the speed of different microprocessors varies in the ratio of perhaps 1 to 5, the speed of existing BASIC interpreters, for example, may vary by a factor of 1 to 50. Software efficiency is the key to speed. There are naturally advantages and disadvantages inherent to each microprocessor. For example, in order to enjoy the possible benefits of standardized boards, any system providing a S100 bus offers an advantage. It requires in turn an 8080 or a Z80 microprocessor. However, provided that the sufficient set of peripherals be available from the start, the option to be able to add new fancy boards may be more appealing to the hobbyist than the business person, and other busses than S100 might be equally acceptable. The choice of this beautiful microcomputer box may therefore be based on the established reputation of the manufacturer, its assumed reliability, or the possible advantages of its bus structure. Other arguments are going to be presented in the remainder of this section.

The hardware items which may have the most important significance for the businessman are by far the peripherals. First the printer. In order to provide printing speed compatible with the printing of lengthy business documents, a 60 characters per second speed is generally assumed to be a minimum. In order to also provide adequate printing quality, daisy-wheel printers are a preferred choice. Unfortunately their cost is in the 3,000 to 5,000 dollar range. However, they may be the current optimum in business printers. One step below daisy-wheel printers may be traditional impact printers or matrix printers, available at half the cost but which are characterized by either a slow speed or else poor printing quality. They are often a

reasonable choice in a first step of business computerization as long as no business forms are required, or as long as no lengthy files need to be printed in a short time. However, it must be remembered that as it has always been the case, the cost of the peripherals will usually be the dominant cost in a system. Peripherals are likely to be usable over a significant period of time whereas the microcomputer main-frame is likely to be obsolete in a short amount of time. It might be more valuable to invest time in the correct selection of the long lived expensive peripherals than in the selection of the main frame. The second most expensive item is the disk file. Mini floppies have been the most successful mass storage medium for hobby microcomputers. They can simply not be considered for serious business applications. They are too slow and offer too limited a capacity. Regular floppy disks offer a marginally acceptable capacity and a marginally acceptable speed. They are clearly inadequate for efficient business operation. Unfortunately, for the time being, there is no low-cost alternative. From a technical standpoint the hard disk is a clear optimum. New "Winchester"type disks are now becoming available which might result in the availability of hard disk facilities in the near future at reduced cost. For the time being, floppy disks must be considered as the minimum facility with which a business system should be equipped as of now. By the way, when one speaks about floppy disks, there must always be at least two disks, so that copies can be created or files merged. In the life of a system it is likely that two disks will become three, four or more disks as time goes by. The cost of such dual floppies and the required controller board is likely to be the second largest cost in the purchase of the system.

The last significant terminal is the CRT terminal. A CRT terminal is a CRT display equipped with a keyboard input. For efficient business use, the consensus is that minimal specifications for the CRT display are a 12" diagonal dimension, and 24 lines of 80 characters, upper and lower case. The keyboard should be of good mechanical quality and should be a full business-type keyboard with separate keys for digits and for control functions on the screen such as scrolling (moving lines up or down) addressing the cursor and moving it. Although new low cost CRT terminals have become available, the reliability compatible with business use still requires a significant expenditure.

Finally, the memory of the system should have a minimum size of 32K bytes, and preferably 48K or more. The memory is no longer expensive, and is usually one of the least expensive components of the system.

Having now presented the minimum configuration of required hardware, let us address one of the most significant topics: software.

The Software.

The software refers to all the programs necessary to make efficient use of the set of hardware resources we have just described. At this time, no complete business software facility exists for microcomputers!

Partial implementations exist and a number of simple packages are now available which will perform, usually separately, payroll, accounts receivable, general ledger, and other functions. However, the crucial task of simultaneous file management and sequential activation of selected programs is, as yet, not implemented. Such software solves business problems individually, but does not provide the comprehensive facility needed for the efficient use of the hardware resources. As an example, the user might have to enter a sale transaction by typing all the data. He might next have to re-enter the data relative to the items being purchased in order to check his inventory. Other data might still have to be re-entered to update the accounts receivable, while other data may still have to be re-entered to update other files or activate other programs. Good comprehensive software is simply not yet available. As a result, all these beautiful hardware microcomputers do have the capability of solving all the business problems that are advertised. In practice, they cannot. Not yet.

Is this a fraud? Current software available for microcomputers makes them capable of solving a large number of tasks commonly associated with business accounting and bookkeeping. Are these benefits worthwhile? Because of the limitation in the automatic file handling capability of most of these programs, the computerization of these tasks may not result in any savings in terms of personnel. The entry of data for computer use tends to be longer and more complex than the manual typing of invoices or filling out of conventional forms. This is because a number of extra fields are required, and the entry format is highly structured. As a result in most small companies, computerization might require somewhat more manpower than less.

In addition, the possible unreliability of hardware and software components might result in catastrophic system breakdown. Every small business owner will fully realize the consequences of having a computer "down" at the time that payroll checks should be generated, especially when the data needed has been saved on a single disk file which has been wiped out because of " accidental " error.

These drawbacks are real. However, let us consider now the real advantages within the acknowledged limitations and deficiencies of existing systems.

The Real Advantages

The real value of contemporary microcomputer systems with their limited software lies in two areas: management education, and future savings. Let us explore these two points.

Every user of new and complex machinery must spend a significant period of time to learn the skills necessary to evaluate it and control it. As an example, any driver must spend an appreciable amount of time learning how to control a car. There is therefore considered highly advisable to practice the used or the low cost car rather than the expensive new one the first time around. The same is applicable to microcomputers. With the introduction of computers in a business, a phenomenon known as computer shock occurs. The radical change of procedures required by computer programs often causes personnel to leave, rebel, or otherwise lose their efficiency. Similarly, catastrophic initial failures are likely to occur in the form of data being wiped out or not being produced at the right time. However, because of the limited cost of microcomputers today, a heretofore unknown opportunity exists for the business owner to familiarize himself and his employees at minimal cost with this new technique. The business owner can now learn to "drive his own computer" and still derive some business benefits from it. By learning to use individual business packages, he will familiarize himself and his employees with the requirements of computerization. After a period of one or more years, he will become familiar with the file structures that his specific business truly requires as well as with the hardware capabilities that his next computer system should really have. In short, microcomputers offer for the first time the opportunity to learn computers by doing. The businessman will learn the specific technical capabilities that his next system should have by practicing on the first one. In addition, there is a possible financial advantage if some business benefits occur. After the initial loss of efficiency traditionally associated with the introduction of the computer, a significant amount of efficiency might be regained if the business expands and if repetitive operations occur. Once computerized, one subsystem is generally capable of handling a very large number of transactions with the same pattern. Less skilled employees then can be used, or rather any single employee should be able to handle a lot more business since the processing of data itself would be done automatically. In addition, at the cost of hiring a programmer, this business should have the capability of finally adapting the business packages to the specific requirements later and possibly completing the initial business packages with the required file automation capabilities it needs. Provided that this additional software investment is made, the system might become a truly functional highly efficient business automation system. However, in such a case a software investment is likely to be many times more significant than the hardware investment.

In summary, microcomputers today offer the capability to learn business computerization at a modest cost. In addition, they have the potential in specific situations to bring modest or sometimes significant savings in the case of business expansion. Finally they may be able to supply business capabilities which were simply not existent before. Specific examples are management reports, sales analyses or statistics which by themselves may have a value superior to the entire purchase cost of the system. A good mail list system might for example have a marketing value far superior to the several thousand dollars that the system did cost. For these reasons, current microcomputers are likely to pay for themselves several times over in direct business benefits as well as education for the business owner. They are far from having attained yet the true business automation capabilities which larger computers have demonstrated so far, and should not be presented as such. Business microcomputers are a reality. The realistic evaluation of their limitations is also a necessity.

THE ECONOMICS OF PURCHASING A SMALL COMPUTER

Casimir C. "Casey" Klimasauskas
295 N. Los Robles #305, Pasadena, CA 91101
(213) 440-1585

Abstract

This paper describes a systematic methodology for establishing the cost effectiveness of converting various manual systems to a small computer system. It attempts to point out the usual pitfalls, and suggests procedures which minimize the chances of failure.

Introduction

For nearly a decade, I have watched computers develop from the massive transistorized monsters of the mid-60's into the compact elegant micro-processors of today. In the process, I have been intrigued to observe that one thing we learn about history is that we don't learn anything from history. I hope that today, what I have to say will be a basis upon which some of you can successfully build, avoiding the repetition of history.

This presentation is primarily aimed at the small businessman who may or may not have any experience with data-processing. Even so, the programmer, systems designer, and hardware vendor may benefit from an overview of the process involved in successfully choosing and implementing a data-processing system.

What does it take? The basic process is relatively simple.

1. Define the applications. Identify potential areas where the computer may help you be more effective.

2. Describe the present system. Discover what you are doing NOW and WHY.

3. Calculate the present cost. Calculate what it is costing you to do it on a per unit basis.

4. Make future projections. Define where you would like to be in 2 to 5 years.

5. Define your requirements. Estimate the system specifications.

6. Shop around. Select two or three systems - manual and/or computer - which might do the job.

7. Calculate the costs/benefits. Calculate the costs - initial and ongoing.

8. Choose a system.

Just a note, when I refer to a "system," I am referring to any combination of equipment, forms, procedures and personnel required to accomplish an isolatable task in coordination with other tasks with which it must interface. When I refer to "computer," I am referring specifically to the electronic gadgetry used to manipulate data.

This process does not end with the choice of a system. At that point, the process is barely begun. The next, and perhaps most crucial phase is the implementation of the system. That is the topic of another paper.

Let's explore each of these areas indepth.

I. Define the Application

Computers cannot, contrary to popular belief, do all things. In fact, they are limited in their effectiveness to a relatively small number of functions. That brings us to the first common misconception. It may seem trite, but its psychological impact on management and employees cannot be underestimated.

FALLACY #1 - A company is "computerized." False. Specific tasks/functions within a company are facilitated through the use of electronic data-processing equipment.

You will still need people. At the heart of every successful business is making the people who work for you more effective through the use of capital invested in equipment. The same is true for computers. They, in and of themselves do not replace people. They facilitate the performance of tasks by individuals in the company, increasing their productivity. In many instances, the computer will free the present employees from tedious and time-consuming tasks to be more effective in the areas they excell, in the tasks for which they were hired.

Again, computers do not make decisions, people do. The subtle danger in talking about computerizing a company is that management begins to expect that it can abrogate its responsibility for making decisions, for taking the risk and responsibility for the future profitability of the operation. They blame the computer for a drop in sales, product quality or loss of profitability. The computer must be seen, just as any other investment, as a tool to make the individuals in the company more effective.

What CAN computers do?

The strength of a computer lies in that it is capable of performing a pre-defined set of operations on specific data with speed and precision.

In practical terms, this means that a computer is particularly well suited to tasks which:

1) Are well defined and
2) Involve a) the transcription of large quantities of information, or b) calculations.

The most successful small business applications of computers have been in the areas of subscription/name list management, inventory control, accounting, billing/invoicing/receipting. Recently, word-processing, data-collection and electronic mail have been growing areas of application.

More sophisticated applications include scheduling, cost analysis, modeling and projections, and marketing analysis. These in general require a very sophisticated user working with a team of specialists who can interpret and explain the results, and their limitations.

II. Describe the Present Situation

People are like frogs. In junior high school biology there is an experiment in which the students take a frog in a pan of water. Slowly, the water is heated until over the period of the class, the frog is cooked alive. And it never even noticed.

Procedures are like that. When first written, they may have been followed religiously, but with time, little changes here and there take place. Individuals do things differently. When personnel changes take place, even in large, well-supervised companies, procedures change unnoticed. Before you know it, your frog is cooked. That brings us to

FALLACY #2 - We know what we are doing. Don't bet on it.

Having identified the areas in which the computer may have application, take a good hard look at them. It is sometimes good to have someone from the outside who knows nothing about YOUR particular operation take a look at what you are doing. This should start with a session with the company management to discover what management thinks is being done, the reasons why, and the methods. With the long-range perspective of management, the analyst can sit down with the personnel actually involved in the day-to-day operation of the present system and discover what they are doing. Finally, the analyst can take a look at the procedure manuals to see what they were supposed to do. Freedom from the preconceived notions of what is/has been allows objectivity in describing the present situation.

At this point, there are three things which it is crucial to establish:

1) How decisions about various situations are made.
2) Problems from the past, and how they have been handled.
3) The ultimate use/purpose for which the the task is being performed.

Unless the process of how decisions are made at each step of the present system can be unambiguously defined, it will be impossible to computerize the system. An extension of the second law of thermodynamics (the universe is tending toward chaos) is that every system, given sufficient time, will develop more exceptions than rules. Understanding, and having on paper the actual processes and exceptions will facilitate the design of a more comprehensive system in which there are no exceptions. The essence of efficient productive systems is a hard line on exceptions.

Problems of the past are in reality situations which the original system designer failed to take into account. If they occur more than once, they are usually institutionalized as exceptions. Recurring problems may indicate that the reasons for the original system have considerably changed. It may be a dinosoar bewildered in the concrete jungle of the present.

Why are we doing this? That question isn't asked enough. Without a clear understanding of the purpose or goal of a specific task, it is impossible to redesign the system in terms of a computer. There is no basis for making a decision about how a new problem is to be handled. At best, the computer can mimic or imitate the present system. In that case, it is unlikely that it will make the people involved any more productive. Nor will it solve the current problems or exceptions. It will simply be trading salary dollars for equipment dollars. And even at that, it may not be a trade, but a sum.

Let me illustrate this. In the area of subscription lists, the Addressograph plate system was very popular until recently. (In fact, several companies continue to use it with mailing lists of several hundred thousand names.) In attempting to make the switch to computers, the easiest thing was to create a program which let the user continue to view the world as if it were made of Addressograph plates. These programs are often referred to as "laundry list" programs. One company I worked for had a relatively sophisticated, though outwardly simple looking system which they wanted to computerize. Two "laundry list" type service bureaus tried and failed. Their programs could mimic the manual system, but could not fulfill the marketing demands of the system.

Once the system as it exists has been thoroughly described, and written up, it is time to sit down and place a value on each of the outputs of the system. What is each worth? What do you really need to know? What are you willing to pay for it? Why is this important? This begins to establish a context in which the value of various tradeoffs can be compared to the cost of implementing them.

You are now ready to begin collecting some hard data.

III. Calculate the Present Cost

This is something which is often overlooked. A basic underlying assumption upon which many small businesses seem to operate is that computers are always cheaper than manual systems.

FALLACY #3 - Computers are cheaper than manual systems. In the right situation, with the right equipment, and the right software in an integrated system, maybe yes.

Particularly if the wrong computer/system is chosen, it can cost far more than the corresponding manual system. For a while, I worked swing shift as an IBM S/3 operator for a company which thought that a computer would save them money. They hadn't done their homework. Two years later, they sold the machine and went back to their old service bureau (I believe). The wrong system spelled disaster.

Crosby has said that "the only practical management objective is improvement." This applies to computers, manual systems, in fact every aspect of a business. The only way that is is possible to measure improvement is by having a thoroughly established baseline from which to measure the impact of changes.

Another reason for calculating cost is that the ultimate goal of a profit-making industry is to optimize the productivity of every dollar placed into the company - from sales or by investors. Unless a base-line is established, there is no way to measure return. So, calculate the cost.

In most operations, there are three kinds of costs involved:

1) capital or initial set-up costs.
2) run set-up costs.
3) per unit costs.

The advantage of computers is that, though the initial capital outlay is large, the per unit costs are very low. Hence, the overall unit cost is less than in similar situations. Manual systems on the otherhand have the advantage that they usually require little initial out-lay though they have considerably higher per unit costs.

The net result of this is that computers tend to be more effective in high volume processes, while manual systems are more effective in low volume processes. In any case, let's look at what it is presently costing to do the job.

The most crucial aspect of costing is to make sure that everything is included.

As the processes involved become more efficient, and the individuals more productive, it may be possible to reallocate some of them to other tasks. With that in mind, it is essential that overhead which is allocated against each individual be taken into account. So, included in the hourly rate of the individual should be benefits, managerial overhead, utilities, insurance, and any other services which the company does not directly allocate to its products/processes. Secondly, it is important to include such things as an allocation for floorspace and interdepartmental services. These may seem like small things, but a lot of small things can add up to quite a bit of money - 10% to 30% of the total costs! Note, if these items have already been included in overhead applied to the employee, it may be well to break parts of them out, especially if they may change with the computer system.

Third, establish a useful unit of production. This might be, in the case of accounting, the number of journal entries completed, or the number of invoices created, etc.

Fourth, establish/estimate the total unit cost - everything included.

Finally, estimate what it would cost, at today's prices to set up the system as it now stands.

From this you have a basis for understanding the capital costs involved in the present system, as well as the unit costs (amortizing the per-run setup across the unit costs).

IV. Make Future Projections

Choosing a system is like marriage - you're stuck with if for the forseeable future (though that used to be more true than it is today.) So choose with caution, and with an eye to the future. Something which just fits today, may be too small tomorrow. And if expansion can only occur in quantum leaps, that can spell disaster.

FALLACY #4 - "Our system is continuously upward expandable." That may be true, but it is unlikely that expansion will be anything like continuous. It will cost in significant chunks of money.

Furthermore, if the origianl system was not designed and chosen so that it could expand, it may require the conversion to an entirely new and different system. That is an unthinkably costly option.

FALLACY #5 - This is only an interim system. I have never in my experience seen an "interim" system. Systems always become permanent until forced to change.

As you choose, it must be with the awareness and committment that this system is forever, and then set up regular times to evaluate. Another danger in approaching this in a fickle or flirting manner is that you will fail to learn from your mistakes. All systems, regardless of how well designed, will have flaws. Any system that you implement should include a monthly evaluation of the effectiveness of the system for six months after installation, and every six months for the year beyond that. In those evaluation periods, it is possible to iron out the new problems/situations which have occurred. Continuous change invites a host of new difficulties, which will never be solved.

All of that to say that it is crucial to think about the future. And to plan with a solid five year committment to whatever system

is chosen.

What should you include in thinking about the future? Projected product volume, new government regulations/reporting requirements which you might foresee, anticipated expansion/termination of all or part of the services/products involved, managerial reporting which is beyond the present capability of the system (or in some cases, the elimination of certain kinds of managerial reporting), marketing potential of the information contained in the processes involved, and so on.

Write all of these things down, and attempt to estimate a value to you and the company.

With the growing rate of change and uncertainty, it is hardly possible to think in detail about much more than the next five years, and in many cases, more than two years ahead. However, do it for as far ahead as possible.

V. Define Your Requirements

This step is a way of formalizing everything which has gone on to this point. It integrates the various areas in which the computer might be used into a composite, and defines the acceptable parameters in which it must operate.

Here are some things to consider:

* How much storage will each of the items take? What is the total storage requirements?
* How many people will need to access the information at a single time? At any time?
* What does the response need to be? How long should each person be willing to wait?
* How reliable must the system be? Can I afford to be without the computer for 1 day? 1 week?
* How much printing will I need to do? 20,000 names weekly? 100 lines a day?
* How will I backup the information? What will I do in the event of catastrophic destruction of the equipment? does that matter?
* What will these things look like in two years? Five years?

Out of this should begin to grow a clear picture of precisely what kind of equipment is needed. It is possible to begin to look at various types of equipment, and do some preliminary system design.

VI. Shop Around

Sometimes, the very best way to begin to select a piece of equipment is to discover what your friends have done. Just as this can be very helpful, it can be equally disasterous - they may have made all the mistakes you are trying to avoid.

Computer stores can also be helpful. Many of them are run by computer enthusiasts/experimenters who have decided to go into business. They are learning about business, as well as very knowledgeable about the equipment they are selling.

As you look around, here are some things to look for:

* Who will service it? If I need some special equipment attached (ex: a special printer or diskette drive), who will service the interface? Are maintenance agreements available?

* Who will program it? Does the store have packages which appear to meet my needs? Is the source available so that it can be tailored to my needs? Does the store stand behind its software? (Beware of packages which are everything to everyone. They are usually everything to noone.)

* How usable is the software? What happens if the person makes an error? Can it be corrected? Does the system crash? (The problem with software is usually not how difficult it is to use when it works properly, but what to do when something goes wrong!)

* Will the system meet my needs two years from now? Five years from now? What will it cost to make those expansions/additions?

* What happens to my data if the system crashes? Are there ways to "backup" the data?

* What kinds of power fail/conditioning is available? (In most cities, powerlines are subject to aperiodic surges/transients which can ruin, or at least cause a piece of equipment to momentarily malfunction. If the application is critical, that can be costly.)

* What special forms/supplies does it require? How campatible is it with other systems? Can it handle the volume of information I have. (Beware: Because a system can handle a list of 2,000 names almost instantaneously does not mean it can handle 20,000 in ten times as long. Most processing times are NOT linear. They (from experience) appear to be at best quadratic, at worst exponential.)

* What is the reliability of the equipment? How often do the components break down? Who else has one? (Go talk to them!)

Armed with answers to those questions, you will probably be able to do a good job of selecting two or three systems which might work for

you.

However, do not limit yourself to a computerized system. There are lots of excellent manual systems around, and it may be that reworking the present system, changing discount structures, or similar can make the present system far more cost-effective and productive than any computer system could ever be.

VII. Calculate the Costs/Benefits

This is essentially the same as step III above, however, it includes the following: Preliminary design of the system tailored to each piece of hardware/software, and an estimate of the additional cost of implementing and operating each of the new features not in the present system.

Here are the steps which must be taken:

First, outline a procedure manual, complete with sketches of forms, paper flow diagrams, decision tables (or whatever you feel comfortable with to describe what to do given various conditions). Make special note of how the computer will be a part of this.

Second, estimate the costs. Think through every item of cost, just as in step III above, required to produce the product. Also estimate all of the capital costs involved, including programming. (A word on programming. Professionals should usually be able to give you a good estimate of what iw will take to solve your problem. However, more often than not, they will underestimate the complexity of the problem and the time to solve it by a factor of two or more. It is probably wise to double whatever they give you, unless you can carefully spec out what you want them to do, and get them to do it on a fixed price basis.)

Third, estimate the costs of any new features in the system. This is essential in order to maintain the same baseline as the current system.

Finally, compute the unit costs of each of the proposed systems. In addition to this, make a list of the unique benefits/hazards/disadvantages of each system which cannot be quantified easily. These should include adaptability to new situations, special software or support which is available, special difficulties which might be foreseen.

The homework is done. It is time for decision.

VIII. Choose A System

This is a management perogative. If others have been helping with the project, they may be consulted, but you will be the one who has to live with the decision. You had better make sure that it is one you can live and thrive with.

Though there has been a lot of emphasis placed on the costs of the various aspects of the system, cost is not the only factor. Ultimately, there may be other over-riding concerns which have far more weight than the difference of a fraction of a percent in cost. This though must be recognized: unless the costs can be kept within certain bounds, your products will not be competitive.

A further consideration as you look at the costs - capital, present unit costs, and projected 2/5 year unit costs - there are tradeoffs between spending a little more now, to be able to do it much more cheaply later. This may have an impact on marketing and other aspects of the company. This kind of highly intermeshed effect can only be evaluated by top management.

Once you have tentatively made a decision, sit on it for a week. Then come back and look at it. If it is still good, and you still agree with your reasons, move ahead. (Note - it is invaluable as you think through the decision alone, or with others to record it in writing or on tape. This will give you something to look back on when evaluating the system in three or four years.) As you move ahead, plan to stick with it for 3-5 years.

Conclusion

With the increasing availability of cheap hardware, computers are more viable for small business applications than ever before. Choosing the right equipment and software involves:

1) Defining your applications
2) Describing the present situation
3) Calculating the present cost
4) Making future projections
5) Defining the requirements of the system
6) Shopping around
7) Calculating the costs/benefits of the proposed systems and
8) Choosing the system.

Following this procedure will not guarentee success, but it will help.

IMPLEMENTING A SMALL COMPUTER SYSTEM

Casimir C. "Casey" Klimasauskas
295 N. Los Robles #305, Pasadena, CA 91101
(213) 440-1585

Abstract

Selecting a small business system is highly dependent upon the needs and economics of the present and future situation. The implementation of the system chosen is primarily a management task. To be effective requires a thorough understanding of the potential pitfalls as well as the overall flow of the process. The key to success is anticipating as many of the problems as possible and solving them before they happen.

Introduction

Once the decision has been made, and a system chosen which includes a small computer, the key to success is management. The two primary pitfalls which occur at this time are:

1) Fear of the computer as a rather mysterious machine capable of wonders untold. This inevitably results in being sold a bill of goods by both software and hardware vendors. It also creates the unwarranted expectation that because it is a computer, it can do ALL that computers do.
2) Failure to grasp the complexity of the implementation task. Oversimplications lead to unrealistic goals, impossible schedules, undue haste, unnecessary waste and ultimately, failure.

This presentation attempts to describe the major phases of the implementation of a computer system, and in the process to highlight some of the common pitfalls which occur. In so doing, I hope that you will gain a sense of confidence in approaching a computerized system, as well as an overview of the complexities involved.

The major phases of the implementation of a system are:

1) PLANNING
2) HARDWARD installation
3) SOFTWARE development/customization
4) SECURITY measures
5) the HUMAN connection - conversion
6) MANAGEMENT of the new system

In practice two or more of these will occur concurrently. However, for ease of presentation, we will consider them sequentially.

I. Planning

The crux of managing anything is thorough planning. Planning should include a detailed description of when significant events should happen, how much it should cost for them to happen, and at least a sketch of backup plans in case critical milestones are not met (including the additional costs expected).

The primary purpose of planning is to establish a means of measuring progress. Furthermore, the more detailed the planning, the more realistic the goals and schedules are likely to be. We all like to win, and planning is the place to start.

Several excellent tools for planning have been developed. I would like to suggest two which many of you are likely to be familiar with: The Gantt Chart and the PERT chart. There are several readily available readable books which describe both of these methods and variations of them.

The most overlooked aspect of planning in small computer systems, is the failure to develop, as part of the plan, specific times for review, and criteria for the acceptance of the work/equipment received to date. It really comes down to the failure to make quality control a part of the plan. Do not be deceived, quality control in the implementation of an operational system is as important or more important than quality control in the production and distribution of manufactured goods/services.

"To fail to plan is to plan to fail." Plan to be a winner. Plan.

II. Hardware Installation

You have signed the purchase agreement. The manufacturer has agreed to deliver a specific piece of hardware. You are now the proud owner of a child about to be born. Over the next weeks and months, you will have the privilege of nurturing this babe through the valleys of chaos and rivers of pressure to a useful, functioning, mature adult.

However, health requires proper pre-natal care. The place to start is the contract with the manufacturer. Here are some things to look for and beware of:

First, hardware vendors, though they may claim things for their system, are not liable for the performance of the finished system. A recent landmark decision held the software vendor responsible. Regardless, the situation is not clear. Working with small systems, you are

likely to have little recourse given failure. Your best bet is to:

1) Have a competent analyst study your situation, evaluating the ability of the hardware and software to perform the tasks you require with a tolerable response time. Hardware vendors tend to consistently underestimate the complexity and magnitude of a task.
2) Talk to other customers who have applications similar to your own. The real proof of anything is in its performance.

You may be able to get a manufacturer to agree to a certain level of performance of the equipment. If they are unable to produce, it could ruin your company. Legal recourse is useless if you are no longer in business.

Second, computers though small are remarkably complex machines. So complex in fact, that even their designers sometimes are unable to understand why a particular machine may fail in a particular way. I worked on a system which was technically inoperative for six months. Teams of specialists from across the country studied the problem without success. Finally in a drastic move they began replacing entire racks of components at an unthinkable cost. The moral of this is: computers need lots of tender loving care.

The first place to begin is with the source of power. Though many come with power fail-safe features already built in, few are able to cope with massive power surges caused by switching large inductive loads (motors) on and off. The power line, even for a small desk-top computer should be isolated from other power lines which are susceptible to power surges. Power conditioning units may also be helpful in stabalizing the operation of the equipment. It takes only one large surge to destroy a small system (or a large one!).

It may not appear like much, but static charges also wreak havoc with small computers. Plastic soles on nylon carpeting have been known to do a very thorough job of imitating lightening striking the machine. The room in which the computer is housed should have tile floors, or special conductive carpeting. Furniture should be made of wood or metal. No acrylic or similar plastics. A further precaution is to run grounding wires explicitly between the various peripherals and the central processing unit to a common ground point which is connected to a nearby global ground.

Finally, dust and smoke can absolutely ruin a system. Many small systems today are economically viable because of the spectacular advances in magnetic recording density. Disk drives today are able to store 100 times as much information in virtually the same amount of space as a decade ago. However, increased density means closer tolerances. In fact, it is not untypical for the record/play head to be 1/100th the distance from the surface as the size of a piece of dust, and 1/5th the size of a particle of smoke. The room housing the equipment should be kept as dust-free as possible. And no smoking. (I should note that manufacturers attempt to include devices within the equipment to keep dust and smoke out, even so, the life and reliability of the equipment is severely reduced by dusty or smoky environments.)

III. Software Development

The real problems with a computer are not when things are going right, but what to do when they go wrong. It may be possible for a $2.85/hr. clerk to enter information into the system, but it may take a $28.50/hr expert to fix it when they type a wrong code. A key to the successful operation of the system will be well-designed, user-oriented, idiot-proof, debugged software.

User-oriented. Most of the people who program computers do not use the programs they write. In one system on which I have worked, one of the most powerful, sophisticated programs on the system was set of routines for debugging system programs. As the manager of a small system, it is your task to make sure that you have worked through the specifications for the mode of input, update and output which interface well with you and your people. Some things to look for:

* The ability to go back and edit/correct the current item/record before releasing it to the system.
* Any codes which may appear in the data should be displayed in an easily understandable manner.
* It should be possible to enter data in a natural sequence. User convience is where the long-term payoff is, not programmer convenience.
* Input should be in as free a format as possible. Severe constraints to specific columns and/or sequences are an invitation to errors.
* If possible, the user should have some immediate feedback as to the gross validity of the information just entered. (You would be surprised how much difference a keyboard which "clicks" can make over a silent one!)
* Update routines should be able to display the information to be modified as well as the new information. On-line interaction is far more costly than a batch-oriented service bureau if it is not possible to instantaneously validate the new information. For quality control purposes, these routines should include the ability to maintain a log or journal of the changes which are made.

Idiot-proof. As I mentioned earlier, the problem with a computer comes not when everything is working properly, but when something goes wrong. If keying a letter where there should be a digit causes the program to abort, the program is useless. Programs should be designed with the resilience to allow the user to make the most hideous of mistakes, and patiently to note the error prompting for a re-input of the faulty data. In a similar vein, batch oriented programs should have the ability to attempt to interpret the intention of the user, and if at all possible to continue processing, at least through the initial input stages. Furthermore, the more explicit and self-explanatory the diagnostics are, the easier and more accurately semi-skilled personnel will be able to correct the problems.

It costs money and system resources to idiot-proof a program. However, I am convinced that it is worth the cost. The purpose of the computer is to make the personnel using it more effective. It is to be a servant to them, not an exacting task-master.

Debugged. For most programmers, the two least liked and most ignored tasks are debugging and documentation. Perhaps it is the premodona complex which develops from being a master of what to most management is a black art, which gives programmers the feeling that whatever they code is intrinsically correct. At this point, the management proverb, "it is not what you expect that matters, but what you inspect" applies.

It is not enough to specify that the system should be debugged. You as the manager should make sure that a significant body of test data has been run through the system, and the results checked. Moreover, the test data should be designed to place the system into every conceivable failure mode - because one of them will inevitably occur at just the wrong time.

In dealing with software vendors, ask them for customers similar to yourself whom you can talk to. Find out from them about the kinds of problems they have had with the software. Regardless of how attractive the package may look, if problems have been persistent, beware.

One final word. Regardless of how thorough you are in attempting to make sure that the system is flawless, the perfect program does not exist, Users have a way of concocting input capable of placing a program into states never conceived of by the designer. Expect and plan for this.

Documented. If you are hiring the programming of the system, or purchasing the package from a company dubious long-term stability, insist on thorough documentation. This should include listings of the programs, written descriptions of the philosophy behind specific alternatives for implementation, the overall flow of the system, interface with other programs, and sample compilation/executions of the program. In summary, you need a roadmap or repair manual which a competent analyst could study and use to enhance/maintain the system.

Well-designed. Software theorists and large-system builders are claiming amazing improvements in programmer productivity through the use of high-level languages and various programming methods, particularly "structured programming." To an extent, their claims are valid. However, they are usually in the context of a very large system with resources to squander. Unfortunately, in the small-computer realm, resources are generally very limited. The result is that a balance is required between response time of the system/program size and the level and methods of programming used. In many ways, given that you have carefully defined the information you need, and the ways you will use it, the system should last for 5-10 years. With that you can afford to put more into the development of an efficient system than to pay for it in the long-term production use of the system.

Furthermore, some things are not only done more efficiently at the level of machine language, they are more easily implemented at that level. Generally, a well-designed system will utilize a core of well-designed machine level routines tied together with a high-level main program.

How do you make those decisions? It's tough. The best advice I can give is to suggest that you find a skilled analyst/programmer who has had experience at all levels of programming on your particular system. It will be expensive, but worth the cost given your long-term committment to the system. Ultimately, it comes down to trading capital now for returns later.

In managing the creativity of programmers and selecting among software vendors, there are two tendencies which are danger signals. They are the "octopus syndrome" and the "all things to all men syndrome." Both of them indicate a failure to understand your needs, and spell unnecessary cost, complexity and confusion.

The octopus syndrome characterizes systems which do all kinds of interesting things, but have difficulty performing the task for which they were designed. These are often characterized by a pleothera of input/output/processing options which have no apparent relevance to the primary purpose of the program. It is like management: unless it continues to concentrate and eliminate, it will tend toward diffuse ineffectiveness.

The all things to all men syndrome is a bit more subtle. In many ways it will be similar to the octopus syndrom in that the user has an abundance of options. However, all of them are required to make the system work. None are optional. It takes a skilled expert to achieve even the simplest results.

Neither of these problems are fatal, just costly.

IV. Security

The three aspects of security are: access, data integrity, and catastrophy.

Access. Though the computer will be an in-house phenomena, it is crucial that there be methods of keeping both authorized and unauthorized users off the system when they are not supposed to be on the system. The computer-related theft in the U.S. is reported at about $2 billion annually. Some experts estimate that it actually runs $20 billion or more. The most distressing aspect of this is that most of it is by "trusted employees."

Access can be controlled through software, by passwords and the like. Perhaps the two best controls are: First, the ability to physically secure the equipment from employee access during off hours. Though somewhat severe, a small company cannot afford the potential disaster which can result from unauthorized access. (In fact, large companies cannot afford it either!) Second, the system can be designed with a series of built-in audits which trace all accesses to the data on hard-copy. This tied to independent spot surveys of transactions and key elements of the data can provide a deterrent to unauthorized use.

Intergrity of the data. We have touched on this to an extent in the discussion of hardware. In most cases, the information contained in the computer will be vital to the operation and success of the business. It is dangerous to leave all your eggs in that one basket without a backup plan. The system should have designed into it the ability to make a copy of all of the data stored within it after every major update. This copy of the data should be stored in a secure place offsite. Included with the data should be a thorough description of the configuration in which it is used, as well as all of the source and object files required to operate.

In the event of a major hardware failure, the inadvertant crashing of a disk-head into the recording surface, at most one or two days of work will be lost. With this in mind, procedures should be developed as part of the everyday work-flow which hold transactions until the next backup is performed. The value of these procedures or the cost of their absence ultimately becomes apparent when it is too late.

Catastrophy. It is hard to get insurance agents to protect you from "acts of God." However, there are a number of common sense things you can do to minimize your liability. All of them extend the concept of backing up the data so that other systems can be used. The best way to protect yourself is to continually seek out other users geographically close to you who have a similar configuration. Then, attempt to work out a mutual aid agreement, enabling you to use their equipment in the event of a major catastrophy. Though you may not be able to operate at the same levels as usual, it may be enough to survive. Key to all of this is a thoroughly integrated disaster plan.

V. The Human Connection

Hardware and software do not make a system. It takes people. Change always seems to be resisted, feared, and even violently opposed. The keys to success in change are the same as the keys to failure. Success demands open, honest communication; a well-defined transition plan which give the personnel a sense of their part in the new system; and a human system/interface including "triggers" which insure that all the tasks necessary do in fact get done. Furthermore, running parallel systems on at least a limited basis, making the transition during a slow period, and keeping the new system to the most basic elements will insure the ultimate success of the project.

Communication. People need to be affirmed. They want to know that they are needed and appreciated. The danger of changing the status quo lies in the insecurity and doubts it raises in the minds of those presently working for you. It is crucial that they understand, at the level where they are at, precisely how the new system will help them, and even to allow and encourage them to have a part in designing portions of it. Win them to your side. They are ultimately the ones who will either make it succeed or fail - regardless of how well designed and implemented it may be.

Well-defined transition plan. This serves two functions. First, it gives those affected by it a sense of security. They know exactly what is expected, and when it will happen. There are no surprises. (Just as you in management don't like to be surprised, those who work for you don't either.) Second, it gives you a way of measuring your progress toward the goal. Significant slips in schedule may point to the need for major action.

Individuals usually won't put something off till tomorrow if they can put it off until next week. The problem with that is that management cannot afford for it to be put off till next week. It then becomes a point of tension. In most instances, it is possible to build "triggers" into the system - constraints which in the course of the normal processing of information require the individual to complete specific tasks, which though seemingly peripheral to the task, are made an integral part of it. These can be in the form of shipping/receiving documents, slips of paper, or other. The key is to make the tasks which must be done as easy to remember and perform as possible.

Parallel systems. These are very costly. In complex systems, they are one of the few ways it is possible to verify the proper operation of the system. It's like playing Russian Roulette to begin using a system without a way to thoroughly test it. In many instances, it may be enough to run a portion of a month with both systems, and compare the results at the

end of the period.

Slow time. I have never seen a new system which did not have bugs and problems which needed to be worked out during the first few months of operation. With that in mind, it is the height of foolishness to make the transition to a new system during a heavy season of the eyar, or immediately before a crucial period. If prior schedules have slipped, it is likely cheaper to wait until the current heavy season is past than to try to make the switch too quickly. Haste makes waste.

Keep it simple (KIS). Remarkable though it may be, the human mind does have limitations. The more "new" or untested features you attempt to bring up at once, the more likely you are to fail. It is my opinion that the probability of failure is proportionate to the cube of the number of new features. So, keep it simple. Do only that which is essential until it is working properly and effectively. From that tested base, move on to implement the frills in the system.

Two final words on this. First, continued development during the transition stage will be inevitable. Plan on allocating 30-50% of the equipment time to development personnel. That will require careful planning on your part for those who will need to interface with and use the equipment in a production mode. Don't expect to get a lot done furing the first week or two of the transition. Most of the time will be spent in training and ironing out the bugs. Though this may seem a bit costly, it can more than pay for itself by requiring each of those trained, as well as the trainer to write a procedure manual in the process of the training. Integrating these with the original system design make excellent training manuals for new employees. Second, a company's most valuable resource is Its people. Once you have gotten started, barring a bad decision in the selection of a programmer/software vendor, stick with them. Programmers/software vendors are really more artists than engineers. Programmers/software houses who can take something written by someone else and maintain it are very very rare. Once you have an acceptable team, keep it. Adding more people to the staff or replacing present staff has been shown to be the most ineffective and costly way to get a system operational.

VI. Management

A computer is a tool. Though very powerful, it is just like any other major piece of equipment. Unless it is treated as such, it can become a useless appendage. The three elements key to management of any capital resource are: scheduling/performance evaluation, people, and maintenance.

Scheduling/performance evaluation. The key to successful management is goals. In relation to a particular piece of equipment, these express themselves in the form of a schedule which outlines particular tasks which are to be accomplished in a particular period of time. Depending on the particular use, this may be highly formalized or rather informal. In any event, the expectations about the equipment must be understood by the people using it. Expectations are not worth much unless there is accountability. Built into the operations of the equipment must be methods of measuring the effectiveness of the operation. This might be expressed in terms of current backlog, turnaround time, processing volume. . . . The key is that it must be measurable, and it must be measured.

People. They are the most valuable resource a company has. When you find a good person, keep them. Continually review procedures and tasks with an eye to simplifying them. Train backup people who can cover for each other. As with all management, the right person in the right job makes for success. Failure to get this match leads to ineffectiveness and frustration for everyone.

Maintenance. Preventive maintenance usually pays off. Maintenance contracts are like insurance. The rates are designed to cover the expected yearly maintenance costs for your equipment. You gain little or nothing in the long run by assuming that you will "save" money by contracting for maintenance when you need it. Furthermore, maintenance contracts often guarantee a response time. Those who take their chances with maintenance on demand wait in line.

From another perspective, preventive maintenance saves money by allowing the manager the ability to control the "down time" on the machine. Unexpected failures will still happen, but reducing their likelihood through scheduled down times facilitates the effective utilization of personnel during that time.

Conclusion

Implementing a small business computer system is primarily a management problem. It involves:

1) Planning
2) Hardware installation
3) Software development/customization
4) Security measures
5) People and
6) Management - scheduling, people, maintenance.

Though similar to management in a manufacturing plant, successfully dealing with a small computer system demands an understanding of some of the pitfalls peculiar to computers.

EDP PERSONNEL AS INDEPENDENT CONSULTANTS

T. Michael Flynn
President
Personal Systems Consulting
P. O. Box 20286
El Cajon, CA 92021
(714) 443-5353

Abstract:

There is a large market for microcomputer hardware, software and services. Today a large percentage of the market is going to retailers with little or no previous Electronic Data Processing (EDP) experience. There are thousands of EDP experienced individuals who are making no contribution to the microcomputer industry nor are they receiving any benefit from the greatest boom in computer history. As a result too many microcomputers are being poorly implemented or are not being implemented at all in applications in which microcomputers are cost effective solutions. Those most qualified to meet the challenge of this dilemma are not participating in any general sense. The following discussion regards supporting EDP personnel as microcomputer consultants.

Overview:

What is the microcomputer explosion? If you are already convinced microcomputers will become intricately involved in the functioning of society you already have the answer. Please bare with some loose predictions regarding the industry. Alan Kaplan shared some observations by his firm, Venture Development Corporation, in the May, 1977 issue of Computer Decisions, where he stated that of the 17,500 computers purchased in 1976 for home use, $\frac{1}{4}$ ended up in commercial and educational environments. VDC determined that a $1,200 system could reach 8,000 to 25,000 units in just 2 years. A consumer system mass marketed for just under $600 could reach sales of 18,000 to 55,000 in the second year after introduction. Radio Shack is bearing this prediction out in TRS-80 sales. Vantage Research projected that system sales for units under $1,000 would reach a volume of $640,000,000 by 1982, a projection that does not consider the large resulting add-on peripherals market.

Who's buying? VDC in a survey of 1500 hobbyists found that about half of microcomputer sales are to people who are already paid as programmers or electronic engineers. It is interesting to note that those who have the most to offer the industry are a substantial part of the market for the industry. The need for low-cost data processing will dictate a market place far beyond the personal interest of EDP personnel and engineers. The market for medical applications alone will amount to $billions in microcomputer sales. The 1978 Mini-Micro Systems market survey, polling over 42,000 subscribers, disclosed that of the 7,800 valid responses returned 35,000 microcomputers were to be purchased in 1978, a 74% increase over 1977 in commercial applications.

The majority of microcomputer sales are being enjoyed by individual retailers who have had little previous EDP experience, but because they have sources of supply and in every city there are people with true EDP needs who will risk a small investment, microcomputers are selling in staggering numbers.

Are not EDP people involved in microcomputer technology? If they aren't they will be. In a U.S. Department of Labor report predicting the job outlook for 1978-79, it was stated that the job market for programmers and analysts, although negligible in 1960, will reach 500,000 by 1985. The job market for the same people reached 230,000 in 1977. The same report stated that job definition for these same categories would be influenced by advanced in technology and the small computer boom. Analysts will focus on the needs of the small user, away from large system houses. Are not EDP personnel interested in the

opportunities afforded by microcomputers? The number of sales to EDP types indicates a resounding "yes." In the June, 1978 issue of The Institute, the IEEE discussed its findings in a survey sent randomly to 3,000 members. Of 83 topics of primary interest to the IEEE constituency, the top 3 topics were respectively: 1) microprocessor applications, 2) microprocessor software design, and 3) starting and managing an independent consulting firm. Why are not more EDP types involved in microcomputer sales? There are a number of reasons, but high on the list are 1) there are no adequat sources of supply for professional hardware and software available outside of retail stores 2) for those never involved in retail, the process seems often mysterious or complicated, and 3) EDP types, although they consider themselves fully qualified for their jobs, usually do not consider their experience and training diverse enough to qualify as consultants. Whatever the reasons the microcomputer market place is deprived of the benefit of thousands of man years of experience and education.

What's the Solution?

Set up organizations that support enterprising EDP experienced individuals as independent consultants. Such an organization must 1) provide professional hardware and software at attractive discounts and reasonable retail prices, 2) provide a maintenance network inabling the consultant to support his customers, 3) provide comprehensive marketing assistance to help the consultant expose his products and services and enjoy follow-on sales, and 4) provide an ongoing educational program that enables the consultant to grow with the advances in the industry and the needs of his present and future customers.

Definition:

Even though the average EDP type has years of training and experience in his or her field, the average EDP type will not consider his background sufficient to consult. Considering that the vast majority of systems, which include disk and printer, are sold to businesses with no EDP support, the EDP type has much valuable expertise to offer. With a brief but thorough exposure to microcomputer techniques the new consultant becomes the vehicle for implementing truly cognizant and functional systems so badly needed throughout society. I choose to refer to such individuals as personal consultants rather than "independent" consultants because consultants, typically high priced and specialized in their fields, are necessary to isolated areas of expertise and need. The personal consultant shares his expertise with individuals with varied needs and applications at a cost that is transparent to the user, since such costs are absorbed in the total system price.

Hardware Support:

There are many high quality systems available today, but few truly versatile systems are massed produced. The Exidy "Sorcerer" may be the first qualified contender in this category. What is required is typically; 1920 characters per screen, a proven ASCII keyboard with the entire ASCII character set including all control characters, lower case letters with true descenders, RS-232 serial capability, a full parallel port, not just 8 bi-directional lines, but 8 in, 8 out and control lines, standard expandability which means S-100 interfacing, memory expansion, the capability to change languages and operating systems easily without having to bypass fixed ROM, the capability to add both flexible and hard disk as well as the most popular printers and specialized peripherals. The system must be in integrated cabinentry and must be durable and attractive. The consultant must have an adequate supply of such systems and peripherals.

Software Support:

Operating system software must be well documented for the end user and technically explicit for the consultant. The most popular operating systems must be made available for standardization and transferability. A network of consultants could do much to determine the specifications for a truly universal operating system which could be later implemented on the agreement of the user community. Flexible and versatile file handlers and data base managers must be provided, as well as sort and merge packages and utility routines. Languages available should include a good BASIC allowing the end user to learn his system, FORTRAN for the large market it attracts, COBOL for the firms embedded in that world, ASSEMBLERs for access to the

full power of the system, PASCAL to insure that at least one decent computer science type of language is available and perhaps APL for that large special interest group. Application packages must be as general as possible providing the basic functions of any particular application. The bells and whistles can be left to the individual consultant. There must be business packages, medical packages, packages for the home, public information, libraries, industrial control and education. Consultants must have access to what other consultants are developing and likewise he must have an outlet for his own products and services that he develops to reduce redundancy and save time and effort.

Software has always been the greatest money sink in generalized systems. Unsophisticated users do not easily accept this fact. Although in any given application one firm tends to reinvent the wheel, the need for EDP assistance will motivate acceptance of standard industry approaches. In such cases software costs must be obscurred in the overall system cost or at least be minimal. For example, a business package composed of payables, receivables, ledger, inventory and payroll for the small business should be well under $1,000. If the need is legitimate, the consultant should work via a letter of intent on a time and materials basis avoiding fixed price arrangements and contracts. If a firm with a standard or common need is resistent to the EDP procedures required (typically because the bookkeeper is paranoid about computers) do not attempt to implement a system in that firm. They will never receive full benefit and you can't afford the bad publicity.

Maintenance Support:

As more consultants congregate in dense market areas, maintenance centers can be established. Until that time the consultant must know that repairs will occur within a week and shipment over night. A flexible "loaner" program must be established giving the customer the confidence that if his printer goes down during Friday's payroll processing, a maximum 2 hour delay is the only penalty he will experience before a replacement is on the scene. The consultant must be thoroughly versed in preventative and routine maintenance for every product he sells.

Marketing Support:

The consultant must be supported with a comprehensive marketing program affording him exposure to the public. He must have a solid follow-on sales program to take advantage of the lucrative add-on market. National, local and individual campaigns must be implemented with an accurate referral program. These campaigns would include possible TV advertising, certainly national magazine ads, radio spots, newspaper ads and individual mailings.

Educational Support:

The consultant must have clear concise documentation describing particular applications and the products designed to meet the needs of that application. This will enable a consultant who is not familiar with the needs of medical research to intelligently discuss those needs with an interested doctor. He must have the market information necessary to be able to traverse wide areas of differse applications.

Conclusions:

Although the above tenets have never been instituted in support of personal consultants, we at PSC are taking on this challenge. We feel the benefits to end-users, EDP personnel and the industry as a whole will be far reaching and in great demand. Thank you for your consideration.

References:

1. Kaplan, Alan R., Computer Decisions, "Home Computing/Personal Computers: The Outlook for Home and Hobby Computers." May 1977, Vol 9, No. 5, Page 36.

2. Cahners Publishing Company, Mini-Micro Systems, Executive Summary: 1978 Mini-Micro Marketing Survey, 221 Columbus Avenue, Boston MA 02116

3. The Institute, "Microprocessors/ management score in continuing education survey," June 1978, Vol 2, No. 6, Page 1.

4. Wickham, Robert F., Personal Computing Industry Report, June 1978, Issue #1.

5. Dooley, Ann, Computer World, "Analyst, Programmer Jobs Increasing - And Changing," April 17, 1978, Vol XII, No. 16, Page 1.

THE CURRENT SITUATION OF THE

JAPANESE MICROCOMPUTER MARKET & HOBBYISTS

Professor Toshiaki Yasuda
School of Telecommunication Engineering
Tokyo Denki University
2-2 Kanda-nishiki, Chiyoda, Tokyo
Japan

1. History of microcomputers in Japan.

It was in the spring of 1973 that a microcomputer was employed for the first time in Japan. To be more specific, it was when two kinds of microprocessors, known as Intel i-4004 and i-8008, were imported to Japan for sale. OEM showed much interest in i-4004 as component parts for high-class desktop electronic calculators and i-8008 as sequence control equipment.

In the summer of 1973, NEC and Hitachi, Ltd. developed their own microprocessors and put them on the domestic market. NEC developed 4-bit and 8-bit types, while Hitachi developed a 4-bit type, and all of them were of the PMOS type. So far as the domestic market is concerned, however, they failed to draw as much attention as Intel products.

Since around 1975, demands for microprocessors in Japan have shown rapid increase as the import of microprocessors manufactured by U.S. makers including Intel is on the increase and new products such as i-8080A, F-8, PACE, SC/MP, M6800 and PPS-8 have been announced in succession. Thus, in addition to their own developed microprocessors, Japanese semi-conductor makers also explored the possibility of manufacturing such U.S. made products popular in Japan. As a result, they began to launch the second source production of U.S. made products. To begin with, NEC started the production of i-8080A and i-4040 and their family LSI under agreement with Intel, and then Hitachi and Fujitsu started the production of M6800 and its family.

2. Microcomputer makers in Japan.

After the second source produc tion of i-8080A and i-4040, NEC is, in principle, continuing the production of Intel's subsequent products such as the 8085 family. Following the Hitachi Ltd. and Fujitsu Ltd. the Japanese makers engaging in second source production under agreement with the U.S. makers are on the increase as shown below:

1) Intel i-8080A
   NEC, Mitsubishi Electric Co., Oki Electric Co. and Toshiba Electric Co.
2) Motorola M6800
   Hitachi Ltd. and Fujitsu Ltd.
3) Rockwell PPS
   Sharp Co
4) Zilog Z-80
   Sharp Co.

In general, the U.S. products had drawn much attention from 1973 through 1977, but since late in 1977 those microprocessors, mainly of 4-bit one-chip type, developed by the Japanese makers, have come to find a large market. The typical makers of such microprocessors are as follows:

---1) NEC
   *4-bit nMOS one-chip
   *4-bit CMOS one-chip
   *16-bit nMOS processor
---2) Hitachi Ltd.
   *4-bit nMOS one-chip
---3) Fujitsu Ltd.
   *4-bit nMOS one-chip
---4) Sharp Co.
   *4-bit nMOS one-chip
   *4-bit CMOS one-ship
---5) Toshiba Electric Co.
   *4-bit CMOS one-chip
   *12-bit nMOS processor
---6) Matsushita (PANASONIC Group)
   *4-bit nMOS one-chip
   *16-bit nMOS processor

3. The Japanese Microcomputer Market

Since 1973, microcomputers have been applied mainly to electronic cash registers (ECR) and automatic vending machines, and then since 1974 to POS terminals and intelligent terminals. Since 1977, they have found a large application in overall control systems and medical information systems. From early 1978 on, they have come to be used widely for color TV receiver sets, FM receiver sets, cassette tape decks, microwave ovens, electric washing machines, and household air conditioners and computerized home electric applicances which are well accepted by Japanese housewives.

Microprocessors marketed in Japan in 1976 were 1,180,000 units for both domestic and U.S. made products; they topped 2 million units in 1977 and are estimated to exceed 4 million units in 1978. Of these microprocessors, more than 85% are system and equipment built-in computers and the remainder are for use in development, evaluation, education/training and hobbies.

4. The Future of the Japanese Microcomputer Market.

Computer hobbyists in Japan number about 50,000 as of the summer of 1978, and they are

now on the sharp increase. At first, they enjoyed the DIY kit TK-80 for educational purposes which was marketed by NEC for use with i-8080A, and then went for imported kits such as Altair 8800, IMSAI and SWTP. In 1978, PET 2001 and TRS-80 have gained great popularity and heavy demands are also concentrating on the BASIC oriented personal computers marketed by NEC, Hitachi, Matsushita and SORD.

A forecast of what great expansion the hobbyists or personal computers will show in the future can be made on the basis of the circulation of the following microcomputer-related publications written by the author and published by Kodansha, Ltd, the largest publisher in Japan.

1) Introduction to My Computer
(Published in March 1977, 350,000 copies sold).
2) How To Build My Computer
(Published in October 1977, 150,000 copies sold)
3) How To Use My Computer
(Published in April 1978, 100,000 copies sold)

From the above, it may be forecasted that in Japan in the 1980's, demands for personal computers such as TRS-80 and PET 2001 will be 300,000 to 500,000 units per year, while demands for microprocessors will top 10 million units per year. With such a huge market as a target, NEC, for instance, is making every effort to develop into the world's largest microcomputer device maker, and needless to say, this effort is also being made by other makes.

LEGAL ASPECTS OF TRADE ASSOCIATIONS IN THE RETAIL MICROCOMPUTER INDUSTRY

Oscar A. Rosenbloom, Esq.
912 Cowper St.
Palo Alto, California 94301
(415) 328-1712

Abstract

There has recently been noticeable interest and activity in the formation of trade associations in the retail microcomputer industry.

Among the newly formed trade associations are the Computer Retailers Association (CRA), the Southern California Computer Dealers' Association (SCCDA) and the Western Computer Dealers Association (WCDA).

After briefly discussing the history of trade associations generally, this paper discusses legal regulation of the following trade association activities:

(1) standard setting:

establishment of engineering standards for the entire industry or any segment of it.

(2) exchanging price information:

association sponsored catalogs containing manufacturers' suggested resale prices and/or association suggested resale price schedules.

(3) statistical reporting programs

association established pools of information regarding members' experience with equipment to serve as conduit for referral of member complaints and manufacturer responses.

(4) professional designation programs:

association operated professional designation programs designed to move the occupation of computer retailer toward professional status.

(5) cooperative buying:

cooperative buying associations among independent enterprises to seek economies and competitive advantages.

(6) agreements to limit price competition:

express or implied agreements to limit price competition among association members.

The relevant provisions of the Sherman Antitrust Act, the Robinson Patman Act, and the Federal Trade Commission Act will be discussed in connection with these activities.

The role of the Federal Trade Commission (FTC), American National Standards Institute (ANSI) and the National Bureau of Standards (NBS) will be discussed as they relate to trade associations.

HOW TO CONDUCT A LOW-COST MARKET SURVEY

Donald M. Dible, The Entrepreneur Press
3422 Astoria Circle, Fairfield, Ca., 94533 (707) 422-6822

If you have a fat budget, it is a straightforward matter to hire a professional marketing research organization to conduct a survey for your company. If, on the other hand, your marketing research funds are definitely limited, you will be interested in the information contained in this chapter.

Primary versus Secondary Marketing Research

Primary marketing research consists of defining your specific marketing information needs and then preparing an information-gathering campaign for the purpose of securing this information. The campaign can be conducted through the form of questionnaires submitted through the mail and through interviews with customers, potential customers, and others from whom you wish information. However, this approach to marketing research can also prove to be quite costly. The approach to be discussed here deals with secondary sources of marketing information. By secondary sources, we mean compiled studies prepared by marketing research agencies as opposed to primary studies prepared to your specifications on the basis of data gathered from sources within your specified marketplace. Let's consider the largest and most readily accessible sources of secondary marketing information.

Trade Associations

Industry is served by literally thousands of business and trade associations. In the bibliography that follows, you will find listed two directories of trade associations. One of them, the Encyclopedia of Associations, published by the Gale Research Company, lists almost 2,500 trade, business, and commercial organizations. In addition, there are listed almost 800 scientific, engineering, and technical organizations. With few exceptions, these organizations regularly produce one or more periodicals offering in-depth coverage of marketing opportunities in their industries. Furthermore, most business and trade associations produce comprehensive annual surveys of their markets. My own company is a member of several trade associations. One of these associations sends a questionnaire every month soliciting detailed information on our sales for the preceding month. When these questionnaires are completed and returned to the trade association, responses are compiled and copies of the overall industry results are submitted to each of the respondents. In this way, the participating companies are able to determine whether their own sales trends are keeping pace with those of the rest of the industry. These reports also enable participating companies to tell whether the overall market is on the increase or decrease relative to the performance in prior years. Thus, questionnaire respondents are not only in a position to get a fast reading on their own performance relative to the industry but on the performance of the industry as a whole in relation to or as compared to previous years' results.

Many trade associations maintain specialized libraries at their national and regional headquarters. Usually these libraries constitute the most comprehensive collections of marketing information available within the particular industry. Trade associations can be particularly helpful to the marketing researcher by referring him to additional sources of marketing information. Many trade associations publish magazines or newsletters that carry current information about conditions in the marketplace as well as reviews of pending legislation that may have an adverse or salutary effect on market conditions.

Stock Brokerage Firms

I'm sure you are aware of the fact that stock brokerage firms make money whenever their customers buy or sell stocks. Therefore, any information concerning a given industry that might prompt customers of the brokerage firm to buy or sell stocks in a given industry would serve to produce business volume for the brokerage house concerned. All major brokerage houses maintain a staff of marketing analysts whose job it is to study various industries that may be on the verge of experiencing a significant change, either favorable or unfavorable. Of course the brokerage firm would recommend the purchase of stocks in industries whose outlook was favorable while at the same time they would recommend the sale of securities in industries whose outlook was unfavorable. You may contact various brokerage houses yourself in the hopes of securing access to their studies of the industries of interest to you. A more effective and efficient way of locating the brokerage houses that have done surveys of your industry might be to contact your trade association and ask if they know of any industry studies that have been conducted by brokerage houses. In preparing its independent study of your industry, the analyst for any brokerage house would, in the process of conducting a thorough analysis, contact the

trade association, thereby insuring that the trade association will know of the existence of the study in question.

Local Business Organizations

Many local business organizations such as chambers of commerce, banks, utilities, telephone companies, newspapers, and colleges and universities engage in varying degrees of marketing research. If a particular industry is important to the financial well-being of the community, you may be certain that one or more of these local business organizations will have completed a study of this industry. If you are fortunate enough to be in a community served by a college where courses in marketing are offered, you may be able to persuade a student to undertake a study of your market as a class project or as a research project. Many banks tend to specialize in certain industries. For example, the Bank of America, headquartered in San Francisco, is heavily involved in financing California agriculture. One of the important sectors of the California agriculatural industry is the wine industry. The Bank of America recently published a study titled "California Wine Outlook." This report contained a discussion of distribution trends, classifications, factors affecting wine sales, the outlook regarding supply constraints, the import outlook, international perspectives, forecasts, and an analysis of the wine grape industry. This report was made available at no charge to clients of the bank.

Trade Publications

Business and trade publications are among the most authoritative sources available of timely, specialized industrial information. Announcements of the latest commercial products, analysis of major innovations in marketing, and discussions on manufacturing and research and development activities are regular editorial subjects. Invaluable market survey and market forecast information is frequently printed in these publications before it is promulgated by any other news medium. Meeting and convention calendars are also commonly included as a convenience to readers. As in any other business, there is competition in the field of business and trade publications. Those publications that are most effective in serving the information needs of their readers enjoy the higher readership. One of the ways in which trade and business publications serve their readers is through the annual industry survey issue. Some business and trade magazine publishers bind their annual industry survey separately, since magazines are customarily disposed of monthly whereas the annual survey is intended to serve the reader for a full year. To give you an example of the specialized nature of some of these surveys, consider the survey produced by the magazine Dental Economics. Annually, Dental Economics Magazine produces a "survey of purchasing trends among dentists" designed to measure the amount and source of dental supply purchases made within that industry. This report is concerned with dental supplies exclusively. I think you will agree that for someone in the business of serving the dental marketplace such a report would be invaluable.

Here's another way in which you may benefit significantly by tapping the business and trade publications as an information source. Industry survey articles generally require months of preparation and research. The first draft of an article for publication might far exceed the amount of space available in the actual publication. Therefore, the article has to be cut and trimmed to fit the space available.

If in studying a business or trade publication you come upon an article, a series of articles, or a full edition that is of particular interest to you, write a letter to, or get on the telephone and call, the editor or editors responsible for the preparation of these articles. Invariably you will find that they have gathered far more information than is reflected in the printed article. Furthermore, editors are very much like radio announcers. They find themselves wondering, "Is anybody out there reading what I have to say?" Call them on the telephone or write them; you will find these fellows to be most responsive to your inquiries. After all, you are letting them know that they have an audience.

Government Agencies

City, county, state, and federal agencies by the hundreds produce marketing information. Just about the most prolific government agency preparing information of interest to marketers is the U.S. Department of Commerce. As you may know, the Bureau of Census is a division of the Department of Commerce. Periodically, the department puts out a Census of Business. The latest edition is a nine-volume set loaded with exhaustively detailed information on markets in the retail trade, wholesale trade, and selected services. If in your marketing survey you require any demographic information, I suggest that you start with one of these nine volumes of the Census of Business, available in any well-equipped library. One of the most useful guides for the analysis of statistical and demographic information is Measuring Markets, a Guide to the Use of Federal and State Statistical Data, published by the U. S. Department of Commerce. In addition to listing hundreds of sources of statistical data available through federal and state governments as well as through business, professional, and institutional organizations, a number of examples are provided in which the application of statistical market measurement is applied to the problems of selected small businesses in the fields of manufacturing, wholesale-retail, and service industries.

Trade Shows and Exhibits

If you were to pick up a telephone and call your competitor to find out what his marketing plans for next year happen to be, the chances are you would get no further than the receptionist. On the other hand, if you were to visit the exhibit of one of your competitors at a trade show, chances are he would provide information to anyone passing by on just this subject. Possibly you'd like the opportunity to "pick the brains" of your competitor's director of research and development. Not uncommonly, you'll find employees of this calibre on "booth duty" at trade shows and expositions. By exercising the proper discretion, you might very well be able to engage such a person in conversation at his exhibit and secure all the information your little heart desires.

Very often people complain to me that trade shows and exhibitions are only open to members of the trade. This is true. The exhibitors spend a great deal of money for the privilege of displaying their goods and services and in preparing literature for distribution to the attendees. Understandably, they do not welcome a horde of school-age youngsters armed with shopping bags who will clean out their supply of expensive four-color literature printed on coated stock. But if you have a bonafide reason for wanting to attend a trade show, a little resourcefulness should get you in. I recommend this approach. Contact the trade association sponsoring the event and secure a list of the exhibitors. Then contact one or more of the exhibitors, indicate an interest in the products or services that they are displaying, and ask them if they can arrange to have you admitted to the exposition as their guest. I find this approach works like a charm. In the bibliography at the end of this paper, you will find listed several directories of expositions, trade shows, and conventions.

Marketing Research Firms

There are a great many marketing research firms that will conduct primary research for you on a fee basis. Such companies include Arthur D. Little: Quantum Sciences, Predicasts; Battelle Memorial Institute; Stanford Research Institute; McKinsey; Booz, Allen, Hamilton; Cresap, McCormick, Paget; F. W. Dodge; Creative Strategies; and Drossler Research Corporation. Some marketing research firms not only conduct studies commissioned in advance by clients of theirs, but they also prepare industry surveys of a more generalized nature in the hope of selling them to anyone willing to pay their price. For example, the Morton Research Corporation has produced economic and market studies in such areas as beverage and drugs, dairy studies, electronics, furniture and fixtures, home furnishings and housewares, industrial machinery, moving machinery, instrumentation, iron and steel products, jewelry, leather, meat, non-metallic mineral products, paper/plastics/packaging, petroleum, publishing/printing, recreation retail trade, service, specialty/snack foods, textiles/clothing, wiring and wood products. Under each of these headings, there are many specialized reports, ranging in price from $15.00 to figures in excess of $200.00. The name and address of two marketing research firms that publish specialized reports are below. If you wish to secure additional information about their services, please feel free to write on your company letterhead. Although you may wind up spending several hundred dollars for a copy of an industry survey from one of these companies, I personally feel the cost is quite modest when compared with the risk involved in venturing into a new market without the benefit of the insight available through these surveys.

1. Morton Research Corporation, 1745 Merrick Avenue, Merrick, New York 11566.
2. Predicasts, Inc., 200 University Circle Research Center, 11001 Cedar Avenue, Cleveland, Ohio 44106.

Recommended Reading

1. Bursk, Edward C., and Chapman, John F., eds. Modern Marketing Strategy. A Mentor Book. New York: New American Library, Inc. 1964.
2. Frank, Nathalie D., Data Sources for Business and Market Analysis. 2nd ed. Metuchen, N.J.:The Scarecrow Press, Inc., 1969.
3. The Futurist. Washington, D.C.: World Future Society.
4. Klein, Bernard, ed. Guide to American Directories. 9th ed. Rye, N.Y.: B. Klein Publications, Inc. 1975.
5. Sales Meetings Staff. Directory of Conventions. New York: SM/Sales Meetings Magazine, annually.
6. Sales Meetings Staff. Exhibits Schedule: Annual Directory of Trade and Industrial Shows. New York: SM/Sales Meetings Magazine, annually.

HOW TO RAISE CAPITAL FOR YOUR BUSINESS

Donald M. Dible, The Entrepreneur Press
3422 Astoria Circle, Fairfield, Ca. 94533 (707) 422-6822

One of the recurrent problems that plagues most owner-managed businesses is the shortage of adequate capital. In the case of new businesses, this problem can be severe.

Promising small businesses traditionally have relied upon such sophisticated money sources as investment bankers, venture capitalists, and federally licensed and leveraged Small Business Investment Companies (SBICs) for their equity capital needs. In today's market, these sophisticated investors have many options from which to choose. Regardless of market conditions, their objective continues to be the maximization of return on investment, consistent with intelligent risk analysis.

Their first option is the purchase of securities in publicly traded companies at bargain basement rates. Their second option is the private placement of growth capital in young, but already established, companies of demonstrable merit. In a tight money market, the terms of such an arrangement may be very attractive for the investor. The investment option representing the highest risk, obviously, is in financing the start-up company. Given the first two options, a private placement with a start-up company is, I think you'd agree, a most unlikely choice as an investment vehicle.

What about banks? Surely, you can get a personally endorsed loan for your business from the friendly loan officer at your commercial bank. Not necessarily. In evaluating the financial statement of any business, a banker will pay particular attention to the ratio of debt to equity capital. A general rule is that you cannot borrow more money than has been invested as equity capital. Furthermore, most bankers are not eager to lend money to new businesses at all, preferring to wait until there is at least some history of profitable operation.

How, then, can you assemble the resources you'll need to get started and to keep you going until you begin to turn a profit? Take heart; there is hope. There are, in fact, four different money sources that can now help you stretch your business dollars: 1) personal financial resources, 2) trade credit, 3) customers, 4) economizing.

Personal Financial Resources

Many of us significantly underestimate the value of our personal resources when taking a financial inventory. Let's look at some of the more obvious (and not-so-obvious) resources you may have:

Home. In the recent inflationary period, the value of practically all residential real estate has appreciated substantially. If you have owned your own home during this period, you are indeed fortunate. The difference between the current value of your home and the balance outstanding on your mortgage may represent an equity asset of between $20,000 and $50,000. You might consider converting this asset to cash by 1) taking a second mortgage in the amount of your equity, 2) refinancing your mortgage, 3) subdividing your lot and selling part of it, or 4) selling your house and moving into an apartment. Every year tens of thousands of people go into business using "house money" to bankroll their ventures.

Life Insurance. A loan based on the cash value of your life insurance policies can be a low-interest source of money. Many policies provide for automatic loans from the insurance carrier at interest rates far below the prime rate charged by commercial banks.

Stocks and Bonds. You may own stocks and bonds that, for a variety of reasons, you do not wish to sell. Such instruments make ideal collateral for loans, and you can borrow anywhere from 40% to 60% of their current value, depending on the lender.

Credit Cards and Personal Credit. When you prepare your financial business plans, you will surely want to take the fullest possible advantage of your credit worthiness. At the same time, you will want to minimize the amount of money that you have to take out of the business to meet your personal financial needs. This is particularly true when the business is just getting started and every dollar in the treasury is needed to finance business growth. During this period, personal credit cards of all types and descriptions can give you just the extra leverage you need. Instead of drawing a heavy salary in the first year of operation, supplement your modest cash draw with the judicious use of credit cards.

Credit cards come in a great variety of classifications: bank cards such as Master Charge and BankAmericard, travel and entertainment (T&E) cards such as Diners Club and American Express, as well as cards for department stores, oil companies, airlines, and so on. With a good credit record, it should be no great feat for you to secure enough credit cards to give you a personal line of revolving credit in excess of $30,000.

Now, obviously, you can't buy a carload of raw materials for your business with your bank credit card, but you can cover a lot of your personal expenses with it--as well as such business expenses as transportation, food, and lodging. With a little thought, I'm sure you can come up with a variety of ways to employ credit cards in financing your particular business.

While we're on the subject of personal credit, you should give serious consideration to

arranging for the installment purchase of an automobile and other major consumer items before you start your business. You'll find it a lot harder to qualify for this kind of financing after you've started. Lenders tend to worry a great deal about the financial stability of new small businesses and their founders. Your credit application looks a lot better when, under "Current Employer," you show that you have been employed with "Solid as a Rock, Inc." for the last six years instead of with "Shaky at Best" for the last six weeks.

Relatives and Friends. I happen to consider relatives and friends to be extremely valuable personal financial resources. Where else can you find lenders who, simply because they like you, will advance money on anything as risky as starting a new business? However, do yourself and your lenders a favor. Document their loans. Draw up a formal note showing interest charged and the dates on which principal and interest are payable. You never know when a personality problem may arise that could precipitate calling the loan. That can become awfully messy.

Take the case of the young man whose rich aunt loaned him $100,000 with which to start his business. The formality of a loan agreement was ignored. Six months later the aunt died. The heirs then forced the entrepreneur to liquidate his business so that they could each get their share of the aunt's estate.

Relatives and friends can also come in handy as cosigners. Let's assume that you apply for a loan and the lender decides that your qualifications are marginal. The availability of a financially strong cosigner can swing the balance in your favor. Your relatives and friends can prove to be very important personal assets. Don't overlook them in your financial planning.

Trade Credit

According to a U.S. government study, trade credit constitutes a 33% larger factor in financing the business community than that represented by bank loans. Clearly, every company treasurer should give serious consideration to trade credit in financing a small business.

Customarily, suppliers provide trade credit as an inducement to their customers to do business with them. Although credit terms in most industries are extended on a net 30-day basis, overall averages may stretch this to 45 days, 60 days, or even longer depending on the condition of the overall economy and the particular industry involved. Under the guise of "cash management," many companies make a habit of vigorously enforcing their collection policies while simultaneously treating their accounts payable with cavalier disregard.

Just how you handle your payables is your own business. But you should be aware of the fact that not everybody in business makes a habit of paying bills the day the first invoice arrives.

When it comes to making use of trade credit as a tool in financing your business, I recommend that you establish with your suppliers, in advance of purchase, the best extended credit terms you can negotiate. Depending on how hungry your suppliers are for your business, you may be able to arrange extemely attractive terms. Securing competitive bids will greatly improve your bargaining position. However, once you have agreed to terms of payment, honor your commitment if you value your credit. Nothing is more difficult than trying to operate a business when your suppliers will ship to you only on a COD basis.

Customers

There are many ways in which customers can be induced to help you finance your business. The degree to which they are willing to do so depends on the extent to which you enjoy a monopoly on the goods or services that you offer. In other words, no one will pay in advance if he can get essentially the same goods or services from another supplier under more liberal credit terms. However, if you have the only game in town, you may be in a position to get your customers to finance your business.

As you know, the telephone company, electric and gas companies, the post office, and a host of government monopolies require deposits in advance (independent of your credit-worthiness), so that you may enjoy the benefits of their services. The same kind of policy may be applied in the operation of certain small businesses and selected industries.

A San Francisco Peninsula electronics company, which sells a patented device available nowhere else, provides a good illustration of customer financing. The company's customers are required to make a deposit of one-third of the purchase price when the order is placed. Another one-third is payable on delivery, and the balance is due in 30 days. As a result of these favorable terms of sale, this company has been able to finance a highly satisfactory growth rate while experiencing almost no problems with cash flow.

An extreme example of this type of financing is the case in which the customer is required to pay the full purchase price at the time the order is placed. You may be surprised to learn that you have been financing certain of your own personal suppliers on this basis for years. I am referring, of course, to magazine publishers. If you take a three-year subscription to a magazine, you may get the first one or two issues on credit based on your promise to pay. However, in order to continue receiving the magazine, you must pay the subscription bill, even though you won't receive your final shipment for almost three years.

Hugh M. Hefner, publisher of Playboy magazine, played this customer-finance game with admirable success when he initially offered lifetime subscriptions for $100. Obviously, he needed the money to finance his growth. Had the subscribers to some of the early issues bought $100 worth of Playboy stock instead of a subscription, they could have picked up a tidy profit.

In many industries, including a number of the construction trades, it is usual for the

suppliers to receive progress payments when previously agreed-upon levels of project completion are achieved. Highway construction, aircraft assembly, and other large-scale projects are often financed in this way.

A special form of customer financing occurs in a number of service and manufacturing industries. Here the customer provides raw materials that the vendor transforms into finished goods. In the book printing trade, a publisher may supply the printer with paper; in a machine shop, the customer may provide the metal to be worked; and in a tailoring service, the customer may provide the cloth to be fashioned into a dress or a suit. In each case, the vendor avoids the expense of financing the purchase of raw materials.

Another way of getting the customer to pay in advance for goods or services is to establish some kind of "membership" arrangement. Here the customer may pay an annual fee for the privilege of attending meetings. Using a similar membership approach, some discount department stores require customers to pay a membership fee for the privilege of shopping there.

A novel twist on this customer-backed approach to business finance is seen in physical fitness spas, dance studios, and other contract service organizations. Customers sign an agreement to purchase the service offered for a period of one or more years. In most cases, the intentions of the customer are honorable and sincere at the time he executes the contract. However, many people lose interest in fitness and other self-improvement programs after a short time, and the incentive to pay the installments on the service contract may falter. In anticipation of these long-range collection problems, the original holders sell the contracts to finance companies at a substantial discount. While the average individual may balk at paying the original contractor for services not used, he is more likely to pay a finance company when notified that it has taken over his contract. The result is that the original contractor gets a handsome chunk of cash almost immediately after the customer signs the contract, whether or not the customer continues to use his facilities. Once again, the customer has financed the business, albeit indirectly.

Economizing

In preparing your pro forma financial statements, you probably allocated quite a bit of money for office equipment and other capital expenditures. If you figured on buying new equipment, figure again. Used equipment is what you want. That way, you may find that you need a lot less cash than you originally thought. Forget the rosewood paneled office with the Italian marble-topped desk, too; that comes later. And most financially strapped entrepreneurs quickly learn the delights of night coach and special tour package rates to save a few dollars when traveling. There are many other ways to economize in the operation of your business; we'll get to them shortly, but first let's concentrate on where to find used equipment.

1. Used equipment dealers may handle anything from office equipment to laboratory test equipment to cash registers to display cases for butcher shops and retail stores. You'll find these dealers listed in the Yellow Pages of your telephone directory under such headings as "Used Equipment Dealers," "Second Hand Dealers," and "Surplus Merchandise." You'll also find dealers listed under generic headings such as "Office Furniture--Used."

2. Dealers in new merchandise invariably find themselves stuck with a variety of goods that cannot truly be represented as new. "Demonstrator" and "loaner" equipment (provided for the temporary use of customers who are awaiting delivery of new equipment or who are having their own equipment repaired) fall into this category. These dealers may also have floor samples, warehouse- and freight-damaged goods, and obsolete-but-serviceable rental equipment available at a substantial savings.

3. Bankruptcy and liquidation auctions provide an opportunity to get some real bargains. To secure information on where and when auctions are to be held, consult the Yellow Pages of your telephone directory under "Auctions" or "Auctioneers." Many auctioneers maintain mailing lists of interested clients and send out brochures announcing forthcoming auctions.

You may also obtain information on bankruptcy auctions by contacting the bankruptcy court in your area. You'll find this court listed under "U.S. Government" in the White Pages of your telephone directory.

4. Bankrupt companies that receive protection under Chapter 11 of the Bankruptcy Act are very likely to be interested in liquidating some of their assets. If you are lucky enough to learn about such a company in your own industry, you are in a unique position to fill your equipment needs quite reasonably. Bankruptcy filings are announced in the legal newspapers of record serving various communities across the country. Also, when a large company in a particular industry files for voluntary bankruptcy under Chapter 11, the trade journals serving that industry will usually carry mention of this fact. Upon learning of such a bankruptcy, don't hesitate to call the company involved to determine whether the owners are interested in selling some of their assets.

5. Now and then major corporations simply decide to get out of a particular industry and shut down one of their divisions. A friend of mine who has his own company read about such an instance in a trade journal. He purchased a $100,000 (price when new) piece of test equipment for a mere $7,000 cash. Then he called a leasing company and received $45,000 for the same equipment when he agreed to lease it back from them over a five-year period. He realized an immediate cash infusion of $38,000.

6. The United States Government is the single largest consumer of goods and services in the world. Not surprisingly, it also disposes of enormous quantities of surplus goods

on a more or less continuous basis. For information on the sale of surplus government equipment at locations all over the world, write to the Department of Defense Surplus Sales Office, Box 1370, Battle Creek, Michigan 49016, and the Assistant Commissioner for Personal Property Disposal, Federal Supply Service, General Services Administration, Room 926, Crystal Mall, Building 2, Washington, D.C. 20406. Local representatives of these agencies are listed in the White Pages of your telephone book under "U.S. Government."

7. Classified ads provide a simple, convenient, and inexpensive means of finding used equipment. You can consult the listings under the headings of interest to you, or you may want to advertise the fact that you are looking for a particular item. In some instances, trade journals carry classified listings, thereby permitting you to confine your search for specialized items to the more likely sources of supply.

Barter. Bartering your goods and services is a primitive-but-fun way to save money. It also affords certain tax advantages. I know of a carpenter who remodeled an orthodontist's home in exchange for having his daughter's teeth straightened. I also know of a plumbing contractor who paid for his appendectomy by plumbing his surgeon's vacation home. Radio stations have been known to exchange advertising time for consumer merchandise to use as premiums. The opportunites are unlimited.

Do It Yourself. When the cash supply is limited and the money for meeting a payroll is nil, you simply have to learn to do things yourself that you might otherwise hire someone else to do. You have undoubtedly heard that many small businessmen work 80 or more hours a week. Many of these companies are also husband-and-wife operations, where the wife works virtually without pay until the business starts turning a profit. This kind of toil may not sound like much fun, but it does save money.

Then, too, a lot of businessmen, in response to economic pressure, find that they have many talents and skills they never appreciated. A janitor used to empty their wastebasket when they worked for Big Business, Inc., but they now take care of this occupational specialty themselves. On opening a restaurant, they may find that they possess the dexterity of a short-order cook. They may learn how to write advertising copy, how to repair their machinery, or how to operate a typewriter and a ten-key adding machine.

In some cases where a specialist is needed but there is no money with which to hire one, the entrepreneur may have to take courses at a local college to learn the skill himself. It is often surprising what you can learn to do when your economic survival is at stake.

Free Publicity. Free publicity can do wonders for your business--and the price is right. You don't have to hire a public relations firm on retainer to get it, either. What you do need is guts enough to call a newspaper or trade journal editor, a radio announcer, or a television personality and explain your story. If you have something to say that will be of genuine interest, educational value, or amusement to the audience served by the medium you have selected, you have a good chance of getting publicity.

One businessman I know sent out new-product releases describing a $200 device to 12 trade journals serving his industry. He received more than 1,500 inquiries in response to articles carried in the two journals that printed his story. His total cost was less than $10.

I could doubtless catalogue dozens of other ways of saving money in your business, but I'd like to conclude by suggesting a different point of view. It is easy to become preoccupied with saving nickles and dimes instead of figuring out how to make dollars. Nothing succeeds like a company with the right product at the right price at the right time. The electronics giant Hewlett-Packard was started in a garage in the 1930s. The company grew rapidly with limited invested resources for a very simple reason: the founders were selling unique and superior products with high profit margins to a market eager to buy. Go thou and do likewise.

Recommended Reading

Baty, Gordon B. Entrepreneurship: Playing to Win. Reston, Va.: Reston Publishing Company, Inc. 1974.

Brady, Frank. Hefner. New York: Macmillan Publishing Co., Inc. 1974.

Deiner, Royce. How to Finance a Growing Business. New York: Frederick Fell, Inc., 1965.

Putt, William D., ed. How to Start Your Own Business. Cambridge, Ma.: The Massachusetts Institute of Technology, 1974.

HOW TO GET DISTRIBUTION FOR YOUR PRODUCT

Donald M. Dible, The Entrepreneur Press
3422 Astoria Circle, Fairfield, Ca. 94533 (707) 422-6822

The typical small business has to operate without exceeding a fairly static overhead--salaries for company officers and a certain number of employees, rent, utilities, and the cost of capital equipment, such as typewriters, postage meters, desks, chairs, manufacturing machinery, and computers. There are two fundamental ways in which sales volume can be increased without a significant increase in this level of overhead. Number one, you can expand the product line. Number two, you can increase the channels of distribution.

Example

Consider the case of a shoe repair shop located in a downtown metropolitan area. Over the years, the customers of this shop moved to the suburbs. Also, much of the footwear sold today is not designed to be easily repaired. For these reasons, the sales volume of the shoe repair shop had dropped to a level that threatened the economic survival of the business. One day the proprietor was looking over his shop. Its capital equipment--heavy-duty sewing machines, cutting and polishing wheels, and shoe stretchers--was paid for; he had made several capital improvements on his shop in previous years. Suddenly he had a brilliant idea. He had a number of signs made up with the words "Shoe Repairing" on them. He then made up a special price list indicating the retail charges for the various kinds of shoe repairing that his shop handled. Our marketing-oriented proprietor then visited all of the dry cleaning establishments within a ten-mile radius of his shop. He talked with each proprietor and proposed that, if they would put a sign in their shop window indicating that they would handle show repairing, the shoe repair man would make pickups and deliveries twice a week. The dry cleaning establishments would receive a fixed percentage of the retail price charged as their commission for acting as a middle man. Almost overnight, the shoe repair shop went from a financially marginal operation to a profitable one. Note that the added sales volume in the shoe repair shop necessitated no increase in capital equipment or improvements to the facilities. For this reason, the profits realized on the increased sales volume were substantial. At the same time, the dry cleaning establishments increased their sales and profits by viture of the fact that they had expanded their product line. In addition to taking in laundering, dry cleaning, drapery cleaning, and alterations, the dry cleaning establishments were now able to sell their customers an additional service--shoe repairing.

Product Line Expansion

If a business has one or more clearly established channels of distribution and does not wish to increase the number of channels, it then becomes necessary to broaden the product line in order to increase sales. Consider the case of Screwy Lewy (TM).

A southern California businessman became frustrated one day while trying to open the lid on a jar. He recognized that the problem of opening sticky jar lids was probably as old as the screw-top jar itself. He considered the various efforts of the business community to solve this problem. In most cases, manufacturers had tried to supply the consumer with wrench-like leverage devices. However, this particular businessman recognized that the problem was not one where the consumer was unable to apply sufficient torque, or "twisting power," but rather stemmed from the fact that the hand tended to slip while trying to twist the jar lid open. Our businessman then conceived the idea of introducing into the marketplace a decoratively shaped, high-friction, thin plastic disc that could be set on the top of a jar lid in much the same way as one would use a towel in trying to increase the friction necessary. The businessman then arranged for injection-mold tooling so that these jar lid openers could be mass produced. He labled his product "Screwy Lewy" (TM) and proceeded to open up channels of distribution through West Coast supermarket chains. For a while he was successful in marketing as many as 50,000 units per year. Then, without warning, one of his major accounts stopped ordering Screwy Lewys. When our businessman friend called the supermarket buyer to find out what the problem was, he received a disconcerting answer.

It seemed that the supermarket purchasing department had computerized its operations. Instructions were furnished to the computer programmers to the effect that any vendor supplying less than $25,000 worth of merchandise each year would be dropped from the vendor list. Since our friend had only one product that he marketed through supermarkets, his sales volume could not possibly approach this figure. However, he analyzed the market and discovered that another manufacturer was continuing to provide supermarkets with a broad line of specialty products compatible with his own, such as corn skewers, egg timers,

measuring spoons, small oven thermometers, and other kitchen gadgets. Each of these products was mounted on a cardboard display sheet and held in place by a form-fitting plastic covering known as a bubble pack. In the top center of each cardboard package there was a hole. All the gadgets were then hung on J-shaped hooks which were in turn mounted on a large pegboard display on the supermarket shelf. Independent distributors known as "J-hook jobbers" visit the supermarkets periodically to take inventory and replenish the stock on the J-hook display board. Slow-moving products are removed from the board permanently and replaced with other potentially faster moving items.

Our businessman friend telephoned the president of this J-hook corporation and made the following proposal: "I'll furnish you with all of the tooling necessary to manufacture Screwy Lewys. You may have the tooling free of charge provided you pay me a royalty of 5¢ for each Screwy Lewy you sell." Since our friend was averaging 50,000 units a year, the larger manufacturer was only too happy to expand his product line to include the Screwy Lewy, and our friend now receives approximately $2,500 per year in royalties on this product, without having to expend any additional effort. The important thing to understand here is the willingness of the larger manufacturer to add a complementary product to his existing line. The addition of this product to his line did not necessitate the hiring of any additional sales force. In fact, his salesmen and the "J-hook jobbers" in the field simply had one more item in their product line that they could offer to the customers they were already serving. Many small and medium-sized businesses are willing to add complementary products and services to their current offerings in order to expand sales volume to their existing clientele.

Consider the case of the soft-drink vending machine. In most cases, soft-drink vending machines are provided by the leading manufacturers of cola beverages. These machines frequently are able to handle five or six different beverages. It wouldn't make sense to limit the potential customer selection to just one cola beverage, so the vending machines handle such other flavors as orange, root beer, and grape. Typically, the brand name of these additional beverages is not seen in any other channel of distribution, such as a supermarket. Recognizing that his vending machine represents a virtual monopoly as a channel of distribution, the soft-drink bottler prepares these off-brand beverages as a way of expanding his product line to fill his existing channel of distribution rather than going to a competing bottling company to secure from them what might be a better-known brand of root beer, grape soda, or orange soda.

Most small companies, however, are not in a position to expand their product line readily. In many cases, such expansion represents a substantial investment in research and development, time, and money. The alternative to this investment is an increase in the channels of distribtuion utilized to bring the product or service to the marketplace. However, even before an intelligent discussion of channels of distribution can be presented, it is necessary to discuss briefly the concept of market segmentation. Market segmentation is the term used to describe a method of market analysis in which the various types of potential customers for your product or service are identified. For example, you may wish to segment your market on the basis of demographics. Are your products or services intended for young children, teenagers, or retired people? You may wish to segment your market geographically by state, county, city, etc. Or you may segment your market in terms of average income, occupational area, hobby interests, political viewpoint, etc. Before selecting the best candidates for channels of distribution for your product or service, it is imperative that a market segmentation analysis by completed. In this way, you can identify the channels of distribution most efficient for serving the market segments you hope to reach.

Four Keys to Channels of Distribution

Before analyzing any channel of distribution, the marketer should develop a full appreciation for the important tools that are available to him. These keys are books, industry publications, trade associations, and mailing lists and directories. Let's analyze each of these keys separately.

Books. Every year in America, 40,000 new books are published. The subject matter of some of these books is extremely specialized. Virtually every major channel of distribution is represented by one or more books. Consider, for example, supermarkets as a channel of distribution. The amount of products sold through supermarkets daily is enormous. Many people fail to realize, however, that more than half the products sold through supermarkets are non-food products. Have you ever tried to eat a light bulb? Have you ever drunk a bleach cocktail? A book titled How to Sell the Supermarkets for Non-Food Manufacturers/Distributors by Julian H. Handler, published by Fairchild Publications, Inc., New York, provides a detailed discussion of this important channel of distribution. I mention this work as just one example of the specialized information available to marketers desirous of entering new channels of distribution. A two-volume work revised annually and titled Books in Print, published by the R. R. Bowker Co., New York, provides a listing of almost all books in print available in most libraries and book stores.

Industry Publications. Several thousand industry publications serve the various channels of distribution and marketplaces in this country. Let's say, for instance, that you wanted to market a piece of costume jewelry designed for women tennis players--say a

small gold-lated tennis racket with a small pearl mounted in the web of the racket. Let's check to see what sort of publications might be available to assist the marketer in marketing such a product. First of all, we might consult the publication known as Business Publication Rates and Data, a monthly publication of Standard Rate and Data Service Inc., Skokie, Illinois. Under the head "Jewelry and Watch Making," we find the following trade publications: American Horologists/Jeweler, American Jewelry Manufacturer, Catalog Showroom Business, Fashion Accessories, Independent Jeweler, Jeweler's Circular Hyphen Keystone, Jewelers Digest, Modern Jeweler, National Jeweler, Northwestern Jeweler, Pacific Goldsmith, and Southern Jeweler. Under the heading "Sporting Goods," we find magazines Tennis Industry and Tennis Trade. In the Standard Rate and Data Service Publication Consumer Magazine and Farm Publication Rates and Data, we find, under the heading of "Sports," the magazines Tennis, Tennis for Travelers, Tennis Times, Tennis, U.S.A., Tennis West, and World Tennis. As you can see, our tennis jewelry manufacturer has a number of industry and consumer publications available to provide him with detailed information on how to establish channels of distribution to reach this enormous market.

3. Trade Associations. Thousands of trade associations exist to serve the needs of various industries. Many trade associations serve the highly specialized channels of distribution which exist in many retail outlets; for instance: the Association of Air Conditioning and Refrigeration Wholesalers, the National Appliance and Radio-T.V. Dealers Association, the National Auctioneers Associations, the Bicycle Wholesale Distributors Association, the National Retail Merchants Association, the Cosmetic Industry Buyers and Suppliers Association, Independent Grocers Alliance Distributing Co., National Wholesale Furniture Association, National Glass Dealers Association, National Retail Hardware Association, National Wholesale Hardware Association, Manufacturers' Agents National Association, Society of Manufacturers' Agents, Museum Store Association, National Association of Chain Drug Stores, National Premium Sales Executives, National Wholesale Druggists' Association, National Association of Sporting Goods Wholesalers, Textile Salesmen's Association, American Toy Export Association, National Vitamin Distributors Association. These organizations represent just a sampling of the trade associations serving the channels of distribution through which products reach the marketplace. A comprehensive listing of trade associations may be found in the Encyclopedia of Associations, edited by Margaret Fisk and published by the Gale Research Co., Detroit, Michigan.

4. Mailing Lists and Directories. The final key to channels of distribution is mailing lists and directories. There are mailing lists and directories for a remarkable array of channels of distribution. One can, for example, secure specialized mailing lists of salesmen of all descriptions--salesmen specializing in pet products, salesmen specializing in pharmaceutical products, salesmen selling lumber products, etc. Furthermore, there are directories of retail outlets of all descriptions, such as chain department and discount stores. The two primary publications of interest are Direct Mail List Rates/Data, published by Standard Rate and Data Service Inc., and The Guide to American Directories, edited by Bernard Klein and published by B. Klein Publications Inc., New York. Now let's see how these four keys to channels of distribution may be used to unlock two important channels of distribution--direct selling and direct mail. Both of these examples are intended to illustrate a method of analyzing channels of distribution.

Direct Selling

Direct selling or direct-to-home marketing systems, as they are called in university marketing courses, are utilized for the purpose of merchandising such products as vacuum cleaners; tableware and cookware; reference books such as encyclopedias and religious sets; costume jewelry; clothing of all description; plants, trees, and shrubs; insurance; mutual funds; electrical appliances; water softeners; portraits; frozen foods; greeting cards; toys; giftwares and novelties; cosmetics; rug upholstery and drapery cleaning; and miscellaneous housewares. According to the trade association serving this industry, more than $10 billion worth of merchandise a year is sold through direct-to-home marketing organizations. The trade association serving this industry is the Direct Selling Association, headquartered in Washington, D.C. A number of trade publications serve this channel of distribution, including Salesmen's Opportunity Magazine, Spare Time Magazine, and Specialty Salesman Magazine. If you wish to secure a mailing list of individuals interested in selling your product directly to consumers, you could start by securing the names of subscribers to Spare Time Magazine. This mailing list can be rented from Dependable Lists, Inc. in New York. Many books have been published on the subject of direct selling. Some of the titles are: The Direct Salesmen's Handbook, by Roy Alexander; How Women Can Make Up to $10,000 a Week in Direct Selling, by Claire Cox; A Foot In The Door, by Alfred C. Fuller; Your Future in Direct Selling, by Foster E. Goodrich; Consumer Attitudes Toward Direct-To-Home Marketing Systems, by Marvin A. Jolson; and Selling Direct to the Consumer, by Robert C. Paddy, Albert Haring, and Harvey L. Vredenburg.

One very successful direct marketing company is Unique Products, headquartered in Chicago, Illinois. The hottest-selling item every offered by Unique Products is called the Tru-Scent Decorator Floral Bowl. Unique Products has now sold more than 2 million of these products through a small army of direct

salesmen. The typical salesman is recruited by a full page ad appearing in one of the three direct marketing publications mentioned above. The Tru-Scent Decorator Floral Bowl has a retail price of $1.98. The product is a small, attractive, clear glass container with an ornate lid. A plastic rose is inserted inside the container along with a "magic fragrance capsule" that serves as an air refresher and room scenter. The order blank in a typical advertisement says: "Here's my dollar. Rush me full-size demonstrator floral bowl, 7-inch-diameter floral ring, showing all six colors, blister card with three magic fragrance refill capsules, 8-page success story brochure, complete sales plans, samples of sales aids, etc. If I can't make instant profits of up to $10.00 to $20.00 an hour just showing this money-making kit, I may return it for triple my money back ($3.00)." Utilizing the direct sales channel of distribution almost exclusively, Unique Products has sold more than 2 million of its Tru-Scent Decorator Floral Bowls. Please note that this product is not patented, copyrighted, or trademarked. The success that Unique Products has had in merchandising this product is simply due to their knowledge of how to reach the marketplace effectively through this important channel of distribution.

Direct Mail Marketing

Billions of dollars worth of products are sold every year through direct mail, including books, magazine subscriptions, financial services, newsletters, shower heads, office supplies, pocket calculators, automobiles, steaks, grapefruit, wine, jewelry, home-study courses, clothing, art prints, phonograph records, audio tape cassettes, and coffee machines for small offices. A leading trade journal that serves the direct mail marketing industry is The Reporter of Direct Mail Advertising, the Magazine of Direct Marketing, published by Hoke Communications in Garden City, New York. The industry is served by the Direct Mail Marketing Association, headquartered in New York. Dozens of books have been written on the subject of direct mail marketing. For example, we have Direct Mail and Direct Response Promotion, by Christian Brann; The Robert Collier Letter Book, by Robert Collier; Direct Mail, by H. Hoke; Planning and Creating Better Direct Mail, by John D. Yeck and John T. Maguire; How to Start and Operate A Mail Order Business, by Julian L. Simon; The Amazing Mail Order Business and How to Succeed in It, by Howard Sparks. Mailing lists of all types and descriptions are used in direct mail marketing.

Many of the most creative marketing practitioners work in the medium of direct mail. Consider the case of a mink rancher from Minnesota. One hot summer, after an unusually warm winter, the rancher found himself stuck with an enormous surplus of mink coats. Obviously, it is not easy to sell mink coats in July. However, his financial condition demanded that he do something to liquidate his inventory in order to pay an outstanding bank loan. Almost in desperation, our mink rancher called in a direct mail marketing consultant. I think you will agree that devising a way to market a large quantity of mink coats in July is a particularly challenging task. However, our direct mail marketing consultant devised a brilliant plan. He was aware of the existence of a mailing list of expectant mothers. This particular list is maintained by months of pregnancy, the expected birth date of the child, and other information. In case you're wondering how a mailing list is originated, maternity patients may find at their doctor's office an order blank inviting them to send for a free "Lady in Waiting" gift package of samples and literature. In order to receive this gift package, the expectant mother provides her name, address, and the anticipated birth date of the child. The information thus provided is placed on a computer. Our direct mail consultant prepared a letter addressed to the husbands of women who were seven months pregnant. The gist of the letter was as follows: "Dear Mr. So and So. Your wife is about to present you with a new heir. Isn't this a marvelous gift for you? In return, wouldn't it be nice if you were to present her with a gift every woman wants--a new mink coat?" Needless to say, the direct mail campaign was remarkably successful. In fact, every year the Direct Mail Marketing Association conducts a competition for the most clever direct mail marketing ideas. The direct mail consultant responsible for the mink coat sales received the "Gold Mailbox Award" for his efforts.

Whether you are in the manufacturing, service, wholesale, or retail business, there are an enormous number of channels of distribution through which you can reach your marketplace. In addition to direct mail and direct marketing channels, businessmen reach their markets through discount stores, consignment stores, original equipment manufacturers, licensing, premiums, incentives, advertising specialties, factory outlets, commission representatives and agents, mail-order catalog houses, catalog showrooms, direct response advertising in print media, radio and television, wholesalers jobbers, distributors, trade shows and exhibitions, private label manufacturing, retail outlets, cooperative mailings, export markets, flea markets, department stores, variety stores, specialty stores, supermarkets, clothing stores, drug stores, toy stores, hardware stores, auto supply stores, garden supply stores, vending machines, etc.

In this paper, I have tried to demonstrate a method of approaching the marketplace. Although several examples were given, it is important that you understand the concepts employed to reach the marketplace. The application of these concepts to your own particular marketing needs is up to you.

Recommended Reading

1. Boyd, Harper W., Jr., and Massy, William F. Marketing Management. The Harbrace Series in Business and Economics. New York: Harcourt Brace Jovanovich, Inc., 1972.
2. Handler, Julian H. How to Sell the Supermarkets: For Non-Food Manufacturers and Distributors. 3rd ed. New York: Fairchild Publications, Inc., 1966.
3. Lewis, Edwin H. Marketing Channels: Structure and Strategy. Perspectives in Marketing Series. New York: McGraw-Hill Book Company, 1968.
4. McCarthy, E. Jerome. Basic Marketing: A Managerial Approach. 5th ed. Homewood, Il.: Richard D. Irwin, Inc. 1975.
5. Mahoney, Tom, and Sloane, Leonard. The Great Merchants: America's Foremost Retail Institutions and the People Who Made Them Great. 1st ed., rev. New York: Harper & Row, Publishers, 1974.
6. Ryan, William T. Principles of Marketing. Programmed Learning Aid Series. Edited by Roger H. Hermanson. Homewood, Il.: Learning Systems Company of Richard D. Irwin, Inc., 1971.

BASIC AND THE BUSINESS COMMUNITY

Richard E. Barnhart
Analyst-Programmer
Levi Strauss & Co.
3903 Canon Ave.
Oakland, CA 94602

Abstract

BASIC offers little of positive value in a mid- or large-size Data Processing shop. As a learning tool it is quite valuable, but its syntax, I/O, and report writing capabilities are too limited for most business applications. In some areas of business, though, BASIC is being used successfully today. In the near future there promises to be a significant increase in the use of BASIC in business --- but outside of Data Processing shops. This holds interesting implications for both hobbyist and "professional" programmers.

When I mentioned to some of my colleagues the possibility of delivering a paper on BASIC and business I was met with a mixture of uncomprehending disbelief and great good humor. In a medium size programming shop with about 100 analyst-programmers, an IBM 370-158 and a 370-168 plus mainframes and minis in our various manufacturing and distribution facilities, BASIC has no place. But I am something of a renegade, a revolutionary. I have a micro at home; at work I enjoy doing maintenance, not new system development; and I'm always looking for other better ways to process data.

Here in this world of micro-computers, you have BASIC. Do you have any idea what it is in terms of the business world? Do you know what is being done with it now? What will happen with BASIC during the next 10 or 15 years? These questions revolve around the language in general, not one particular implementation. These questions apply equally to mainframe, mini-, and micro-computers. These questions stand between the formal language known as BASIC, a way to use computers, and businessmen who depend on computers.

What is BASIC in relation to the marketplace? Business is interested in data, not computation. In the business world there are Data Processing Departments and DP Centers as opposed to computer centers and programming staffs. This distinction is not iron-clad, there actually are some computer centers and some companies do have a staff of programmers, but the distinction between data processing and computing is the essence of the business use of computers. In the business world, computers are primarily tools to lighten the clerical workload and allow speedy correlation of masses of data and facilitate better business decisions. Ford makes cars and trucks. Ford's DP establishment exists solely to aid Ford's task of making cars and trucks.

Over against the business approach to data processing, schools and colleges have computer centers to help people learn how to use computers. Scientific establishments such as NASA have computers which actually compute esoteric mathematical problems. Home computing is mostly a leisure time activity peripherally associated with education, or perhaps with entertainment. All of these areas comprehend computers more as an end in themselves than as another tool to get work done more quickly and efficiently.

In the context of data processing, BASIC seems at first glance to be a rather poor choice among the available languages. BASIC is simple, a subset of FORTRAN, and most at home in mathematical calculations or conversing with a user at a terminal. DATA processing has been built on the concept of associating many different fields of data containing diverse kinds of information into single records. Records are arranged by customer (e.g. a sales file), by product (e.g. inventory or manufacturing files), by individuals' names (e.g. personnel files), or by some other significant key. The point is that these records comprise many different kinds of information which can be processed as a unit. Data bases, in the last analysis, are simply very sophisticated record/file structures.

Most implementations of BASIC ignore the concept of records as conglomerates of data fields. A$ is a string variable. It can be displayed on a terminal, read from an external device, or written to the same. If A$ represents a person's name and SS represents his Social Security Number, the two must be read or written individually. The overhead involved in reading and writing masses of fields for one record inhibits the use of BASIC as a serious contender in the area of DATA processing.

Some implementations of BASIC are more amenable to record oriented processing than others, it is true. Hewlett-Packard and North Star BASICs use substringing to address segments of records. Substringing, or addressing particular characters in a string variable has the major

advantage of being able to insert new data into existing string variables. The DEC/Microsoft approach of LEFT$, MID$, and RIGHT$ allows breaking apart an existing string variable, but not modifying it in place. In general, then, BASIC sould be seen as more of an analytical language rather than a processing language since there may be difficulty in modifying a part of an individual record.

There is a definite place for analytical languages in the marketplace, but to date, BASIC has not entered the arena to any great extent.

Analytical languages such as RPG II or MARK IV are very popular in business because of their ability to quickly and easily format reports as well as analyze data. The end product of data processing, in business at least, is succinct and comprehensive reporting, and very few implementations of BASIC have any easy method of formatting printed reports. BASIC is primarily an interactive, terminal oriented language designed to present the user with quick, short answers. It is possible to implement business systems such as billing, sales history, or other applications in BASIC, but the primitive formatting capabilities of the language militate against it.

The ultimate business test of any language is cost effectiveness. The cost of data processing during the short history of computers has completely reversed itself. Originally the machines were extremely costly and quite limited in their capabilities. IBM's 1400 series was the first computer family to really have any wide use in business because of the cost/capability tradeoff. At this time though, 70% to 80% of the cost of data processing is involved in salaries for programming, for running the computer, and for maintaining the equipment. Furthermore, recent studies indicate that during the 5 to 8 year life of a business system, only 40% of the cost is involved in developing and writing the system while the other 60% goes to maintain the software after it is up and running.

While BASIC is one of the easiest languages to learn, its very simplicity makes it difficult to use in systems of any size. So-called "business" programs published in the hobby literature illustrate this point. BASIC programs longer than 200 statements begin to look like piles of spaghetti. Internal documentation must be achieved through REM statements. Variables, because of their length limitations, are arbitrarily assigned and are extremely difficult to comprehend in a large program. It must be admitted, though, that maintaining and modifying a BASIC program is a true challenge when compared to the relatively mundane and boring task of maintaining COBOL programs.

Business programs are expected to have a lifespan of over 5 years. I have yet to see one business program which has not been modified to some extent during the first three years of its life. When a program's internal documentation is limited to REMark statements, and the names of the variables have no intrinsic meaning, you can see that the cost of maintenance would not be 150% of the development cost, but would be more on the order of 500%. In terms of maintainability, and ultimately cost effectiveness, BASIC must bow to a COBOL-like language.

One further point about maintainability is the BASIC language itself: there is not one BASIC language, for they are legion. Digital Equipment Corp. (DEC) and Hewlett-Packard have disparate approaches to BASIC. Datapoint, G.E., and IBM all disagree about what should and what should not be included. North Star, Microsoft, Benton Harbor, and TRS-80 BASICs share the name, but are in no way compatible. This is not to say that a program written in COBOL on a Burroughs machine can be compiled without change on Control Data equipment, but the changes needed are minimal. The difference between COBOL and BASIC is that there is a standard COBOL. The American National Standards Institute has published and updated its COBOL standard for a number of years. Each manufacturer offering a COBOL compiler uses relatively the same format to describe the language (this is covered in the standard), and the non-standard features are clearly marked. The ANSI committee on BASIC has only recently published a proposed standard for a first level BASIC, one which will, if accepted, rank among micro users as a Tiny BASIC.

When programmers are faced with a variant on a language they have used and loved (or hated), they will make mistakes. The learning curve and the mistakes all cost money. Until there is a relatively stable standard BASIC accepted and implemented by most manufacturers, the portability and maintainability of BASIC programs will be in serious question by the business community.

Does this all mean that by its very nature BASIC is not viable in business data processing? In the classic sense of data processing, the answer is yes. However it should be noted that the Classical period of data processing ended with the introduction of the System 360 by IBM, and with the advent of powerful, low cost microcomputer systems, we seem to have entered a "post-industrial" phase in data processing.

When IBM introduced the System 360 nearly 15 years ago, the cost of computing dropped nearly an order of magnitude and <u>data</u> processing, as opposed to computation, became really viable. Companies with only a few million dollars in sales could afford a small mainframe while those with sales as small as the hundreds of thousands of dollars could afford time sharing or a

service bureau. The world of programming was growing and programmers were needed.

Generation 3, Version 1: COBOL and FORTRAN emerge out of the chaos of Autocoder and Assembler, but powerful as the new languages are, people cannot be trained in them quickly. The time is 1964, the place, Dartmouth: Enter BASIC. Born out of and borne on the crest of the computing power so recently introduced, BASIC, developed by Kemeny and Kurtz, stands forth as a teaching tool, an introduction to FORTRAN and ALGOL. BASIC is not intended as a replacement for these languages, nor is it seriously considered as even an introduction to COBOL. It is intended to help people quickly learn how to use computers.

The success of BASIC as an educational tool can be measured by counting on the fingers of both feet the number of schools with a computer curriculum which do not have BASIC, in some form, implemented on their system. The success of BASIC as an educational tool can be traced to two characteristics: it is very simple; and for all of its simplicity, it is a relatively powerful, high-level language.

Historically then, BASIC entered about the time that IBM's second major breakthrough occurred. The Classical age of computing with its heroes and Hoppers, its enigmas, ENIACs, and 1401s, was drawn to a close by the humdrum, day-to-dayness of business problems. But far more people were being introduced to computers than were entering programming. Business majors and science majors were required to take at least minimal computer introductions (and learn BASIC). On some campuses Geology, History, and Music found their way onto the Machine. In all of these areas, BASIC was most often the first language.

Microcomputers have followed the mainstream, and wisely so. THE language shown to be most usable by more people is the one which has been implemented on most of the existing systems. BASIC interpreters and compilers, on paper tape, cassette, and disk as well as ROM, have flooded the field. BASIC is being used to write systems software, interpreters, compilers, and assemblers, as well as games and small business systems. This is not to say that BASIC is the best choice of language for many (any?) of these tasks, but it is being used because it is available, and it is powerful enough to handle the load.

At this time the most spectacular applications in BASIC are on micros. This can be traced directly to the fact that BASIC, in one form or another, is available on most micros, and nothing else but assembler or machine language is available. On mainframes, professional programmers use FORTRAN and COBOL to compute or process data relatively efficiently. These languages haven't generally become available on micros and there are a limited number of programmers (professional or amateur) who are willing or able to write large scale systems software without the incentive of a salary. BASIC has taken first place with micros by default.

But there are other uses and users of BASIC today as well. Many schools use time-sharing services in their computer curriculum. These service companies generally implement BASIC in order to satisfy their educational users. When BASIC is available to one user through CYBERNET, for example, it is available to all. Auditing and accounting firms commonly use timesharing services, but few of them can afford the luxury of a full-time programming staff. The same people who have been trained at a basic level in BASIC in order to fulfill their Business prerequisites are using BASIC to solve their auditing or accounting business problems. As noted above, BASIC is probably one of the least effective languages to use on business/accounting/auditing problems, except for one thing: the people who have the questions also have BASIC. The language is easy to use, it doesn't require years to learn, and in non-repetitive kinds of jobs, its inefficiency can be ignored in the context of the fantastic computing power of contemporary mainframes.

Engineering firms also commonly use BASIC, probably to the disgust of the engineering faculties which spent semesters trying to drum FORTRAN into the heads of their burgeoning students. The fact is that their students wanted to be engineers, not programmers. It takes at least three years to develop reasonable facility with FORTRAN. Full time. An engineer being paid for his abilities in analysis or design does not have the leisure to develop expertise in computing tools. He has a job and is paid on the basis of how well and quickly he finishes. The relapse into BASIC is normal and apparently frequent. Very often a "quickie" program can be written better in BASIC than in FORTRAN because BASIC is the simpler language and errors are less likely to creep in.

On mainframes and time-sharing systems, there are existing business users of BASIC. Since these people are programming for themselves instead of turning to professional programmers or systems analysts, they save their companies money (salaries) and get the job done more quickly. BASIC is probably a poor choice of language for their problems except that they really don't have a good grasp of any other language. Therefore BASIC is their best choice.

The combination of managers trained in BASIC during college, high school, or even grammar school, and the availability of BASIC on time-sharing systems, is already bringing a change to data processing. Accountants, engineers, and managers are beginning to ask computers more and more complex questions --- directly.

In terms of professional programmers, BASIC is not their death warrant. The need for well written, efficient programs organized into processing systems remains. BASIC has too many limitations to seriously contend with either COBOL or FORTRAN, let alone a language as sophisticated as PL/I. The mainstream of scientific computing and business data processing will, of necessity, remain with the established languages.

On the other hand, professional programmers will be called upon more often to analyse and debug BASIC programs for their end users, the sales administrators, the personnel specialists, and occasional VP's who want to get answers NOW. Professionals will need to know BASIC in order to maintain credibility outside of their own department. As more people who have studied a little about data processing in school move into upper management, the trend of non-professionals using computers will only expand.

What I've said today is really rather simple, even mundane. BASIC is here, many know how to use it well enough to gather information with it, and these people are moving into positions in business where they will use it. The corollary is that professional programmers will have to know BASIC as well in order to retain their credibility as at least competent technicians. This is hardly a world-shaking premise, but it needs to be said because no one is saying it. It needs to be said to remind us where the real world is.

APL, PASCAL, FORTH, and PILOT as well as other languages are repeatedly the subject of glowing reviews in computing literature. The reason that they are news is that they are literally new --- very few people use them or have any experience with them. Who writes about BASIC? Who needs to? Everyone knows it or can learn it in a day. It isn't as powerful as APL, it's more awkward than PASCAL, and it's harder to learn than PILOT. BASIC is just what its name implies: the bread and butter of computing. It is very simple fare, but quite good enough to keep body and soul, or information and profit, together.

Understanding where BASIC stands in business is important to people who are or want to be in data processing just as understanding where COBOL stood in business was important about six or eight years ago. Then PL/I stood like a giant on the horizon, backed by grandaddy IBM. PL/I should have replaced COBOL and superceeded FORTRAN in months if not days. But more people knew COBOL and FORTRAN, and managers could begin to understand these languages. The business standard today is COBOL; the scientific standard is FORTRAN. Only a few shops use PL/I, and PL/I programmers are often locked into their jobs with very little opportunity to find greener pastures.

The more forward thinking people in data processing were surprised and frustrated by the strength of COBOL and FORTRAN, but they were only looking at the technical potential of the languages. The technical potential of BASIC is a similar case in point. No matter what aspect of BASIC you examine, at least one language is technically better. Some languages are technically superior to BASIC on all counts. The fact remains that more people know BASIC than any other language, and that many with knowledge of other languages choose to use it for particular tasks because of its simplicity.

A realistic appraisal of the role of BASIC in business is that it is being used now for many tasks, sometimes surprising tasks. More people have learned it than any other language and with the advent of personal computers, even more will become proficient in it. As managers and even clerks become more sophisticated in their uses of computers, BASIC will rise in popularity far beyond its present level. Like COBOL, it's facilities will expand somewhat, but it will remain a simple language. Professionals will do well to heed this trend, and those with personal computers are already one step in the right direction.

CIS COBOL BRINGS BUSINESS TO MICROS

by

Paul O'Grady
Micro Focus Ltd
London, UK

Introduction

This paper starts by surveying the programming languages that are available on microcomputers, and the merits and demerits of each are briefly described. All of these have been used in the past to write business applications on microcomputers. The result of such a survey leads inevitably in the opinion of the author to one language, namely COBOL.

In the second half of the paper, CIS COBOL, a specific implementation of COBOL for microprocessors, is described. This is a product developed by Micro Focus Limited in the UK. It is resident on the target microcomputer and contains a number of extensions to COBOL which are aimed at the small machine environment.

Available Languages

In assessing the merits and demerits of each language I will not discuss their detailed syntax and statements. The interested reader has only to consult the appropriate reference manual for detailed and accurate descriptions which are far too voluminous for a paper of this type. What I will do is to highlight those points which have a major impact on the decision of which language to choose.

Until quite recently, the most likely choice of language for any application on a microprocessor would have been Assembler Language or even Machine Language. The decreasing cost of hardware, and in particular memory, coupled with the increasing power of the microprocessors themselves has now radically altered the balance between development costs and production costs. The result is that Assembler Language need now only be used when it is necessary, and the rest of the application can be written in the appropriate high level language.

Since Assembler Language is simply a mnemonic version of Machine Language it is most appropriately used when precise timing loops are required, or when store is at a premium as in large volume applications. The application programs for such applications are usually small.

The high level language most commonly used to date in microprocessor applications is probably BASIC. This was originally developed as a teaching aid at Dartmouth College and has proved to be very popular with hobbyists. It is a very active language and there are now many variations of it in existence, including extensions for business applications.

A particularly appealing characteristic of the language is the ease with which a program may be developed. The programmer may key in his program and run it immediately.

Following close behind BASIC in generality of use are the many derivatives of PL/1, (PL/M on the 8080, PL/Z on the Z80, MPL on the 6800 and so on). These usually have good control structures which allow structured programming techniques to be used. On the other hand they usually have limited data types. They are good as assembler replacements and are used by the professional rather than the hobbyist programmer.

Another well-known language which is now becoming much more common on microprocessors is FORTRAN. In the past, this has been used extensively to write cross compilers and cross assemblers for micros and it has been extensively used on mainframes and minis for scientific work. It has limited I/O and limited record structuring and is intended more for the 'number crunching' type of application.

The list of languages might continue with PASCAL, a language which may eventually become a standard system programming language and APL for the more mathematically oriented. By this time, however, we are getting a long way from languages which could be said to be 'ideally suited' for business applications. Even so, all of the languages cited above, with the possible exception of PASCAL, have been used to do just that.

Until very recently, the list of languages available on microprocessors would not have included COBOL. This is the language which was designed for business applications and has been an industry standard for 20 years. It has extensive data structure facilities and sophisticated I/O. It is above all the language that all programmers know.

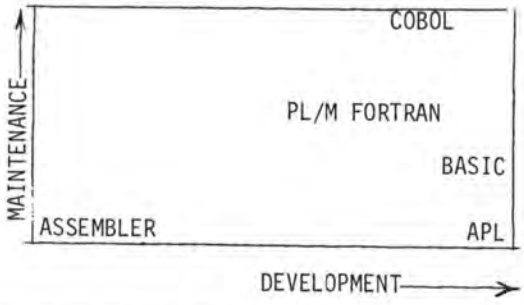
Three Criteria for Language Comparisons

Before looking at a specific implementation of COBOL in some detail, I will complete the comparison of the various languages with reference to three criteria, namely,

* ease of use
* style
* type of application

Ease of use can be split into the two main categories of maintenance and development (see Figure 1).

Figure 1:

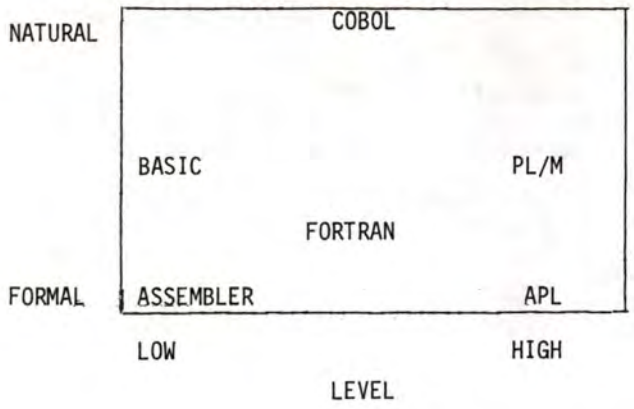


Ease of maintenance is associated with the life of a system, both in terms of hardware and software. It is probably true to say that many COBOL programs have outlived the hardware the were designed for, whereas at the other end of the scale, an assembler program is normally tied to a particular processor. The person who carries out the maintenance is also important. If it is not the person who wrote the program, good documentation is essential. COBOL is an English-like language designed to provide programs which are self-documenting.

Ease of development is associated with the speed with which programs can be compiled and tested. Pure compilation speed is important, but so also are the error handling in the compiler and the debugging facilities available to the programmer. Interactive, symbolic debugging is the ultimate requirement.

Ease of Maintenance and Development are in may ways simply reflections of two characteristics of the language itself, namely its legibility and level. These are shown in Figure 2 under the generic heading of style.

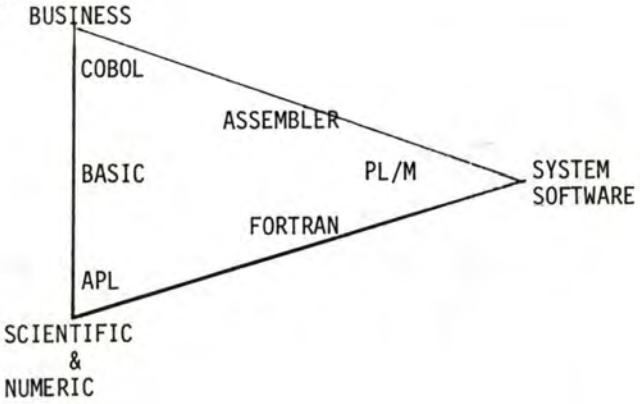
Figure 2:



Legibility is plotted in a range from formal to natural language. Probably the ultimate in formality is APL for which it is sometimes said that the only way to amend an APL program is to write another one. On the other hand it is a very high level language.

The final criterion to compare the languages against, is that of type of application (see Figure 3).

Figure 3:



Business applications typically require little computation, a lot of I-O and structured data. Scientific and numeric applications on the other hand require a lot of computation and little I-O. Finally, system software requires little computation, powerful control structures, flexible data handling and last but not least, access to the hardware registers, ports etc.

Assuming these characteristics for the three types of application the placement of the languages in Figure 3 is self explanatory.

This comparison of the programming languages that are currently available attempts to give an objective basis for choosing which to use for a business application. It assumes that all the languages start equal. Even on these terms COBOL is very much the first choice.

However, the languages do not start equal. All the surveys show that the vast majority of business programs are written in COBOL. In order for microprocessors to be cost effective in this area it will be necessary to protect this software investment.

One can foresee therefore COBOL being the automatic choice for all business applications on micros, once it has become more widely available.

Differences Between Mainframes and Micros

CIS COBOL has been implemented initially for the Intel 8080, 8085 and Z80 microprocessors and runs under the ISIS and CP/M operating systems. It is however, easily portable to other microprocessors and operating systems.

Portability between different microprocessors is an extremely important characteristic in pratice, since new ones appear so regularly. For the purposes of this paper however it will not be stressed further, and all microprocessors will be assumed to be the same.

What is important is not so much the difference between microprocessors themselves, but the difference between the microprocessors and the mainframe environment, where hitherto, COBOL has been found. The CIS COBOL extensions have been designed to deal with this new environment.

What then are the differences between the two environments? The main ones are outlined below:

* operator interaction
* programmer close to the machine
* COBOL is not the programmer's main language
* limited store
* ROM

It was by designing a compilation system to cater for these differences, that the CIS COBOL extensions came into being. Before looking at them in a little more detail however, it is worth stressing the major similarity between mainframe and micros, namely the need for standard ANSI COBOL, in order to use the existing fund of programmers.

General Description of the CIS COBOL Compilation System.

The CIS COBOL Compiler generates object code for a COBOL "virtual machine". This code is designed for optimum representation of COBOL verbs and data types. Instructions are variable in length so that more frequently used constructs are shorter than less frequent ones. A very compact form of the code is used for the "computational subset" which is the form of COBOL in which the Compiler itself is written. The code in which the Compiler is written and the code which it generates are both interpreted by a common Run Time System. This consists of an interpreter which emulates the COBOL virtual machine and two interfaces to the Operating System and CRT.

The Compiler operates in a single pass over the source code. Forward references are fixed up at the end of compilation in a table added to the end of the procedure code. The whole process is carried out by the Compiler itself; there is no need for a separate consolidation phase. This means that the Run Time System is able to load the output from the Compiler directly in its "fast load" facility, thereby shortening the application program development cycle.

When an application program has been tested it is possible to link it permanently to the Run Time System to form a free standing loadable program. Any linking convention may be supported. The Compiler itself is just such a program. It is compiled with a variant of itself in which additional code has been included to handle the computational subset, and it is then linked to the same Run Time System as that used for application programs. In the case of the overlayed form, some optional modules of the Run Time System which the Compiler does not use are omitted from the linkage to save space. In the ROMable form, all are included so that both the Compiler and COBOL application programs may share access to them.

The whole CIS COBOL system is totally resident on the target microcomputer. The Compiler will run in a 32K Byte microcomputer including operating system and a 5K Byte Dictionary for itself. This same configuration will support application programs of the order of 5000 source statements. In order to transport the software to another microprocessor it is only necessary to rewrite the Run Time System, since the Compiler is also written in COBOL and is therefore machine independent.

Operator Interaction

Probably the single most important difference between the mainframe and micro environments is the degree of interaction. This shows itself both during the

development of programs and also in the running of the application programs themselves.

The programmer develops his microcomputer programs seated at the console, operating the equipment himself. There is thus a requirement to provide him with debugging tools as an integral part of the system. To be of maximum value these tools should be interactive. They should allow him to insert breakpoints, change memory and restart the program using addresses and statements from his (high level language) application program.

Business application programs that are developed for microcomputers are of the transaction processing type. One or more messages are entered in sequence using a CRT, and files are updated immediately. The first transaction is finished (eg. its invoice is printed) before the second transaction begins.

It was with all these thoughts in mind that CIS COBOL was supplied with an Interactive Debugging Package of the type described above, and interactive screen handling.

CIS COBOL screen handling has two facets to it, namely formatting and cursor control.

The programmer can format the screen of any teletype compatible CRT into protected and unprotected fields by using standard COBOL statements. The screen layout is defined as a record in the DATA DIVISION. An ACCEPT statement nominates a record description which permits input to the character positions corresponding to variables identified by data-names. These may be separated by FILLERS to position them on the screen. Conversely, a DISPLAY outputs only from non-FILLER fields in the record description which it nominates. The programmer can easily build up complex conversations for data entry and transaction processing.

When data is being keyed in, the operator has full cursor manipulation facilities, each variable in the program acting as a tab stop on the screen. Non-numeric digits may not be keyed into fields defined as PIC 9. Finally, when the operator has checked that the data is correct, the release key is pressed and processing continues. There is no transmission delay; all characters having been input as they were keyed in.

The CIS COBOL Compilation System is intended for use with an existing text Editor, and is designed to be exceedingly simple to use - even for those brought up on a Basic. The easy option allows the user to leave out all the "red-tape" statements and thus to have simple programs up and running within ten minutes of meeting the system. This is particularly useful for building up confidence in the early stages. If on the other hand, the installation insists on full ANSI compatibility the ANS compile time switch forces only standard ANSI '74 statements to be clearly compiled.

Finally, since the Compiler generates separate segments for Intermediate Code and data, and since both the Run Time System and the Intermediate Code are re-entrant, CIS COBOL application programs may be held in ROM. For similar reasons, the CIS COBOL Compiler may itself be in ROM. Furthermore, segment contents are referenced relative to the segment start, with the result that segments may be positioned anywhere in the machine.

Summary

There seems little doubt that as COBOL becomes widely available on microprocessor systems, it will replace all other languages for business applications. Equally, there is a need to extend what is essentially a batch processing language to take advantage of the new environment.

The present version of CIS COBOL shows some of the extensions that are possible. This is only the beginning however. As microprocessor based hardware becomes more sophisticated, especially in the realms of teleprocessing and multiaccess, one can expect to see further development to the CIS COBOL language.

IN SUPPORT OF COBOL
AS THE STANDARD LANGUAGE
FOR SMALL BUSINESS APPLICATIONS

© Dick Burkhalter, President, Diverse Developments
2044 N. Kenwood St., Burbank, CA 91505, 1978

The growth of the microcomputer market has brought with it a proliferation of programming languages. One can hardly pick up a computer magazine these days without finding at least one article describing some new way of communicating with a computer. As an individual with a deep interest in computing, I enjoy the mental stimulation that comes from reading these articles, and applaud the innovators who are spending their energies on these developmental pursuits.

As a business analyst/programmer, however, I feel that the small systems world has ignored the one language that is the standard for business applications, and am compelled to speak out in support of COBOL. I feel COBOL has been unfairly maligned and ignored for long enough, and as we push ourselves into the business world, it is time to set the record straight.

I agree that COBOL is not a perfect language, and in theory, a number of other languages offer a greater degree of efficiency and power. But in the business world, theory must take a back seat to practical realities. So before you brand me as a reactionary advocate of buggy whips, I ask that you listen to what I have to say about the suitability of COBOL to solving business problems.

A Short History of Languages

In the beginning days of computing, the choice of programming languages was limited to those the computers understood. And since they only understood their own languages, people who wished to converse with computers had no other choice but to learn machine languages.

As the first computers were the province of scientists and mathematicians, it was only natural that the first symbolic languages were those suited to science and math. The few businesses which used computers in those days held little influence in the halls of science, and so businessmen suffered in silence as they waited for their printouts. And waited. And waited. And waited.

When the advent of the transistor precipitated the boom in business computing, things began to change, however slowly. But since the business was big business (and big government), change they did. At first it seemed that everyone was developing their own language, much the same as we see happening today in the micro world. The result was, of course, chaos. Seeing this, and recognizing the true need for uniformity in programming practices and the definite need to reduce the cost of program creation and maintenance, our business and government leaders met under the banner of the American National Standards Institute (ANSI) to hash out a set of specifications for a COmmon Business Oriented Language.

Why COBOL?

The main reasons for developing a standard business language were few, and quite fundemental. And they are just as viable now as they were when COBOL was originally concieved. Let's review them briefly.

Business users realized that the problems they wished to solve with computers were quite different from the problems of scientists and mathemiticians. And they also saw that these problems were common to all types of businesses. I'll discuss these factors later.

Business users also recognized that technology would work to reduce the costs of hardware, and at the same time, economic conditions would work to increase the costs of labor, most importantly, the costs associated with the professions and skilled trades. They were enxious to be able to take advantage of technology without being held at the mercy of rising labor costs. Remember, in big business, then as now, program development cycles tend to be measured in years, not months or days.

Additionally, business users recognized the economic dangers of being locked into doing business with a single vendor. They all knew how expensive it was to create computer software, and didn't wish the cost of program conversion to inhibit their freedom to change hardware.

And finally, business users wanted to get out from under the domination of their businesses by the computer professionals. They wanted to be able to understand programs themselves, and not have to rely on an interpreter who often as not, didn't even speak their language! So the specifications of COBOL were written to describe a syntax closely resembling English.

Did COBOL Meet the Goals?

Yes, for the most part. The last requirement, that of making programs understandable by non-technical types, was perhaps based on a certain degree of wistful thinking and a lack of real understanding of the system design process, of which program coding is but a small part. But the end result is that COBOL is the easiest to read of all languages. That ought to count for some points, at least.

A look at the other objectives reveals that they have been met to a great degree, and that the standardization of COBOL has paid off in bringing very real economic benefits to its users.

What About BASIC?

Now I know that there are many of you who feel that BASIC is just as viable, if not more so, than COBOL. These feelings are usually expressed by people who have come from the mini computer ranks rather than mainframers like myself. However, in my mind, the BASIC language has some serious shortcomings which place it a distant (far distant) second to COBOL as a choice for business systems development.

BASIC was developed as a teaching language. It was designed to instruct students in the rudiments of logic and computing. To that end, it has been very successful. Personally, I feel that the interpretive nature of BASIC has more to do with that success than any features of the language itself.

The problem with BASIC is that its creators never anticipated it would be used to write business application programs, and so left out of the language those very things which support the functions most common in business data processing, namely the creation and maintenance of large files of data and the production of reports based on the contents of those files. These two functions, above all else, characterize business programs, and the lack of standard methods of handling these problems is the best argument against BASIC in business. Conversely, the standardization of constructs for handling these significant functions is the greatest single argument in favor of COBOL.

Before I return to an in-depth look at the benefits of COBOL, I feel I must address one last problem which confronts the users of BASIC; the lack of standardization of the language itself. Not only are the various extentions of BASIC bereft of any commonality, but there is not even standardization of the basic BASIC. The original Dartmouth language has been so bastardized that we might as well be writing in assembler! We have DEC's BASIC-PLUS, which is not the same as Data General's Extended BASIC, which is not even the same as DG's Business BASIC. Not to mention IBM's, Altair's, Microsoft's, and on and on. This list is endless, and continues to grow.

I am aware that attempts to produce a standard for BASIC are underway, but if attempts to standardize COBOL were any indication, we are still a long time away from seeing concrete results. A lot of time and trouble could have been saved by going with COBOL in the first place.

The Nature of Business Data Processing

As I mentioned before, the two key things that differentiate business data processing from the scientific and mathematical are the creation and maintenance of files of data, and the reporting of this data in a variety of sequences and formats. COBOL supports these requirements better than any other language.

First of all, COBOL, in its standard form, not just individual vendors' extentions, supports the four most commonly used file access methods, which are:

* Sequential
* Random, by direct addressing
* Random, by relative record
* Indexed

To use any of these methods, a programmer needs only to know how to

code a few simple statements, and the language processor or operating system will take care of all the rest.

Second, COBOL allows the creation of reports, whether they be on a hardcopy printer or a video terminal, via a simple set of instructions, most of which are data descriptions. Some vendor's extentions, such as IBM's Report Writer or Data General's Screen Section, make this work even easier. But we're discussing the standard COBOL, not extentions.

Third, the ability to use meaningful names for data elements, procedure paragraphs and condition values not only makes it possible for the non-technical people to determine a COBOL program's objectives, but of greater importance, removes ambiguity of meaning among systems analysts and programmers, a very critical point, you will admit. Those of us who have been around in this profession for a while can attest to the problems we often encounter in communicating with each other, which are often just as difficult as communicating with an end user.

Fourth, COBOL programs have the ability to call other programs, and be called themselves. This facility allows subroutines to be created, compiled and tested independently, which in turn can be translated into three kinds of benefits:

* Large programs can be parcelled out among a staff, reducing the elapsed time needed to complete the job
* Independent testing allows the programming and testing efforts to proceed concurrently, which reduces programmers' unporductive time and also speeds the testing and debugging, as small programs are faster to debug
* Programs made up of commonly used subroutines occupy less space in source and object libraries, reducing wasted disk space.

Another benefit of the "modular" technique is that by using this approach, a greater degree of consistancy between programs is achieved. This means that differences in rounding, for instance can be eliminated. Report A always agrees with Report Q, which agrees with Report F, and so on. This is a benefit not to be taken lightly in business.

Fifth, and while we're on the subject of modularity, it is appropriate to discuss the phenomena known as Structured Programming. Regardless of what you may have read or heard, and with due appologies to Mr. Djikstra and Mr. Yourdon, structured programming concepts that are being widely praised as the greatest things since canned beer and sliced bread, are nothing more than the formalization of techniques that good programmers have been using for years. (And, I might add, the need to publish volumes and hold expensive seminars to explain what is in reality just good common sense, might be legitimately be taken as a comment on how far we have yet to go before we will be recognized as true professionals instead of tradesmen.)

The key issue behind the development of structured programming techniques is the need to increase programmer productivity. The need is not only to speed program development, but more importantly, to reduce maintenance time. The nature of business programs is that they are changed often and continuously, to suit the changing needs of management. The use of structured techniques, combined with COBOL's English-like syntax and use of meaningful data, procedure and condition names, provides the flexibility that business needs and demands.

The Labor Market

I've purposely saved my best and most pervasive argument for COBOL for the last. That is, the availability of trained seasoned business programmers. In a recent survey of over 5500 users of propritary software conducted by Datapro Research (1), survey respondents were asked to indicate the proportion of their programming that is done in each of the popular programming languages. The results:

Language	Frequency
COBOL	38%
RPG, RPG II	20%
FORTRAN	12%
Assembly languages	12%
BASIC	7%
PL/1	2%
APL	1%
All Others	8%

These figures show the overwhelming popularity of COBOL in the business world, certainly no surprize to those who have been trying to find good programmers for any other language.

What isn't so apparent from the above figures is that the proportion of programmers who know COBOL versus those who don't is much greater than the percentages shown. This is because the COBOL shops include the large installations which employ up to hundreds of programmers, whereas companies which use BASIC, FORTRAN or other languages are most often the small ones with a mere handful.

What this translates to in the real world is that installations using COBOL have a larger labor pool from which to draw. In today's market, this factor often spells the difference between getting a job done and not. As an independent software consultant who often uses contract programmers, I find it much easier to find experienced COBOL programmers than to get, for instance, experienced BASIC programmers. In fact, I am sometimes convinced that there are only three good, experienced BASIC programmers in all of Los Angeles, and they all work for my competitors. But after talking with them, I find they have the same problem.

The Problems With COBOL

As convinced as I am of the values of using COBOL, I have to admit that it is not all a bed of roses.

Until recently, there were no compilers available to allow the use of COBOL, so all my arguments were moot points. Now however, there are two, and I hope, more to come.

The most often heard argument against COBOL is that is inefficient (supposedly) in use of memory. This I think is more a criticism of the particular compilers than of the language itself. I would like to see some of the creativity that is now being expended in developing new languages diverted to the problem of developing some truly wondrous COBOL compilers. I feel that the greatest gains in the small business market will be the result of these efforts.

References:

(1) User Ratings of Proprietary Software, #D09-200-005, Datapro Research, Delran, NJ 08075

The Proposed IEEECS Floating Point Standard:
What It Means to Hobbyists, Engineers, & Businesses

by Tom Pittman

ABSTRACT

The proposed IEEECS Floating Point standard is examined from a user's point of view with particular emphasis on the benefits of the proposal over alternative floating point systems. Unbiased rounding and denormalized numbers yield results which are more correct and less sensitive to roundoff error. The exception handling requirements of the proposed standard reduce spurious error messages while enhancing true error reporting. Several examples from scientific and business application areas are cited. Cost considerations of the proposed standard are discussed and found to be acceptable.

INTRODUCTION

At a recent meeting of the subcommittee working on the floating point standard it was asked, "How many of those present represent actual users of the proposed standard, that is, are not now employed by manufacturers (present or future) of floating point hardware and are not primarily members of the academic community?" Of the 21 people present, only two were able to raise their hand affirmatively. It would appear that we, the users of computing machinery, have very little representation at the centers of decision-making that will profoundly affect the quality of the machines we use. Thus we are very pleasantly surprised that the proposed standard is in fact what we really want out of our machines.

What do we want floating point hardware or software to do for us? First of all, we want the answers to be Correct. Secondly, we want to know if we have blundered into some ridiculous circumstance for which there is no correct result. But we do not want to be hassled with a lot of spurious error messages reporting exceptions that cannot possibly affect the results, nor do we want to be bothered with circuitous or excessively complicated procedures to arrive at our results. Of course we want efficient through-put from the floating point system. Finally, for a standard to be meaningful, we expect not only that all implementations of the standard should produce the same results for the same data and algorithms, but also that the representations should be compatible. Not only does the proposed standard give us all these things, but the ranges and precisions specified in the standard are as good as those we are used to getting in major mainframe computers.

CORRECTNESS

I am not the final authority on correctness. I am not a numerical analyst; my business is primarily custom applications software for microprocessors, though perhaps I am better known for setting the market price for hobby software packages. But I am very concerned for the quality of my software: I can afford to promise my customers revisions free (but for the postage), because no revisions are necessary.

What does it mean to get "correct" results from a floating point processor? Obviously not all real numbers can be represented in the finite memory of a computer. In fact only a small subset of the rational numbers can be exactly represented in four or eight bytes. But every real number not larger in magnitude than the largest representable number can be approximated by the nearest representable number. We call this approximation rounding, and every time a computation gives a result which is not exactly representable, it must be rounded. To be correctly rounded means to round to the nearest representable number in some unbiased way.

No floating point hardware in common use today rounds correctly. Most do not round at all, but chop rightmost bits off in a variety of capricious ways. A few round to the nearest, unless the correct result is exactly between two representable numbers, and then they round away from zero. In the first case (chopping) you get the familiar result,

$$(1/3)x3 = 0.99999$$

which if it were correctly rounded would give 1.00000 in a binary machine. Some of these oddities are unavoidable, and we simply call them "roundoff error."

Roundoff error should not, however, lead to the peculiarity which I call "ratcheting." Suppose you have three numbers, X, Y and Z; we already know, because of unavoidable roundoff error, that it is unreasonable to expect for every X, Y and Z,

$$X+(Y+Z) = (X+Y)+Z$$

For instance suppose Z=-Y and X is very tiny by comparison; in (X+Y) some or all of the low-order bits of X are lost to rounding and when Z is added (subtracting off the effect of Y), the final sum is no longer the same as X. But if (Y+Z) is computed first its sum is zero, so the final sum is exactly the value of X. Suppose, however, that X is much larger in magnitude than Y or Z. The value of X dominates and you would expect to lose the same bits when adding Y as when subtracting it (by adding Z), so that

$$X = X+(Y-Y) = (X+Y)-Y$$

Yet as long as the true value of X+Y differs from its rounded-up value by exactly 1/2 in its least significant bit, alternately adding and subtracting Y, that is, repeating the computation of

$$\text{new } X := (\text{old } X + Y) - Y$$

will cause the value of X to creep up, unless the rounding is unbiased.

Retailers in California are aware of this problem in the computation of sales tax. If they simply always round up, they will accumulate slightly more sales tax than they are required to pay the State. The proposed standard specifies carefully how rounding shall be done. Unless some other option is demanded by the programmer, unbiased rounding is the rule.

Another feature of the proposed standard which helps users to get more nearly correct results is the so-called "gradual underflow." Most implementations of floating point, when underflow is detected, simply force the result to zero, and for most applications that is probably good enough. But in this standard, once the exponent underflows (i.e. a non-zero result

is computed which is smaller in magnitude than the smallest normalized number), the number simply loses some of its least significant bits and becomes "denormalized." These denormalized numbers can be used in subsequent computation with the result that in many cases they contribute no more than unavoidable roundoff error. Later in this paper I will show examples where this is a critical advantage.

It is important to realize that it is possible to achieve correct results, at least within roundoff error, with almost any implementation of floating point arithmetic, just as it is possible to code any algorithm in almost any computer language. But who wants to code in PDP-8 assembly language when Pascal or some other high level language can be used? This brings us to the next topic.

EASE OF USE

The floating point processor should accept our commands and data and, if at all possible, return correctly rounded results with no hassle. Only if there is something wrong with the results do we want to know about it. The five exception traps proposed by the standard are designed to cope with all of the kinds of errors we might be interested in, and the ability to disable the traps allows the user to ignore inconsequential exceptions. But even if the trap is disabled, a flag bit is there to remember that such an exception occurred, so that sometime downstream it may be noticed.

One of the distinctive traps in the proposed standard is **Inexact.** This is of particular interest to users of BASIC and certain business systems, where the floating point processor is used for all calculations including those for strictly integer data. The inexact flag or trap signals a roundoff error. If it occurs, the result is not the correct integer computation you expected. Considering the number of business programs written in BASIC, this may enable the ABC Widgit company to locate the reason the company books balance every month, but not for the whole year. It might help in trapping subscript calculation errors in BASIC or any other language that uses the floating point processor to do a fast multiply. This kind of feature will not bother the average user until it is needed.

Most floating point processors today will trap on overflow. Normally you want to know if your numbers got too big. But sometimes you may know that the overflowed number is destined to be divided into another number, and you would like to tell the machine, "set overflows to 'infinity' and continue," knowing that later when any finite number is divided by infinity it will produce zero. Large CDC machines do something like this instead of trapping. The proposed standard gives you a choice. Similarly, a divide by zero can either trap or return infinity. These provisions to deal with infinity are useful in the computation of continued fractions, which are known to give efficient approximations to certain integrals and other special functions.

For those who want to trap overflow or underflow, scale the values, and continue, the standard provides a well-defined result which is correctly rounded but whose exponent part has been adjusted (scaled) by a standard amount sufficient to move it away from the overflow/underflow threshold for subsequent computations.

Perhaps the subtlest convenience to the user comes from the gradual underflow feature, from which he benefits without having to know anything about it in advance. With underflow traps turned off (almost everyone does so), instead of results being forced sharply to zero at the underflow threshold, this standard allows the gradual loss of significant bits through a denormalizing process, thereby extending the range. But more importantly, the underflowed results are clearly marked. In most cases it is probably perfectly valid to force underflowed results to zero whenever they are negligible compared to the quantities to which they will be added later. But if this is ever an invalid assumption the denormalized numbers will either give better results, or they will probably stay around long enough to call attention to themselves and to the fact that the original assumption was false.

A good example of this is the computation of the square root of the sum of squares. The obvious algorithm for this is

```
SUM := 0;
FOR J := 1 TO N DO  SUM := SUM + X[J] * X[J];
RESULT := SQRT (SUM)
```

If this is executed with all traps turned off, and if the result is a normalized number, it is correct to within noise caused by rounding error. If the result is not normalized, then the insertion of a half dozen statements to test whether the overflow or underflow flags are set and scale the the computation yields a correct result whenever one is possible. By contrast, compare a program published by J. L. Blue of Bell Labs[1] which takes three pages of code to do exactly the same thing because he is not allowed to assume any standard responsg to overflow or underflow. The effect of underflow is more likely to be felt than is generally realized: in doing curve-fitting, a computation like this is repeated until a minimum result is obtained: the final iterations often come close to underflow.

As it turns out, this example is not unique. In matrix multiplication underflow can be completely ignored because the denormalized numbers contribute at most only roundoff error, unless a final result is not a normalized number. I understand that Horner's method of polynomial evaluation is made much simpler with the help of denormalized numbers, because of scaling to prevent overflow. And so on.

An optional feature in the standard is the **Extended Precision,** which adds enough range and precision to the computation to significantly diminish the accumulation of roundoff errors without adversely affecting the speed of computation. Extended Precision and range is achieved at low cost by exploiting the unpacked format usually used in implementing floating point hardware or software. Because of its awkward word size (not a power of two) you do not want to define arrays of variables in this form, but intermediate results can be computed in extended without even the awareness of the user of a high level language compiler, except that the results are correct to a roundoff error in the last bit instead of the last few bits. Consider again matrix multiplication: each elementwise product is rounded, then their sum is rounded, so that the accumulation of roundoff errors goes up with the dimensionality of the matrix (i.e. the maximum possible error goes up with the dimension; probable error goes up with the square root of the dimension). With only eight more bits of precision (as specified in this standard) you need dimensions greater than 10,000 before roundoff error is likely to go past the last bit; even worst-case errors cannot go past the last bit for dimensions less than 100. The same reasoning applies to the sum of squares problem. Furthermore, the extended range means that you can completely ignore the possibility of overflow or underflow, except at the end. Given this extended capability in the system and a good compiler that makes use of the feature, the user need not even be aware of its existence.

What does extended precision do for the business user? Imagine that you are trying to solve for the interest rate on a 30-year mortgage, compounded daily. You are dealing with an equation where X, which is 1+(interest rate in %)/100, is raised to the 10,000th power. Need I say more?

I suppose it may be argued that you get all of this and more by going up to the next level of precision, which is true, but that brings us to the next topic.

EFFICIENCY

The proposed standard specifies two (or three) storage representations for floating point numbers, each twice the length of the next lower. That is, Short (or Single) form at 32 bits, Long (or Double) at 64 bits, and possibly Quad at 128 bits. It also allows the system to support the unpacked forms, Short Extended at 48 bits and Long Extended at 80 bits[2]. In eight- and sixteen-bit CPUs the shortest format will necessarily require between 4 and 24 memory accesses just to pass the data to a (hardware) floating point processor and back for a single operation. Double precision would be twice that. Applications which require the precision afforded by Double may be unwilling to spend the time to compute intermediate results in Quad, but the time to compute intermediate results in Long Extended is only slightly more than that for Double. Furthermore, in hardware (or software) with 16-bit data paths the extra precision and range of the extended formats means nothing more than filling out the unused bits in the separated exponent and fraction registers. Add a few extra registers or an on-board computational stack (as most chip designers are likely to do) and the memory accesses are reduced by almost half; but 80-bit registers are much less costly of chip space than 128-bit registers, so you will probably get more of them or more functionality if Long Extended is implemented rather than Quad.

But this is not the only efficiency issue. Just knowing that the simplest algorithm produces correct results (because of correct rounding, denormalized numbers, etc.) will mean that the programs using a floating point processor that meets this standard will run faster. The root-sum-of-squares problem will produce correct results most of the time with no scaling and no tests inside the inner loop; all exceptional cases can be discovered by testing the quality of the result.

The real efficiency issue is, do you get the right answer with a reasonable effort?

COMPATIBILITY

I will briefly mention two areas of compatibility which will interest the users of the floating point standard. The first is historical or external and the second is internal or between conforming implementations.

Most of the initial users of this floating point standard (or of any other floating point specification that is implemented in a microcomputer environment) will be coming from an experience with existing floating point implementations, primarily on mainframe computers and minicomputers like the PDP-11. Those of you who have looked at the proposed standard will notice immediately that the Single precision has a very similar structure to that of the PDP-11, but the exponent part is biased two lower than the DEC counterpart. The result is that the precision is the same, but the range covers numbers twice as large as those on the PDP-11 down to numbers somewhat smaller than those of the PDP-11 (due to the gradual underflow in this standard). With only 24 bits of significance (including the so-called "hidden bit") it is three bits less precise than the 7094 and the Univac 1100 series, but marginally better than the hexadecimally normalized IBM 360/370. The range is exactly the same as the 7094 and 1100.

With eleven bits of exponent part, the double precision has the same range as the large CDC machines and slightly more precision. Its precision is comparable to double precision on the 360/370 but the range is considerably greater.

What this means is, almost any problems that ran correctly on the larger machines will also run correctly on this system, with results as good as or better than you are used to (within rounding error). Some problems that did not run correctly on the larger machines will now start to give different (probably better) answers. Some problems that seemed to run correctly on the other machines will die on this system as they encounter hitherto undetectable logical flaws. On the whole, users should find the differences to be an advantage.

When (and if) this proposal becomes an accepted standard, there will be a number of LSI circuits designed to implement some or all of the features. Whole instruments will be designed to communicate in "IEEE Standard Floating Point notation", just as RS-232C is a standard peripheral interface even for local devices which could otherwise communicate in byte parallel. No self-respecting software house would seriously consider bringing out a new package that used a format less well-designed than this, and the hooks to replace the software routines with the LSI hardware would no doubt be well-documented.

Eventually, (and I can assure you, they tremble in their boots!) the mainframers will be compelled to implement the standard on new machines, because the users will become aware of the better quality results. I suppose IBM will be the last holdout for hexadecimal normalization, and some of the high-speed number crunchers and signal processors may choose to leave out most of the features through the 1980s. Beyond that, well, we really cannot see that far very well.

The standard is written so that you can connect a signal processor to a microcomputer which massages the data then passes it to a mini, which does further computation before making a binary data tape which is fed as input to yet another computer, which passes the data to a DAC to close the loop. The same binary format is used, and the same operations in each system give the same results.

Of particular note here is that the microcomputer in the chain need not have a floating point processor if its only function is to sort and select ranges of data to pass on to the mini. The number system is perfectly ordered on a sign-magnitude scale, and the same routine can sort with equal ease either Single or Double precision numbers (but not mixed) or text data or positive integers. The denormalized numbers are, as you would expect, between the ordinary normalized numbers and zero; infinity is greater than the largest normalized number and non-numbers are beyond that.

COSTS

I have outlined some of the advantages of the proposed standard. Since there is no such thing as a free lunch, what do we give up for all these benefits?

For one thing, many of the features entail some trivial performance cost: estimates range from 1% to 20% total loss of speed (while actually doing floating point operations; this is reflected in perhaps something less than 3% loss of total throughput). Since next year's semiconductor process change increases speeds more than that, I claim the loss is negligible.

Some of the exception handling is complicated and will require extra code space in software or ROM space on microprogrammed LSI chips; wild estimates here run as high as 50%, which also corresponds to the chip size decrease in one year. I suspect clever programmers and chip designers can reduce that apparent overhead considerably.

One of the costs in the proposed standard is not readily apparent to the casual user. In order for the system to give exactly the response that makes your life easier, the treatment of exceptions must be complicated. Many people have difficulty understanding the whole system and how it fits together. One of the consequences of this fact is that implementation takes a great deal of forethought and understanding that is not necessary for a simpler, less convenient format. This means that the initial implementations will probably be priced a little higher. It also means that several chipmakers will rebel and come out with non-conforming parts, parts which either are honestly different or which are counterfeits (seeming to conform, but which perform differently). But the costs are almost entirely design costs, which can be quickly amortized. There is no reason to suppose that the floating point LSI chips will be any more expensive than the LSI microprocessors, which have to go through the same development cycles.

There is one hazard in the path between here and the adoption of the proposed standard: it may not find enough support in the standards committees to pass the required three-quarters mail ballot. Mainframe representatives in the committees will continue to drag their feet. Semiconductor representatives may boggle at the supposed complexity, as we have also already seen. There is rumor that academic politics may contribute to the resistance. If it fails to pass as a standard, it will nonetheless surely become a defacto standard, since at least one major semiconductor house is committed to its implementation and two or three second sources are inevitable. Once a $200 (or so) LSI part is available to do floating point arithmetic, minicomputer manufacturers will need to think twice before designing new systems that do arithmetic differently. When the word gets out that its results are better than most alternatives, the rest of the world will be compelled to go along.

CONCLUSIONS

I have tried to show that the proposed floating point standard will give results that engineers and businessmen can trust. I have tried to show that the way to get these results is often by the simplest and most obvious programs, as might be used by an unsophisticated hobbyist. And I hope I made it clear that the up front costs of getting good results are not significantly more than the costs for mediocrity.

You, the user, are the final judge. You will have a chance to use this standard: "Try it — you'll like it."

ACKNOWLEDGEMENT

A special note of appreciation is due to Professor W. Kahan of the University of California at Berkeley, who is the original advocate of the proposed standard, and without whose patient explanations I should not have been able to prepare this paper.

REFERENCES

[1] Blue, James L., "A Portable Fortran Program To Find the Euclidean Norm of a Vector" ACM Transactions on Mathematical Software, Vol.4, No.1, pp15-23.

[2] Coonen, Jerome, "Specifications for a Proposed Standard for Floating Point Arithmetic" presented at the Third West Coast Computer Faire in November, 1978.

SPECIFICATIONS FOR A PROPOSED STANDARD FOR FLOATING POINT ARITHMETIC

by Jerome T. Coonen October 1, 1978

INTRODUCTION

This draft standard is one of several before an IEEE subcommitee whose goal is to standardize floating point arithmetic for mini- and micro-computers. This is the seventh in a series of drafts, the first of which was presented by Harold Stone, William Kahan and Jerome Coonen in May, 1978. The present document benefits from extensive discussions in previous meetings of the IEEE subcommitee. The author wishes to acknowledge the help of Professor William Kahan and Tom Pittman in preparing this draft.

This proposal specifies meticulously its data formats and its very complete complement of arithmetic operations down to the last bit. It must do so to be of use to the producers of micro-processor hardware and software, who cannot afford to provide the support software and personnel to perform conversions between systems conforming to a less rigid standard. The present standard would allow communication between systems at the data level without conversions.

While this standard is precise in its specification of the results of arithmetic operations, it must be flexible enough to accommodate a variety of computer architectures (e.g. stack, multi-register, single-accumulator, storage-to-storage, dedicated auxiliary processor,...) and several possible levels of implementation (SINGLE precision only, both SINGLE and DOUBLE precisions,...). At the same time it must allow for future developments; some of them, hitherto precluded by ill-advised aspects of current designs, should instead be supported by the new system's design in so far as that support is achievable without an excessive burden upon the performance of today's tasks. Among the necessary developments are:

> Interval Arithmetic, which provides a certifiable result despite roundoff and over/underflow and other exceptions; and
>
> A degree of collaboration between numerical (approximate) procedures on the one hand and automatic symbolic algebraic (exact) prodedures on the other; and
>
> The use of reserved operands to extend the numerical data structure, say with complex infinities, or pointers into heaps of numbers with extended range and precision.

The new standard takes great care in the handling of exceptional conditions such as over/underflow. An attempt has been made to achieve a higher level of safety than has been customary, with enhanced utility but without excessive cost. This will impact existing software in the following way. Programs, which run now in higher level languages like FORTRAN should be portable to a system with the new standard arithmetic at the cost of a modest amount of editing and a recompilation, and then should execute with results almost certainly no worse than before, though programs which used to give incorrect results might now give diagnostic messages instead.

PART I: NARRATIVE DESCRIPTION OF THE STANDARD ARITHMETIC

BASIC LEVELS OF PRECISION

Any nonzero real number x may be expressed as $x = +2^e * f$, where e is a signed integer and $1 \leq f < 2$. We call this the "floating point" representation of x, with exponent e and significant digit field f. Our object is to describe a machine representation for real numbers based on this floating point decomposition, and to prescribe rules for arithmetic on these numbers. In our scheme, e and f will be represented by bit strings of prescribed lengths so that necesarily only a finite subset of the real numbers will be representable exactly.

This standard for floating point arithmetic admits two basic levels of precision, SINGLE and DOUBLE. It is possible that a third, QUAD, will be added. An implementation of this standard may provide SINGLE only, or both SINGLE and DOUBLE precisions. We require SINGLE in all standard systems, recognizing both its value as a debugging precision and its efficiency for a wide range of applications where storage economy matters.

We will discuss only SINGLE here, referring the reader to PART II for the analogous details of DOUBLE. A normalized nonzero number X in SINGLE precision has the form

$$X = (-1)^S * 2^{E-127} * (1.F) \qquad \text{where}$$

S = sign bit = bit string of length one
E = biased exponent = bit string of length 8 encoding integers in the range 0 to 255
F = significand = bit string of length 23 encoding those of X's significant bits that follow the binary point, yielding a 24 bit significant digit field for X that always begins "1.___".

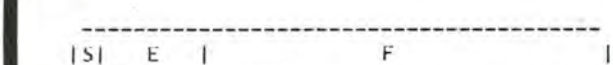
In terms of the above decomposition of X as a normalized number we have the following relations provided $0 < E < 255$:

sign of X = $(-1)^S$
e = E-127 = X's unbiased exponent
f = 1.F = X's signifcant digit field.

Note that the leading 1 bit of X's the significant digit field f is "implicit" in the storage representation. The values 0 and 255 of E are reserved to designate special operands to be discussed in later sections. It may at least be noted here that signed zero is represented by E = F = 0. With this constraint on E, a normalized nonzero SINGLE precision number X can range in magnitude between

$2^{-126} = 2^{-126} * 1.000...00$ and $2^{127} * 1.111...11 = 2^{128}$, inclusive.

The number X above is represented in storage by the bit string



This encoding has the special property that the order of floating point numbers coincides with the lexicographic order of their machine counterparts when interpreted as sign–magnitude binary integers.

The normalized nonzero SINGLE precision numbers are elements of a finite model for the real number system, a model we shall develop further in the following sections. In this model the normalized nonzero SINGLE precision numbers simply represent themselves as infinite precision real numbers.

ARITHMETIC OPERATIONS

The following arithmetic operations are completely described by this standard:

ADD REMainder INTeger_part
SUBtract CoMPare Floating_to_INTeger
MULtiply MOVe INTeger_to_Floating
DIVide SQuare_RooT
binary_INTeger_to_DECimal_string
deciaml_STRing_to_BINary_integer
BINary_floating_to_DECimal_floating
DECimal_floating_to_BINary_floating

The description is complete in the sense that, given operands at any levels of precision, and given the precision desired for the result, the result is specified by the standard. We refer the reader to PART II for tables detailing the operations.

An implementation of this standard must at least provide

1. ADD, SUB, MUL, DIV and REM for any two operands of the same precision, for each supported precision, with the result having no less exponent range than the two operands.

2. CMP and MOV for operands at any, perhaps different, supported levels of precision (in the case of MOV the second operand is the destination).

3. INT and SQRT for operands at all supported levels of precision, producing results at the precision of the specified destination.

4. Conversions between floating point integers and binary integers in the host processor.

5. Binary_decimal conversions to and from all supported levels of precision. These conversions are supported by conversions between floating point integers and decimal strings. A section of PART II is devoted to the details of these operations.

This standard provides a notably complete set of arithmetic operations in an attempt to facilitate program portability by guaranteeing that results obtained using standard arithmetic may be reproduced on different computer systems, down to the last bit. SQRT is included as a primitive operation because it appears so often, for example in matrix calculations, and because, when implemented in hardware according to published algorithms, its execution time is comparable with that of a divide. In many current systems, SQRT is implemented too slowly and/or too inaccurately, so that unnecessary effort is expended coding around it. The same applies in the case of REM. We point out that REM is preferable to the MOD function because REM is computed without rounding error. Consider, for example

0.01 MOD (–95) versus 0.01 REM (–95)

on a 2–digit machine. MOD yields the result round(–94.99) = –95 for a complete loss of accuracy while REM yields the correct result 0.01. The standard's insistence upon standard binary–decimal conversions is both a reflection of the delicacy of the calculations involved and an attempt to allow comparison of data from different systems at the decimal output level rather than via hexadecimal dumps.

ACCURACY AND ROUNDING

Every operation except MOVE and COMPARE is presumed to deliver its result to a destination no narrower than its input operand(s). Besides simplifying the narrative, this constraint avoids unnecessary complexity in the implementation. Systems which implemented the rare operation

DOUBLE * DOUBLE —> SINGLE

directly instead of indirectly via

DOUBLE * DOUBLE —> DOUBLE
MOVE (ROUND) DOUBLE —> SINGLE

would enjoy a slight advantage in speed and rounding error at the cost of a richer instruction set and considerably more complicated responses to over/underflow. There are better ways.

The simplest systems may restrict their instruction sets to allow no mixed–precision operations besides MOVE and COMPARE, providing only

SINGLE * SINGLE —> SINGLE and
DOUBLE * DOUBLE —> DOUBLE,

but they would not support efficient numerical computation adequately. There are many occasions when constructions like

SINGLE + SINGLE —> DOUBLE or

(SINGLE*SINGLE->DOUBLE)+DOUBLE->DOUBLE

ought to be used in innermost loops, and then the costs of "padding" like

(SINGLE —> DOUBLE) + DOUBLE —> DOUBLE or

(SINGLE —> DOUBLE) * (SINGLE —> DOUBLE) + DOUBLE —> DOUBLE

respectively become unnecessary nuisances. Rather than prohibit mixed-precision operations, the standard is designed to encourage the provision of some such operations, though it is unreasonable to expect operations that cater to every possible combination of input types and yield results at every supported precision level.

What is needed is control over the precision to which a result is rounded Independent of the ostensible precisions of the input operands. This is achieved in a standard system in either or both of two ways. One way is to specify the width of the destination subject to the constraint, mentioned above, that the destination be no narrower than the operation's input operands (except for the operations MOVE and COMPARE). The second way is to specify in advance the precision to which a result is to be rounded subject to the constraint that that precision be not too wide for the intended destination. For instance, the standard allows for operations of the form

((SINGLE*DOUBLE) rounded to SINGLE) -> DOUBLE.

Operations like that are appropriate for a system which, in an attempt to economize on the size of its instruction set, delivers all results except those of MOVE and COMPARE only to the widest destinations supported by the system. Such a system would encourage a healthy practice, namely the preservation of intermediate results for a long expression in the widest available precision, with just one serious rounding error at the end when the expression's value is stored in a narrower destination. But that healthy practice is practically precluded at present by ill-considered constraints built into current programming languages. Therefore, the standard's specifications for roundoff control are burdened by the complications necessary to provide, on the one hand, compatibility with past practice however ill-advised, and on the other hand an opportunity for better procedures in the future.

If the result of one of the arithmetic operations, when computed to infinite precision, is exactly representable within the exponent range and precision specified for the result, then it must be given exactly. Otherwise the result must be rounded according to one of the schemes to be presented in this section. In any case, the error will be less than one unit in the last place to which the result is rounded unless Over/Underflow or some other such exception intervenes.

To illustrate the rounding modes we let z be the infinitely precise result of an arithmetic operation. Then we determine Z1 and Z2, numbers representable exactly with the precision of the intended destination field, such that Z1 and Z2 most closely bracket z, that is $Z1 \leq z \leq Z2$, barely. Certain details concerning the exponent range and the alignment of the binary point of Z1 and Z2 are deferred to PART II. We then have:

Round_to_Nearest(z) = Unbiased_Round(z) = the nearer of Z1 and Z2 to z; in case of a tie choose the one of Z1 and Z2 whose least significant bit is 0.

Round_to_Zero(z) = Chop(z) = smaller of Z1 and Z2 in magnitude.

Round_to_Plus_Infinity(z) = Z2.

directed roundings

Round_to_Minus_Infinity(z) = Z1.

As noted, the latter two rounding modes are often referred to as the "directed roundings". They are intended to support Interval Arithmetic and to calculate certifiable upper and lower bounds, and to control conversion to integers. Round_to_Zero is useful too in controlling conversions to integers in accordance with conventions embedded in certain progamming languages like FORTRAN.

An implementation of the standard may support either of the following combinations of rounding modes:

1. Round_to_Nearest only, with Round_to_Zero for certain specified integer operations.
2. All four rounding modes.

If more than one rounding mode is supported then Round_to_Nearest shall be the default mode for all operations. Despite the apparent redundancy of rounding to Zero, to Plus_Infinity and to Minus_Infinity, both directed roundings are required if either is implemented. It is a false economy to attempt a saving in the number of rounding modes implemented, because codes are much simpler when given direct access to all rounding modes. In interval arithmetic, for example, one often computes the upper and lower bounds of an interval by executing the same sequence of instructions twice, rounding up during one pass and down the next.

Calculation of Round_to_Nearest requires the so-called Sticky Bit, as shown in PART II. Once the Sticky Bit is implemented, the directed roundings may be supplied at very little extra cost, the bulk of which lies in the mechanism, e.g. mode bits or extra opcodes, for exercising the choice of rounding mode. While the standard leaves this mechanism up to the implementer, we remark that the mode bits are usually preferable. For example, consider the interval arithmetic computation of the previous paragraph. This task is greatly expedited if changing rounding modes from one pass to the next is simply a matter of flipping a pair of mode bits.

EXTENDED PRECISION

To perform the arithmetic operations on numbers stored in SINGLE or DOUBLE pecision, a system will generally "unpack" the bit strings into their component fields

S = sign
E = biased exponent
F = significand.

Moreover, the leading bit of the significand will be made explicit, and perhaps the bias will be removed from the exponent. Since most current machines use data paths of widths 4, 8, 16, 32 or 64, the 24-bit SINGLE significand, with explicit leading bit, will likely be unpacked into a 32-bit field. Similarly, the 53-bit DOUBLE significand may be dealt with in a 64-bit field.

The standard provides a way to exploit this unpacked format, by admitting SINGLE_EXTENDED and DOUBLE_EXTENDED precisions; perhaps QUAD_EXTENDED will be added later. Support of the extended precisions is optional. If implemented at all, only one extended precision shall be provided, namely that corresponding to the widest basic precision supported.

As with the basic precisions, we will describe SINGLE_EXTENDED here, referring the reader to PART II for the full details of the extended precisions. SINGLE_EXTENDED is comprised of the fields

S = sign = bit string of length one
E = unbiased exponent = a signed integer of at least 11 bits
F = significand = a bit string of length at least 32 bits, with an explicit leading bit followed by an implicit binary point.

A number X encoded in SINGLE_EXTENDED is then given by

$$X = (-1)^S * 2^E * F$$

except possibly when E takes its most positive value. Signed zero is given by E = most-negative-value and F = 0.

The exponent width is so chosen to provide at least the range of DOUBLE precision. The most positive value of the exponent is reserved for the encoding of special operands to be discussed later. Having at least eight extra significand bits greatly simplifies the accurate computation of the trigonometric, logarithm and exponential functions, and the power function Y^X, to full SINGLE precision. Matrix calculations also benefit from SINGLE_EXTENDED accumulations of products of SINGLE precision data. Moreover, the extra bits of precision are so important in binary-decimal conversions that some extended capability must be simulated by system software if extended precision is not implemented; this is discussed in PART II.

If implemented, extended entities are assumed to be few in number, used to evaluate complicated subexpressions, for example. They are not intended to be indexed in arrays in higher-level languages.

Another way to obtain most of the computational benefits of extended precision is by using the "next higher" basic level of precision. Indeed, QUAD may be included in the standard solely as an alternative for those not wishing to implement DOUBLE_EXTENDED in a system with SINGLE and DOUBLE. One important difference between the basic and extended precisions is the leading significand bit, which is explicit only in the extended precisions. The section on treatment of Underflows will indicate how special classes of unnormalized numbers arise in the basic precisions. In extended precision, on the other hand, the explicit leading significand bit allows encoding of unnormalized numbers over the entire exponent range (except, of course, the reserved value). Thus EXTENDED is a more flexible way to get extended range than is the next higher basic level of precision; however it is less precise. Moreover, in most implementations, extended precision should be faster and cheaper than the next higher basic level of precision would be if implemented.

EXCEPTIONS

So far we have described the basic and optional extended precisions for encoding real numbers, and we have specified a large family of arithmetic operations on them. This is all quite straightforward, given the word sizes of current machines and the needs of people seeking floating point capability. A much more interesting question remains, with regard to the exceptional conditions that arise during arithmetic operations — how are the responses to exceptions to be standardized? The remainder of this narrative addresses this question.

The standard organizes the exception conditions under the five headings:

Invalid_Operation
Underflow
Overflow
Invalid_Division
Inexact_Result

The following sections treat these conditions individually, ultimately prescribing the results, if any, to be delivered in each instance. It is important to note that these results, as standard system responses, are independent of whether the standard is implemented entirely in hardware or software, or in a combination of the two. Certain diagnostic information passed via "results" is necessarily implementation-dependent; however, in the context of a given system, the results are uniquely determined by the standard.

For each of the five exceptions, an implementation of the standard may

a. Force a trap to user software.

or

b. Deliver a result specified by the standard and proceed.

or

c. Provide the user with a Trap_Enable bit whereby to choose (a) or (b), i.e. whether or not to trap.

Whenever a choice is given, the default shall be to proceed without a trap. The standard provides a precedence rule to determine which single trap is to be invoked in case several exceptions occur simultaneously.

Associated with each of the exceptions is a "sticky" flag which is set on each occurrence of the corresponding exception, regardless of the system response. Each flag may be tested by a program to determine whether an exception has occurred. Following an exception, a flag remains set until cleared by the user's program (or programming environment). In certain instances, e.g. when the end of a job is obviously at hand, a humane operating system may draw the user's attention to flags still set, thereby perhaps reminding him of exceptions that were overlooked by his program.

To deal effectively with traps, programmers need access to certain vital information, ideally:

What event caused the trap?
Where in the program?
What did the instruction try to do (what opcode)?
What were the operands (source and destination)?

In response, the programmmer will normally either:

Depart from the offending block of code to something utterly else.
Fix up the aberrant result and resume after the offending instruction.
Fix up the aberrant operands and re-execute offending instruction.

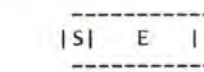
Sometimes the full range of information and responses is not needed, especially when the correct result is available, possibly in encoded form as in the case of Over/Underflow. One might dispense with some of the above information in these cases.

INVALID_OPERATION

The Invalid_Operation exception encompasses problems arising in a variety of arithmetic operations; it is the blanket covering those errors not frequent or important enough to merit their own fault condition. Here are examples of Invalid_Operations:

- $\sqrt{-5}$
- 0/0 with the Invalid_Division trap disabled
- plus_infinity + minus_infinity (the infinities will be introduced later)
- Attempted arithmetic with a designated reserved operand (these "Not-A-Numbers" will be introduced below).

We see that some invalid operations, like 0/0, cannot deliver numerical results that would be reasonable in all circumstances. For these situations we utilize one class of reserved operands, the Not-A-Number symbols, or NANs. In SINGLE and DOUBLE precisions, with the format



NANs are characterized by

S = sign bit (it may be irrelevant)
E = 111...11
F not= 0.

In extended precision, NANs have E = 0111...11, i.e. the most positive exponent. The sign bit S participates in the obvious way in the execution of statements like

X = -Y and Z = X-Y = X+(-Y)

without loss of information in the event that Y is a NAN with a numerical connotation.

The nonzero significand field F of a NAN will contain system-dependent information, for example:

a. A distinguished class of NANs, say with two leading zero significand bits, may be used by an operating system to initialize storage. The significand of such a NAN may be a name or a pointer to the region where the NAN is stored. As we will see below, these NANs will propagate through arithmetic operations, ultimately providing pointers to those areas of user-uninitialized storage which are the ancestors of meaningless final results.

b. A NAN generated by an invalid arithmetic operation on numeric data, for example 0 * infinity, may be a pointer to the offending line or block of code.

c. When complex arithmetic is implemented, it is often useful to think of infinity as a line rather than a point in the projective plane. A distinguished class of NANs, say those with two leading one bits in the significand, may be used in pairs to provide the relative sizes and signs of the real and imaginary parts of numbers tending to infinity along a fixed ray emanating from the origin. While these entities have the format of NANs, they contain signed numeric data and would be handled in software invoked by traps.

d. Sometimes an operation could generate an acceptable result but for its inability to pack that result correctly into the intended destination (see the discussion of over/underflows). In such a case, a NAN could be supplied as the "result", with a significand pointing to a place, e.g. an extended field or a heap in storage, where the correct result may be found.

e. Sometimes a subroutine may encounter data for which only a partial result can be delivered in the time available. The rest of the result can be replaced by NANs which point to a piece of the program which will resume execution of that subroutine only if that undelivered portion of the result is really needed.

f. List-oriented systems like LISP may use SINGLE precision NANs to point to DOUBLE numerical data.

As elements of our model of the real numbers, the NANs are extensions of the real number system. Their role in arithmetic operations is quite simple. While certain classes of NANs, for example those in (c) above, will cause an invalid_operation exception when picked up as operands, NANs will generally propagate through arithmetic operations without generating exceptions. For example 5+n —> n if n is a NAN. If two NANs are picked up as operands, the one with the smaller significand has precedence; this is more precisely specified in PART II.

We now specify system action on Invalid_Operation exceptions. If no trap is to be taken then the result of any Invalid_Operation is a NAN bearing some system-dependent information.

If an Invalid_Operaton occurs and a trap is to be taken, the result, if any, to be delivered is highly machine-dependent as well as operation-dependent. In some implementations, the trap will effectively occur before the operation is carried out, so no result need be written into the destination field. On the other hand, the trap may be invoked too late by some machines, i.e. after some result is produced and delivered. In this case the usual result is a NAN, though an implementation may, in certain situations, deliver a numeric result, for example it may make sense to deliver -5 in place of -5; these special cases are noted in PART II.

UNDERFLOW

Exponent Underflow is the most interesting of the exceptions because of the care taken by the standard to provide as much information as possible when proceeding without a trap. In the case of Overflow, on the other hand, a bare minimum of information is passed; this is discussed in the next section. For this reason, the range of normalized numbers in SINGLE and DOUBLE precisions has been chosen to diminish slightly the risk of Overflow compared with the risk of Underflow. This was done by picking the exponent bias and alignment of the binary point in the significand in such a way that the product of the largest and smallest positive normalized numbers is roughly 4 in both of the basic levels of precision.

We now discuss the treatment of Underflows. In each case we let z be the infinitely precise nonzero result of an arithmetic operation and we let z have the form

$$z = +2^e * f \qquad \text{where}$$

e is a signed integer and $1 \le f < 2$. We assume no prior knowledge about e. Note that before rounding z we check whether or not we'll trap on Underflow, if it occurs.

If a trap is to be taken on Underflow, then z is rounded to the precision of the destination field. If the exponent lies below the exponent range of the destination field then Underflow occurs. Because of the restrictions on arithmetic operations presumed under "ACCURACY AND ROUNDING", the exponent can be out of range by at most a factor of two, except to the MOV instruction which is discussed in PART II. The exponent is wrapped around into the desired range with a bias adjust specified in PART II. The result is then delivered to the destination and the trap is invoked.

If no trap is to be taken on Underflow, then the exponent of our infinitely precise result z is tested before rounding. If it lies below the minimum possible exponent of the destination field, then z is "denormalized", that is:

> the significant digit field of z is right-shifted
> while z's exponent is incremented until it reaches
> the minimum possible exponent of the destination.

Then z is rounded to the precision of the destination field, and the result is delivered, in a manner to be described presently. Unless z rounds up in magnitude to the smallest nonzero normalized number, Underflow is signalled.

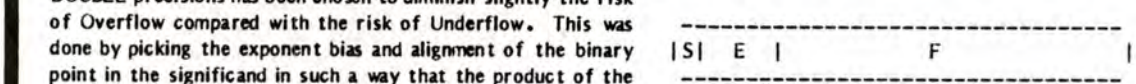
To illustrate the denormalization process let us consider an example: Let $Z = +2^{-130} * 1.01101...$ and suppose that the destination is a SINGLE precision field. As a further simplification let us assume there are only 6 bits of precision to be carried, plus the implicit leading bit, in SINGLE. Then

$Z = 2^{-130} * 1.01101...$ $\quad -130 < -126$ so we denormalize

$Z = 2^{-126} * 0.000101101...$ we round (to nearest, say)

$X = 2^{-126} * 0.000110$ = the result to be delivered.

We call the above result X a "denormalized number" in SINGLE precision; it is a special type of unnormalized number, namely one with the smallest possible exponent for the given basic level of precision. Note that the exponent of a denormalized number links it to a basic level of precision. We will discuss only SINGLE precision denormalized numbers here. DOUBLE precision is essentially the same; see PART II for details of the differences. In terms of the format



a SINGLE precision nonzero denormalized number X is encoded as

> S = sign bit
> E = 0
> F = nonzero string of bits to the right of the binary point of X.

We reconstruct X via the formula

$$X = (-1)^S * 2^{-126} * (0.F)$$

observing that E is not the true biased exponent in SINGLE precision. Comparing this formula with its analogue for normalized numbers we see that, when unpacking a denormalized number, the 1-bit that would have gone to the leading bit of the significand for a normalized number is instead added into the unbiased exponent $E-127+1$.

The denormalized numbers and signed zeros are the family of reserved operands corresponding to a biased exponent of zero. The values ± 0 are obtained just when $F = 0$ above. Zero may result from an Underflow, depending on the rounding mode, when the Underflow is so severe that all nonzero bits are shifted out of the significand field.

The denormalized numbers and ± 0 join the normalized numbers and NANs as elements of our model of the real numbers. Both $+0$ and -0 correspond to the real number 0 and are identical in every operation except division; this will be discussed along with Invalid_Divisions. A denormalized number X represents, roughly speaking, all of the real numbers which would round to that bit-string X in the specified rounding mode and precision. We note that the denormalized numbers are designed not so much to extend the exponent at any level of

precision, but rather to allow further computation with some sacrifice of precision in order to defer as long as the possible the need to decide whether the underflow will have significant consequences.

In add/subtracts, denormalized numbers behave in much the same way as normalized numbers, with never more than a rounding error committed in any operation. The situation is different in multiply/divides, where multiplying a SINGLE precision denormalized number by a large power of 2 and attempting to store the result in SINGLE is an Invalid_Operation. The unnormalized significand, having suffered loss of precision during some prior Underflow, may not be promoted to normalized status merely by multiplication. If, however, the destination had been an extended field, the unnormalized significand with large exponent would have been a (perhaps temporarily) legitimate result. PART II gives the full details of denormalized numbers in arithmetic oprations.

The implementation of denormalized numbers, whether in hardware or software, is required only of those systems in which, on Underflow, users may proceed without trapping. Implementations not supporting denormalized numbers, and thus forcing a user trap on every Underflow, must nonetheless sense the denormalized numbers as bit strings, when they are picked up as operands, and generate an Invalid_Operation fault. Note that implementations whose default, or only, option upon Underflow is to underflow abruptly (i.e. from anything smaller than the smallest normalized number) to zero, do not conform to the standard.

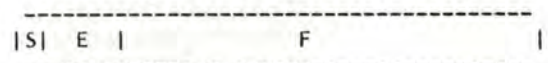
OVERFLOW

In contrast to the graceful treatment of Underflows in no-trap situations, Overflows are dealt with swiftly and surely, with a corresponding loss of information. It is noted in the discussion of Underflows that to take the greatest advantage of the treatment of Underflows, the number system is slightly biased away from zero, in the hope of making Overflows more rare than Underflows.

We now discuss the treatment of Overflows. In each case we let X be the normalized result of an arithmetic operation; we assume that X has been rounded to the precision of the destination field and that X has overflowed the exponent range.

If a trap is to be taken then, because every arithmetic operations's result goes to a destination no narrower than its input operands, the exponent can be out of range by at most a factor of two, except for the MOVE operation which is discussed in PART II. The exponent is wrapped around into the desired range with a bias adjust specified in PART II. This result is delivered to the destination field and the trap is invoked.

If no trap is to be taken, then infinity with the sign of X is written into the destination field. In SINGLE and DOUBLE precision with format



infinity is encoded as

S = sign bit
E = 111...11
F = 0.

In extended precision E = 0111...11 = the most positive exponent and F = 0.

The signed infinities and NANs thus comprise the family of reserved operands with most positive exponent. As elements of our model of the real numbers, the infinities are given two interpretations. In the Affine Closure,

minus_infinity < real_numbers < plus_infinity.

But in the Projective Closure the sign of infinity is ignored, i.e.

infinity = minus_infinity = plus_infinity,

and all comparisons between infinity and a real number involving order relations other than = or not= are invalid_operations.

Aside from the compares, all operations on the infinities in the two closures are the same except that both

infinity + infinity and infinity - infinity

are Invalid_Operations in the Projective Closure. Systems supporting the infinities shall provide an Affine/Projective mode bit so that choice of closures can be made under program control. The Affine mode is the default mode, and is appropriate for most engineering calculations involving exponentials or disparate time constants or infinities generated by overflows. The Projective mode is appropriate for real and complex rational arithmetic, for continued fractions, and for infinities generated by division by zeros which are not generated by underflows.

The infinities interact with ±0 in a very special way. It was noted in the last section that, aside from a trivial exception noted in PART II, +0 and -0 participate identically in all operations except division. The only way to distinguish +0 and -0 arithmetically is to use the fact that

+1/+0 = plus_infinity > +1/-0 = minus_infinity

can be recognized in the Affine mode. In terms of our model of the real numbers, this situation is to be expected. Since we associate the two elements ±0 with the single real number 0, we should not be able to distinguish machine ±0 using arithmetic on real numbers; rather, we find that we can disitinguish them only in a proper extension of the real numbers that includes infinities.

As we saw in the case of Underflows, systems forcing traps on Overflow need not support infinities but must recognize them when they are picked up as operands for arithmetic operations, and generate an Invalid_Operation exception.

INVALID_DIVISION

The Invalid_Division exception arises when a zero divisor occurs in a division operation. It also arises when a denormalized divisor is picked up in a system not implementing division by denormalized numbers; see PART II for details.

If the divisor is zero and the dividend is nonzero, the result is infinity with sign according to convention.

If both the divisor and dividend are zero, or if the divisor is (too far) denormalized, then if the Invalid_Division trap is enabled, it is invoked; if a result must be delivered it is a NAN. If the Invalid_Division trap is disabled then an Invalid_Operation exception arises; if a result must be delivered it is a NAN.

INEXACT_RESULT

The Inexact_Result exception arises when a round-off error is committed in an arithmetic operation. It is intended for essentially integer calculation as in COBOL and to facilitate multiple precision calculation. The rounded result is delivered to the destination field and the trap is invoked if enabled. When this exception occurs together with Over/Underflow, the latter traps have precedence.

PART II: SPECIFICATIONS FOR STANDARD ARITHMETIC

BASIC LEVELS OF PRECISION

SINGLE and DOUBLE are the basic levels of precision. A standard system shall provide either SINGLE only, or both SINGLE and DOUBLE. In addition a system may provide the extended precision corresponding to the widest basic precision supported.

The tables that follow detail the levels of precision and the data types specified by this standard. Of the reserved operands, the denormalized numbers and infinities need not be implemented in hardware in systems trapping to user software on all Overflow, Underflow and Invalid_Division exceptions, provided these operands cause an Invalid_Operation exception when picked up as operands in an arithmetic operation.

The signed infinities, when implemented, will be interpreted in either the Affine or Projective closures of the real numbers. In the latter case the sign of infinity is ignored by the add, subtract and compare instructions, i.e. "plus" and "minus" infinity are treated as the same, unsigned, infinity. Choice of closures shall be exercised via the Affine/Projective mode bit, which may be sensed and changed by user programs. Affine mode shall be the default for all arithmetic operations.

ARITHMETIC OPERATIONS

The following arithmetic operations are completely described below:

ADD	REMainder	INTeger_part
SUBtract	CoMPare	Floating_to_INTeger
MULtiply	MOVe	INTeger_to_Floating
DIVide	SQuare_RooT	

BINary_integer_to_decimal_STRing
decimal_STRing_to_binary_INTeger
BINary_floating_to_DECimal_floating
DECimal_floating_to_BINary_floating

An implementation of this standard must at least provide:

1. ADD, SUB, MUL, DIV and REM for any two operands of the same precision, for each supported precision, with the result having no less exponent range than the operands.

2. CMP and MOV for operands at any, perhaps different, supported levels of precision (in the case of MOV the second operand is the destination).

3. INT and SQRT for operands at all supported levels of precision, with the result having the precision of the specified destination.

4. Conversions between floating point integers in all supported levels of precision and binary integers in the host processor.

5. Radix conversions, as described in a separate subsection below.

All operations except MOV must have a destination field with exponent range at least as wide as the range of their wider operand.

For simplicity, the arithmetic operations are broken into two steps. In the first step a preliminary result Z is formed and, if numeric, rounded to the precision of the destination. This step is peculiar the the specific operation. In the second step the result Z is delivered to the destination, any exceptions are noted, and any traps invoked. The second step is the same for all operations except REM and MOV; the minor differences are noted.

One or more of five exceptional conditions may arise during an arithmetic operation: Overflow, Underflow, Invalid_Division, Invalid_Operation and Inexact_Result. For each of the exceptions, an implementation of the standard may

a. Force a trap to user software.

or

b. Deliver a specified result and proceed.

or

c. Provide the user with a Trap_Enable bit whereby to choose (a) or (b).

Whenever a choice is given, the default shall be to proceed without a trap. Associated with each of the exceptions is a sticky flag which is set on the occurrence of the corresponding exception, regardless of the system response. The flags may be

PRELIMINARY -- SUBJECT TO REVISION

BASIC LEVELS OF PRECISION

	SINGLE	DOUBLE
Length in bits	32	64
Fields:		
S = sign	1	1
E = exponent	8	11
F = significand	(1)+23	(1)+52
Storage format:	|S| E | F |	|S| E | F |
Interpretation of sign:		
Positive	0	0
Negative	1	1
Normalized numbers:		
Interp. of E	unsigned integer	unsigned integer
Bias of E	127	1023
Range of E	1 <= E <= 254	1 <= E <= 2046
Interp. of F	significant digit field = 1.F	significant digit field = 1.F
Relation to represented real number	$(-1)^S*2^{E-127}*(1.F)$	$(-1)^S*2^{E-1023}*(1.F)$
Signed Zeros:		
E =		0
F =	0	0
Reserved Operands:		
Denormalized numbers:		
E =	0	0
Bias of E	126	1022
Interp. of F	significant digit field = 0.F	significant digit field = 0.F
Range of F	nonzero	nonzero
Relation to represented number	$(-1)^S*2^{-126}*(0.F)$	$(-1)^S*2^{-1022}*(0.F)$
Signed Infinities:		
E =	255	2047
F =	0	0
Not-A-Number, or NAN:		
E =	255	2047
Range of F	nonzero	nonzero
Interp. of F	system-dependent diagnostic and possibly numeric information	
Ranges:		
Max positive normalized	$2^{126}*(2 - 2^{-23}) = 1.7*10^{38}$	$2^{1022}*(2 - 2^{-52}) = 9*10^{307}$
Min positive normalized	$2^{-126} = 1.2*10^{-38}$	$2^{-1022} = 2.2*10^{-308}$
Min positive denormalized	$2^{-149} = 1.4*10^{-45}$	$2^{-1074} = 4.9*10^{-324}$

EXTENDED PRECISION

		SINGLE_EXTENDED	DOUBLE_EXTENDED
Length in bits	>=	44	80
Fields:			
S = sign		1	1
E = exponent	>=	11	15
F = significand	>=	32	64
Storage format:		not specified beyond minimum field widths	
Interpretation of sign:			
Positive		0	0
Negative		1	1
Interp. of exponent:		unsigned integer	unsigned integer
Max E	>=	1023	16383
Min E	<=	-1024	-16384
Nonzero numbers:			
Range of E		Min E to (Max E - 1)	Min E to (Max E - 1)
Interp. of F		significant digit field with binary point to the right of the leading bit	
Relation to represented number		$(-1)^S*2^{E*F}$	$(-1)^S*2^{E*F}$
Signed zeros:		use special indicator or condition bits, or else	
E =		Min E	Min E
F =		0	0
Reserved operands:			
Signed infinities:		use special indicator or condition bits, or else	
E =		Max E	Max E
F =		0	0
Not-A-Number symbols, or NANs:		use special indicator or condition bits, or else	
E =		Max E	Max E
Range of F		nonzero	nonzero
Interp. of F		system-dependent diagnostic and possibly numeric information	
Ranges:			
Max positive normalized	>=	$2^{1022}*(2 - 2^{-31}) = 9*10^{307}$	$2^{16382}*(2 - 2^{-63}) = 6*10^{4931}$
Min positive normalized	<=	$2^{-1024} = 5.6*10^{-309}$	$2^{-16384} = 8*10^{-4933}$
Min positive unnormalized	<=	$2^{-1055} = 3*10^{-318}$	$2^{-16447} = 9*10^{-4952}$

Note that table entries giving specific values for the exponent E of the zeros and reserved operands depend on the number of bits in the exponent field.

sensed and changed by user programs. Following an exception, a flag remains set until cleared by user software.

A system providing a trap on an exceptional condition must give sufficient informatkon to allow correction of the fault. The correct result may be given encoded, as in Over/Underflow with exponent wraparound, or in a heap pointed to by a NAN written into the destination. On the other hand, if no numeric result can be given, the opcode and aberrant operands must be provided; in this case if the destination field is the same as one of the source fields then the trap must be taken before any "result" is written over the source operand.

While the specifications of the arithmetic operations indicate that NANs propagate through operations without raising exceptions, a system may raise the Invalid_Operation exception for a system-specified distinguished class of NANs. If the Invalid_Operation trap is enabled it should be invoked at the start of the operation, i.e. before any results are produced; if the trap is disabled a NAN should be generated as in any Invalid_Operation.

In the event that two NANs occur as operands in an arithmetic operation, and neither is designated to cause an Invalid_Operation exception the following precedence rule determines which will be propagated as the result of the operation:

> The sign and exponent are ignored, and the signficands are compared as numbers of the form 0.bbbb..., i.e. the leading bit, whether explicit or implicit, is taken to be 0. The NAN which is smaller by this comparison is the result of the operation.

"M": In the tables specifying the arithmetic operations, the entry "M" indicates that the above precedence rule is to be applied to two NANs picked up as operands.

ADD/SUBTRACT

Form a preliminary result Z = X + Y. On a SUBTRACT set Y = -Y and ADD. Z is given by the following table:

X + Y	±0	V	±infinity	NAN
±0	a	Y	Y	Y
V	X	b	Y	Y
±inf	X	X	c	Y
NAN	X	X	X	M

where V is any nonzero number, possibly unnormalized.

a.

	+0	-0
-0	-0	A
+0	A	+0

A = +0 in rounding modes RN, RZ, RP and -0 in rounding mode RM.

b. 1) Align the binary points of X and Y by unnormalizing the operand with the smaller exponent until the exponents are equal. Note whether either of the resulting significands is normalized (for step 4 below).

2) Add the operands, yielding a result which may be viewed as:

sgn	exp	V	N.	G	R	S

<--rounding precision-->

where the binary point follows N, and S = sticky bit = logical OR of all bits to the right of R.

3) Addition of magnitudes: If V=1 then right shift one bit and increment exponent. During the shift R is ORed into S.

4) Subtraction of magnitudes: If, after binary point alignment in (1), either of the operands was normalized, then

i) If all bits of the (unrounded) significand are zero, set the sign to "+" in rounding modes RN, RZ, RP, and set the sign to "-" in mode RM, as in case A of (a) above. Moreover, since both operands were normalized after step 1 above above, then the result is true zero, i.e. the exponent is set to its most negative value.

ii) Otherwise (some significand bit is nonzero)... Normalize the result, i.e. left shift the significand while decrementing the exponent until N=1. S need not participate in the left shifts; either zero or S may be shifted into R from the right.

5) Round, as specified under "ROUNDING".

c. In Affine mode (+infinity) + (+infinity) -> (+infinity) and (-infinity) + (-infinity) -> (-infinity). In Affine mode on (+infinity) + (-infinity) and (-infinity) + (+infinity), and in all cases in the Projective mode, signal Invalid_Operation, and if a result must be delivered set Z to NAN.

MULTIPLY

Form preliminary result Z = X * Y. Z is given by the following table, with sign = exclusive OR of the input signs:

X * Y	±0	V	±inf	NAN
±0	g	g	i	Y
V	g	h	j	Y
±inf	i	j	j	Y
NAN	X	X	X	M

where V is any nonzero number, possibly unnormalized. (Perhaps the standard should specify that NAN * 0 = 0).

g. Z=0 with sign.

h. 1) Generate sign and exponent according to convention. Multiply the significands. The result may be viewed as:

```
 ---   ---   ----------------------------
|sgn| |exp| |V|N.                  |G|R|S|
 ---   ---   ----------------------------
             <--rounding precision-->
```

where the binary point follows N, and S = sticky bit = logical OR of all bits to the right of R.

2) If V=1 then right-shift the significand one bit and increment the exponent, and go to (4). Else, when V=0...

3) If N=0, then left shift the significand one bit and decrement the exponent. S need not participate in the left shift; a zero or S may be shifted into R from the right. (This step is contentious, and may not be included in the standard.)

4) Round, as specified under "ROUNDING".

i. Signal Invalid_Operation. If a result must be delivered, set Z to NAN.

j. Z = infinity with sign according to multiply convention.

DIVIDE

Form a preliminary result Z = X/Y. Z is given by the following table, with sign = exclusive OR of the input signs:

	X / Y	±0	unnorm	norm	±inf*	NAN
X	±0	m	m*	x	x	Y
	V	k	m*	n	g	Y
	±inf	k	j	j	m	Y
	NAN	X	X	X	X	M

(Y)

where V is any nonzero number, possibly unnormalized. (Perhaps the standard should specify that NAN / infinity = 0).

g. Z=0 with sign.

j. Z = infinity with sign.

k. Z = infinity with sign. Signal Invalid_Division.

m. Signal Invalid_Division. If a result must be delivered then set Z to NAN.

n. 1) Generate sign and exponent according to convention. Divide the significands. The result may be viewed as:

```
 ---   ---   ----------------------------
|sgn| |exp| |N.                    |G|R|S|
 ---   ---   ----------------------------
             <---rounding precision--->
```

where the binary point follows N, and S = sticky bit = logical OR of all bits to the right of R.

2) If N=0 then left shift significand one bit and decrement exponent. S need not participate in the left shift; either a zero or S may be shifted into R from the right. (The standard may allow a second left shift if N=0 after the first.)

3) Round as specified in "ROUNDING".

*These divisions may be implemented provided the result has no or one more significant bit than the operand with the fewer significant bits. The standard may allow divisors whose significant digits have the form 0.1bbb... where the b's are either 0 or 1.

REMAINDER

Form the preliminary result Z = remainder when X is divided by Y, with integer quotient Q. Q does not participate in STEP TWO of the operation unless an exception is raised there, in which case if Z is set to NAN then Q is assigned the same value. Z and Q are given by the following table, with the sign of Q given by the exclusive OR of the signs of the input operands.

	X / Y	±0	unnorm	norm	±inf	NAN
X	±0	w1	x	x	x	Y
	V	w2	w	y	x	Y
	±inf	w2	w	w	w	Y
	NAN	X	X	X	X	M

(Y)

where V is any nonzero number, possibly unnormalized.

w. Signal Invalid_Operation. If results must be delivered then set Z and Q to NAN.

w1. Signal Invalid_Operation. If results must be delivered then set Z to X and Q to NAN.

w2. Signal Invalid_Operation. If results must be delivered then set Z to 0 with the sign of X and set Q to infinity with sign according to divide convention.

x. Q=0 with sign. Z=X. (This is equivalent to (y) when the divisor is 0.)

y. Set Q to the integer part of X/Y, with sign. If Q contains more significant bits than its intended destination, then discard the excessive high order bits and signal Inexact_Result. Set Z to the remainder, with sign of X. Normalize and round Z as in "ROUNDING".

SQUARE_ROOT

Form a preliminary result Z = X. Z is given by the following table:

X	Z
±0	X
-unnormalized	s1
+unnormalized	s
-normalized	t1
+normalized	t
-infinity	u
+infinity	X
NAN	X

s1. Signal Invalid_Operation. If a result must be delivered, set Z to NAN. The standard may allow - -X as in s.

s. Compute Z = X to at least the number of bits required to produce a correctly rounded result*. Then unnormalize Z until it has just one more significant bit than X has. Round, as specified in "ROUNDING". (The standard may classify this as an Invalid_Operation with NAN as the result.)

t1. Signal Invalid_Operation. If a result must be delivered, set Z to NAN. The standard may allow - -X as in t.

t. Compute Z = X to the number of bits required to produce a correctly rounded result*. Round as in "ROUNDING".

u. In Projective mode Z=X. In Affine mode signal Invalid_Operation; and if a result must be delivered then set Z to -infinity, if possible, otherwise set Z to NAN.

*To round correctly in all cases, calculate two more bits of X than the precision of the destination, which precision is never less than that of X.

INTEGER_PART

Form a preliminary result Z = integer_part(X). Z is given by the following table:

X	Z
±0	X
V	p
±inf	q
NAN	X

where V is any nonzero number, possibly unnormalized.

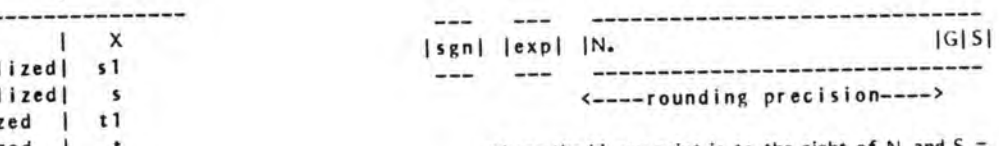
q. Signal Invalid_Operation. If a result must be delivered then set Z to X.

p. 1) If X has no (zero or nonzero) fraction bits in its significand then set Z to X. Ouherwise...

2) Right-shift X's significand while incrementing the exponent until no bits (zero or nonzero) of the fractional part of X lie within the rounding precision in effect. The exponent's value will then be:

23	single
31	single_extended
52	double
63	double_extended

The result may be viewed as:



2) Round as specified in "ROUNDING".

3) If all of the significand bits of Z are 0 then set Z to zero with the sign of Z. Otherwise normalize Z. S (which was set to zero after rounding in step 2) need not participate in the left-shifts of normalization; zero or S is shifted into G from the right.

MOV

If the source field is wider in range and precision than the destination field then the MOV is an arithmetic operation. It is the only operation whose destination may have shorter range and precision than its source operand. A preliminary result Z is given by the following table:

X	Z
±0	X
V	r
±inf	X
NAN	X

where V is any nonzero number, possibly unnormalized.

r. Z=X rounded, as specified in "ROUNDING".

STEP TWO of the MOV operation differs from that of the other arithmetic operations in the following way. On Over/Underflow with the corresponding trap enabled, the exponent may be more than a factor of two beyond the exponent range of the destination. In this case the BIAS ADJUST routine is not invoked, rather a NAN is written to the destination field indicating that the correct result is the (unchanged) source operand of the operation.

If the destination field is wider in range and precision than the source field than the MOV is exact with one exception:

In MOV SINGLE —> DOUBLE, if the SINGLE operand is denormalized then an Invalid_Operation exception arises; deliver a NAN to the destination indicating that the (unchanged) source operand is the correct result.

ROUNDING

Round the preliminary result W of an arithmetic operation to get the rounded result Z. Four rounding modes are described by the standard:

RN — Round to Nearest
RZ — Round to Zero
RM — Round to –infinity
RP — Round to +infinity.

An implementation of the standard may support eitherof two combinations of rounding modes:

1. RN only, with RZ for certain specified integer operations.
2. All four rounding modes.

If all four rounding modes are supported then RN shall be the default mode for all arithmetic operations.

Many systems will support more than one level of precision; some as many as three (SINGLE, DOUBLE, DOUBLE_EXTENDED). When a system supports more than one level of precision it must provide users with the option of rounding to a shorter precision results intended for a wider destination. The specification of that option will require at most two bits of information:

One bit to specify whether to round to EXTENDED or to a basic level of precision;

One bit to specify either round to SINGLE or round to DOUBLE, effective only when rounding to a basic level.

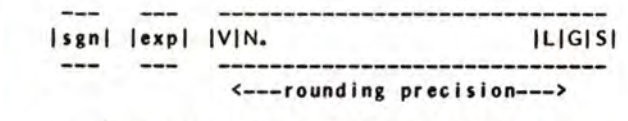
If the rounding precision specified is wider than can be held in the intended destination, the latter width will prevail. The standard does not specify how this rounding option will be specified, whether by

preset rounding mode bits , or

rounding mode options in each instruction , or

rounding instructions which can follow the operation whose result is re–rounded , etc.

The number W to be rounded may be viewed as:



where the binary point follows N, and S = sticky bit = logical OR of all bits to the right of G. V=0 at the start of rounding. If the exponent underflows the intended destination and the Underflow trap is disabled, then denormalize W, i.e. shift the significand right while incrementing the exponent until the exponent reaches its most negative allowable value. During each right–shift the G bit is OR–ed into the S bit; the S bit is not shifted.

Determine W1 and W2, numbers representable in the desired rounding precision, as follows:

If G=S=0 then W1 = W2 = W and
Z = RN(W) = RZ(W) = RP(W) = RM(W) = W.

Otherwise:
Signal Inexact_Result.
Set t = W with G and S = 0.
Compute T:
Add 1 to the L bit of t's significand.
If V=1 right–shift the significand one bit and increment the exponent.
If sgn=0 then W1 = t and W2 = T; otherwise W1 = T and W2 = t.
Then the rounded values are determined by:
Z = RN(W) = the nearer of W1 and W2 to W; in case of a tie choose the one of W1 and W2 whose L bit is 0.
RZ(W) = the smaller of W1 and W2 in magnitude.
RM(W) = W1.
RP(W) = W2.

STEP TWO OF ARITHMETIC OPERATIONS

Rounded preliminary result Z was developed in the first step.

1. Special cases involving numeric values of Z:
   a. Test whether Z's exponent over/underflows the intended destination.
   b. If Z is unnormalized...
      i. If the rounding mode is RP or RM then normalize Z as far as possible without allowing Z's exponent to fall below the underflow threshold. Otherwise...
      ii. (In rounding modes RN and RZ...) If the destination is not EXTENDED and Z has not been denormalized by Underflow with the trap disabled, then signal Invalid_Operation and Inexact_Result and if a result must be delivered set Z to NAN.
2. Over/Underflow cases:
   a. On Over/Underflow with the corresponding trap enabled, adjust the exponent bias as specified below.
   b. On Overflow with the trap disabled signal Inexact_Result. Then set Z to infinity with the sign of Z if the rounding mode is:

      RN
      RZ
      RP and Z is positive
      RM and Z is negative.

      Otherwise set Z to the largest normalized number representable in the destination field, with the sign of Z.
3. Set the exception flags:
   a. Do not signal Over/Underflow if the rounding mode is RP or RM and corresponding trap is disabled.

b. If Invalid_Division has been signalled but the corresponding trap is disabled, then signal Invalid_Operation.

c. Set the sticky exception flags corresponding to the exceptions signalled.

4. Deliver Z to its destination. (This may not be required when certain exceptions occur.)

5. Trap if any exception has been signalled for which the corresponding trap is enabled. In the event that more than one signalled exception have their traps enabled, only one trap shall be invoked, according to the following precedence:
   Overflow
   Underflow
   Invalid_Division
   Invalid_Operation
   Inexact_Result.

BIAS ADJUST

On over/underflow, with the corresponding trap enabled, the exponent of a rounded result Z is wrapped around into the required range of the destination. Compute B =
   190 in SINGLE
   1534 in DOUBLE
   $3*2^{n-2}-2$ in (SINGLE or DOUBLE) EXTENDED, where n is the number of bits in the signed exponent.
On Overflow subtract B from Z's exponent; on Underflow add B to Z's exponent.

FLOATING_TO_INTEGER

This instruction converts a floating point integer X into a binary integer of the host processor.

If X is a NAN, infinity, or non-integer then leave the destination unchanged and set the Invalid_Operation bit, trapping if the corresponding trap is enabled. Otherwise...

If X is +0 then 0 is written into the destination, with the sign of X if the processor supports signed zeros. Otherwise...

Convert nonzero integer X to a binary integer and write the result into the destination. If X overflows the destination field then truncate excessive high order bits and signal integer_overflow in the host processor, if it recognizes such an exception.

INTEGER_TO_FLOATING

Map the binary integer x in the host processor into a floating point integer. If x cannot be represented exactly then round as described in "ROUNDING" and set the Inexact_Result bit, trapping if the corresponding trap is enabled.

COMPARES

One of four conditions can result from a floating point compare:

<, =, >, different_and_unordered.

From these conditions there follow:

```
= implies not<,     <=,     >=, and not>  ;
< implies not=,     <=, not>=, and not>  ;
> implies not<, not<=,     >=, and not=  .
```

The following table specifies the compare operation:

X \ Y	-inf	AFF	n	-0	+0	p	+inf AFF	inf PROJ	NAN
-inf	=	<	<	<	<	<	////		b
AFF							////		
n	>	c	<	<	<	<	a		b
-0	>	>	=	=	<	<	a		b
+0	>	>	=	=	<	<	a		b
p	>	>	>	>	c	<	a		b
+inf	>	>	>	>	>	=	////		b
AFF							////		
inf	////	a	a	a	a	////	=		b
PROJ	////					////			
NAN	b	b	b	b	b	b	b	b	b

where n is a negative number and p is a positive number.

a. different_and_unordered.

b. Set the Invalid_Operation exception bit. If X and Y are equal as bit strings then the result is "equal", otherwise the result is "different_and_unordered". Trap if the corresponding trap is enabled.

c. Determine <, =, or > by comparison of X and Y as sign-magnitude bit strings if both have the same level of precision, and by floating point subtraction if either X or Y is EXTENDED or if X and Y are at different levels of precision. The subtraction may not have to be carried out completely. The standard does not yet specify the result of the comparison when the difference is a nonzero number with zero significand, as can be obtained only if either X or Y is EXTENDED and one is unnormalized.

RADIX CONVERSION

A. A system need provide standard conversion to and from only its basic levels of precision. Conversion of EXTENDED numbers, to full precision if desired, is straightforward and intended to be done in auxiliary software if at all.

B. The decimal field widths are:

1. SINGLE: up to 2-digit exponent and up to 9-digit significand.
2. DOUBLE: up to 3-digit exponent and up to 17-digit significand, with the option of using up to 19 digits in decimal to binary conversion.

C. Two floating point functions perform conversions between binary floating point integers and signed decimal strings. The latter are character strings consisting of a sign followed by one or more decimal digits. Choice of the character code (BCD, ASCII, ...) is left to the implementer.

1. BINSTR converts a binary floating-point integer X to a signed decimal string whose length is at most 9 for SINGLE and 17 for DOUBLE. BINSTR converts zero with its correct sign. In case X is not an integer round X as in "INT". If X is too large to be expressed by a decimal string that fits into the intended destination an Invalid_Operation exception arises and the corresponding trap is invoked if enabled and, if a result must be delivered, the result is a non-decimal string.
2. STRBIN converts a signed decimal string with at most 9 digits in SINGLE, 19 in DOUBLE, to a normalized floating point number X whose value is that of the decimal integer the string represents. If the string contains non-decimal characters, the standard does not yet specify what happens. If the integer cannot be represented exactly in the intended destination, an Invalid_Operation exception arises and the corresponding trap is invoked if enabled; if a result must be delivered it is a NAN.

D. Conversion over the full range of floating point quantities could be required to be done correctly rounded, but the cost of doing so is probably more than its value. What follows is a compromise designed to ensure that conversion is uniform and in error by appreciably less than one unit in the last place delivered, at a cost which is nearly minimal. But correctly rounded conversion should also be regarded as conforming to the standard.

E. The function log_base_10 is required and may be computed from the formula

$$\text{log\_base\_10}(X) = \text{log\_base\_2}(X) * \text{log\_base\_10}(2) \quad .$$

It need only be computed to the nearest integer for this calculation.

F. Within the conversion process arithmetic must be done to extended precision. Systems without extended precision must therefore effect extended floating arithmetic using fixed point arithmetic on 32-bit significands (in systems with only SINGLE precision) or 64-bit significands (in systems with DOUBLE precision) while processing signs and exponents separately.

G. Powers of 10 not exactly calculable in the stated precision shall be procured from values stored in tables. Negative powers shall be obtained by dividing by the corresponding positive powers instead of multiplying. The following are suggestions for tables requiring minimal storage.

1. Systems with SINGLE precision only: 10^{13} can be computed exactly using a 32-bit significand. To cover the range up to 10^{38}, a table with the single entry 10^{26} suffices.
2. Systems with DOUBLE or both SINGLE and DOUBLE precisions: 10^{27} can be computed exactly using a 64-bit significand. To cover the range up to 10^{308} a table of 10^{54}, 10^{108}, and 10^{216} suffices.

H. BINary_floating_to_DECimal_floating. Given binary floating point number X and integer k with $1 \le k \le 9$ for SINGLE precision and $1 \le k \le 17$ for DOUBLE precision, we compute signed decimal strings I and E such that I has k significant digits and, interpreting I and E as the integers they represent,

$$X = I * 10^{E+1-k} = sd.dddddd * 10^{E}$$

where s is the sign of X and the d's are the k decimal digits.

1. If X is ±infinity or NAN deliver a non-decimal string.
2. If X = ±0 then return I = BINSTR(X) and E = BINSTR(0). Otherwise...
3. Remember the sign of X. Let Y = absolute_value(X).
4. If Y is normalized compute U = log_base_10(Y), otherwise let U = log_base_10(smallest normalized number).
5. Remember the current rounding mode. Compute V = INT(U)+1-k with mode RZ. Restore the original rounding mode.
6. Compute W = INT(Y / 10^{-V}), drawing powers of 10 from the table if necessary.
7. Adjust W:

   If $W \geq 10^{k} + 1$ then increment V and go to (6).

   If $W = 10^{k}$ then increment V, divide W by 10 (exactly), and go to (8).

   If $W \leq 10^{k-1} - 1$ and Y was normalized in step (3) then decrement V and go to (6).
8. Return I = BINSTR(W with sign of X) and E = BINSTR(V).

I. DECimal_floating_to_BINary_floating. The decimal floating point number X has the form

$$X = \text{sddddd.DDDDDDD} * 10^{E} \quad .$$

We are given
signed decimal string E
signed decimal string I = sddddddDDDDDDD
integer P indicating how many digits of I are to the right of the decimal point so that X can be written

$$X = I * 10^{-P} * 10^{E} \quad .$$

1. Compute U = STRBIN(I) and W = STRBIN(E).
2. Compute result $X = U * 10^{W-P}$.

HOW TO AVOID ROUNDING ERRORS

David M. Collison, M.A. 2215 W. Broadway, #F226
Anaheim, Ca. 92804 Tel: (714) 956 - 2581

Abstract

Small computers are especially prone to rounding errors. The article describes programming techniques for avoiding them and gives several examples. There are some hints for the user and technical editor.

Why Rounding Errors Occur

Rounding errors are like termites. They leave little mark, typically an occasional error of one in the last place, and are a nuisance to eradicate. This is especially true of the languages used in personal computers. While BASIC and APL are easy to learn, they have disadvantages for business applications which deal with money.

These two languages store variables in floating point notation: a power of two and a fraction, usually between a half and one in size. Storing money as a fraction of a dollar results in rounding errors because a cent cannot be held exactly. A familiar sight in a BASIC program is:

```
LET P = INT(P*100 + .5)/100
```

which rounds P to two decimal places for printing. APL uses the same technique.

The first suggestion is to keep all amounts to the nearest cent or mill. They are now integers, to the limit of accuracy of the computer. The same applies to all constants -- store them as integers if possible. Divide by ten instead of multiplying by a tenth.

Another way to control rounding errors is by differences. Subtracting the current and previous totals tends to cancel out errors. As an example, try expressing three equal amounts of $100 as percentages of the total, $300 -- but the percentages must sum to 100. Rounding the amounts down individually gives 33%, rounding them up gives 34%. Taking the difference between the current and previous total handles each amount in the same way and gives the correct percentage sum. The first percentage is 33 1/3 which rounds to 33. The second is 66 2/3 - 33 (the first) which rounds to 34. The third is 100 - (33+34) which gives 33. This method is used in the British payroll systems to give the correct deductions whatever the fluctuations in the taxpayer's income.

The same technique is useful in plotting routines. The accuracy of a plotter is determined by the step size, typically a few hundredths of an inch, and this is a function of the stepping motors controlling the pen. Calculating the number of steps from point to point may result in an infuriating error: at the end of a plot, the pen does not return precisely to its starting point. Accumulated rounding errors cause this. Using the differences between the distances of each point from the origin will keep the error to one incremental step or less.

Output Conversion

Even if the calculations are correct, errors may still creep into the printing conversion. The routine on the next page converts pence to the old British system of pounds, shillings and pence. It starts with multiplication by a constant which is rounded up. The constant must have as many significant digits as the maximum number of pence and it is safer to have an extra guard digit. No further rounding is necessary. Adding a half after the number of pounds is printed may result in an error. The routine is compact enough for a calculator and the same principles apply to other conversions. Most currencies are now decimal but land measure is in square miles, acres and feet. It is unlikely that this will change to hectares and metres because the whole country is plotted on a mile square grid.

This insistence on accuracy is hardly necessary for personal programs but it is essential for some businesses. Banks can be picky. A savings and loan made an error of one cent in my account. At the end of the day their balances did not match. They found the error and sent a notification through the mail.

```
10 REM POSITIVE ADDITION OF MONEY
15 REM E.F = A.B + C.D
20 REM A,C,E ARE IN TENS OF DOLLARS
30 READ A,B,C,D
35 DATA 9000300, 6.25, 5000500, 5.92
40 PRINT A;B
50 PRINT C;D
55 LET B=B*100
60 LET D=D*100
65 LET G=INT((B+D)/1000)
70 LET E=A+C+G
75 LET F=B+D-G*1000
90 LET F=F/100
95 PRINT E;F
90 END

READY


:RUN
 9000300    6.25
 5000500    5.92
 9000801    2.17

READY
```

Double length BASIC arithmetic

Old Sterling: 1 pound = 20 shillings = 240 pence
1 shilling = 12 pence

Step	Description	Value
	Starting value in pence	99999
1	Multiply by 1/240 = 0.0041667, rounded up. This is sufficient for 5-digit accuracy.	416.66583
2	Print the number of pounds, to the left of the decimal point	416
3	Take the fractional part of the number	0.66583
4	Multiply by 20	13.3166
5	Print the integer number of shillings	13
6	Take the fractional part	0.3166
7	Multiply by 12	3.7992
8	Print the integer number of pence	3

Routine to convert pence to old sterling

When I visited them a week later, the computer terminal printed the correction automatically. It cost the bank several dollars to correct a one-cent error.

Extended Precision

There is a more serious problem for large amounts of money -- accuracy. Ignorance of the limit can require a hasty reprogramming job, as we shall see presently. A typical 32-bit computer uses 8 bits for the power of two or exponent, 1 bit for the sign and 23 bits for the fraction or mantissa. This handles a maximum amount of $83,886.07 to the nearest cent. It is safer not to approach this limit too closely as some machines round automatically. One such method forces the least significant of the mantissa to one after an operation. This results in unbiassed scientific calculations but is unsuitable for business applications.

Since dollar amounts are usually positive, it is simple to program extended precision in BASIC. The routine on the previous page has separate variables for cents and tens of dollars, giving three extra digits. This permits the use of the standard output package which prints a digit to the left of the decimal point even when it is zero. The space between the two parts of the number can be removed by writing a special print routine. A neater solution is to modify the output package so that there is a non-printing format separator. This allows two numbers to be printed as one with the full range of formats.

This method is cumbersome for subtraction and negative numbers -- it is simpler to put the amount in standard form only for printing. APL users will find vector arithmetic convenient. An alternative is to use a version of BASIC with optional extended precision. VS BASIC permits 15-digit accuracy by specifying a parameter. This applies to all variables but unfortunately not everyone can afford an IBM 360 or 370 for personal use. DTC BASIC permits classes of variables to be defined as double precision or integer. This makes it quite powerful and two examples are given on the following page. The upper limit would satisfy the Federal Government's spending requirements.

While double precision is certainly convenient, it is also inefficient. A floating point addition typically has the following operations:

1. Compare the exponents of the two numbers to be added
2. Shift the number with the smaller exponent down until the exponents are equal
3. Add the numbers
4. If the mantissa exceeds one, shift it down one place and add one to the exponent
5. If the mantissa is less than a half, shift it up one place and subtract one from the exponent. Repeat as necessary

This is a roundabout way of adding two whole numbers, and testing for zero and overflow adds to the time.

Other Languages

One alternative is to use a language with integer variables. FORTRAN, COBOL and PL/1 have 31-bit integers, although the length may vary with the computer. The maximum amount is over twenty million dollars, to the nearest cent. FORTRAN and PL/1 also permit double precision. PL/1 allows closer control of the printing format than FORTRAN which is desirable for business applications.

If other languages are unavailable, it may be possible to write assembly language subroutines. This was popular when computers were less powerful and programming was cheaper. The added effiency paid for itself. The hobbyist has a decision to make: is it better to upgrade the computer or spend a month or two programming? The answer will depend on the time available, his pocketbook, his skill as a programmer, the documentation of the compiler and other factors. It is a tribute to his enthusiasm that there are so many decompilers around -- routines for dumping memory in various formats so that he can patch the compiler to run his programs. The easy way is to trade the computer for a fancier model.

Published Programs

In summary, we have examined various ways to avoid rounding errors, especially in business applications. However the errors still crop up in published programs. If the writeup has no discussion of limits or accuracy it is safer to assume the worst. Another warning is the example for illustration. If it is absent or unduly simple, there may be a reason -- the example should exercise most of the program. On the other hand, I have been pleasantly

```
10 REM POSITIVE ADDITION USING INTEGERS
20 REM B.F = A.B + C.D    B,D,F HAVE 4 DIGITS
30 DEFINT A-G    'A-G ARE 15-BIT INTEGERS
40 READ A,B,C,D
50 DATA 20000,1234,10000,9987
60 PRINT A;B
70 PRINT C;D
80 LET G=(B+D)/10000
90 LET E=A+C+G
100 LET F=B+D-G*10000
110 PRINT E;F
120 END
```

```
  20000  1234
+ 10000  9987
= 30001  1221
```

```
10 REM ADDITION OF MONEY, DOUBLE PRECISION
20 REM DEFINE VARIABLES STARTING WITH A-C AS DOUBLE LENGTH
30 DEFDBL A-C    'LIMIT IS 16 DIGITS
40 READ A,B
50 DATA 1234567890.12,1000000002.98
60 C=A+B
70 PRINT USING "##########.##";A
80 PRINT USING "##########.##";B
90 PRINT USING "##########.##";C
100 END
```

```
  1234567890.12
+ 1000000002.98
= 2234567893.10
```

Double length arithmetic using DTC Extended BASIC. Note the DEFINT & DEFDBL statements.

surprised by the quality of some software. By becoming discriminating users we can improve the software available to us.

Acknowledgments

My thanks go to Kevin Nixon of TMI Enterprises for technical assistance.

MATHEMATICAL PROGRAMMING ON A MICROCOMPUTER WITH HIGH RESOLUTION GRAPHICS

Christopher L. Morgan, Associate Professor of Mathematics,
California State University, Hayward, CA 94542

In this talk I shall describe and illustrate how high resolution, low cost computer graphics can help the mathematician develop his ideas and communicate them to the lay person.

Somebody once defined mathematics as the study of all possible patterns. Patterns studied by mathematicians have largely been inaccesible to the lay person. This is because of the form in which these patterns have traditionally been expressed, namely as abstract concepts usually involving complicated algebraic formulae and careful logic. Mathematicians, however, tend to think spatially (that is, in pictures) and then translate those ideas into words and formulae.

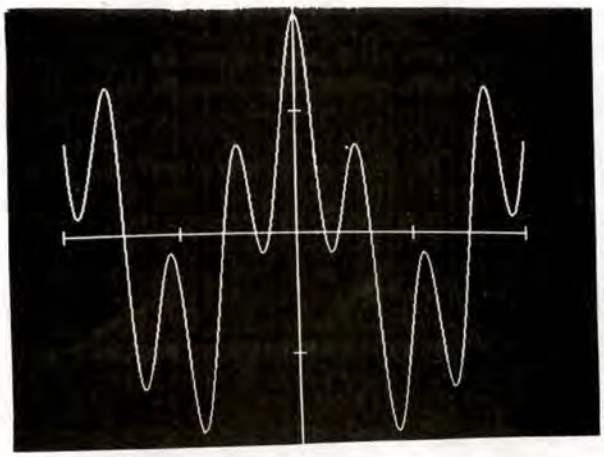
A computer is designed to handle complicated formulae and is built on careful logical principles and thus is a perfect device for working out examples of such mathematical theories. With the advent of high resolution computer graphics, computers can turn these formulae and logic back into pictures. This can give the lay person a much better feeling for the beauty of the mathematician's ideas and give the mathematician a much better idea of how well his output matches his original thoughts. For this process to be effective, a mathematician should have such a system in or near his work area. This is now easy to arrange (for a few thousand dollars) with the advent of microcomputers which have low cost, high resolution graphic output.

For the last few months I have been working with such a system. I would like to discuss some of the results I have obtained so far. First I shall briefly describe the system. The computer is a standard Sol-20 with 56K bytes of RAM, a Micropolis floppy disk drive (Mod II), and a set of two boards by Digital Graphics Equipment to produce raster scan video graphics with 32K of RAM dedicated to the screen. There are actually 30 graphics formats, three of which I have found useful so far. These are: 1) 480 lines with 512 pixels per line, 2) 454 lines with 576 pixels per line, and 3) 408 lines with 640 pixels per line. In each of these formats, one bit controls each pixel. In other formats there are more bits per pixel, giving as much as 16 levels of gray scale. Because there is only 32K RAM, resolution must be sacrificed to get this gray scale (becoming approximately 256X256).

The 32K video RAM is accessed through a 2K or 8K window, and paging is done through an output port.

My first job was to build routines for initializing the graphic system, clearing the screen, plotting points, and drawing lines. In order to get reasonable speed, these routines needed to be in machine language. These routines were designed to be called as functions from BASIC. They now reside in the area above the Sol-20 monitor to give plenty of room for the DOS, BASIC, and BASIC programs in the 48K which lies below the SOL-20 monitor.

Now let me describe some of the mathematical uses for this system so far. The most obvious one is to plot functions:

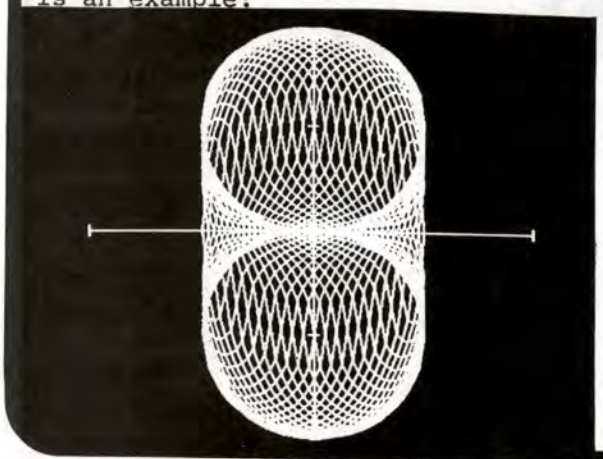


```
10 REM - FUNCTION
100 REM - DEFINE THE LOCATIONS OF
110 REM - THE GRAPHICS ROUTINES
120 DEF FAI=16RD000
130 DEF FAC=16RD003
140 DEF FAP=16RD006
150 DEF FAL=16RD009
160 REM - FUNCTIONS TO CHANGE TO
170 REM - SCREEN COORDINATES
180 DEF FNA(X)=FIX(320+100*X)
190 DEF FNB(Y)=FIX(204-100*Y)
```

```
200 REM - INITIALIZE AND CLEAR THE
SCREEN
210 X=FAI(FIX(2))
220 X=FAC(0)
230 REM - SET UP COORDINATE SYSTEM
240 X=FAP(FNA(-2),FNB(0))
250 X=FAL(FNA(2),FNB(0))
260 X=FAP(FNA(0),FNB(-2))
270 X=FAL(FNA(0),FNB(2))
280 REM - AND MAKE TIC MARKS
290 FOR I=-2 TO 2
300 X=FAP(FNA(I),FNB(.05))
310 X=FAL(FNA(I),FNB(-.05))
320 X=FAP(FNA(.05),FNB(I))
330 X=FAL(FNA(-.05),FNB(I))
340 NEXT I
400 REM - DEFINE THE FUNCTION
410 P0=3.14159
420 P1=3.7*P0
410 DEF FNC(X)=COS(P0*X)+.8*COS(P1*X)
500 REM - DO THE GRAPH
510 FOR X=-2 TO 2 STEP 0.01
520 Y=FNC(X)
530 IF Y<-2 OR Y>2 THEN F%=0:GOTO 600
540 IF F% THEN
A=FAL(FNA(X),FNB(Y)):GOTO 600
550 F%=1:A=FAP(FNA(X),FNB(Y))
600 NEXT X
1000 END
```

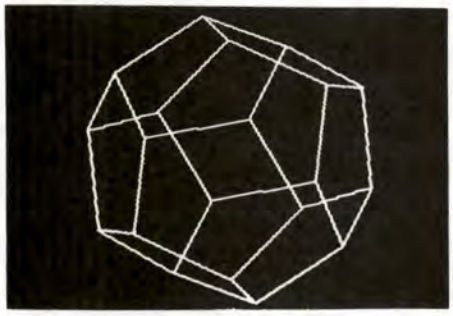
This application can be extended to plotting the graphs of functions of two variables. Using a program described in an article by Mark Gottlieb [1], I have obtained some very beautiful pictures. The ability to easily make these plots is very useful to the mathematician and mathematics students, for the graphs may have much more meaning than just the formulae. It is often desirable to see what effect some small change in the formula has on the graph. With such a set-up one can make changes in the formulae and inspect the graph until it has just the desired properties.

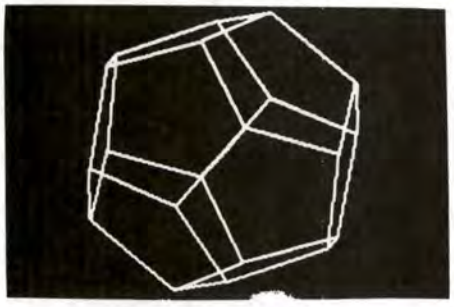
Another variation is to plot the graphs of a whole family of functions on the same xy-coordinate system. Deeper patterns may then emerge. Here is an example:



I call this picture "Addition of Two Circles". A first circle is centered at the origin and has radius one. A second circle of radius one is centered somewhere along the x-axis, say at (c,0). Solve for y in the equations for each circle and add the y values together. Because of the plus or minus signs on the square roots in each formula, there are four distinct y values for each x value (when the formula is defined). Plotting this as x varies and c stays fixed will give a figure eight pattern. The above picture was obtained by plotting this for a closely, evenly spaced sequence of values for c.

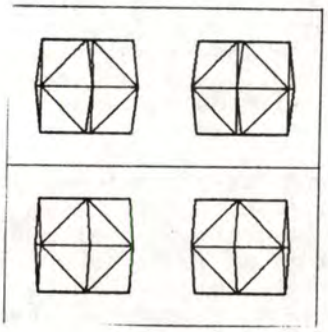
Another application for the mathematician is drawing three-dimensional polyhedra, in this case dodecahedra:

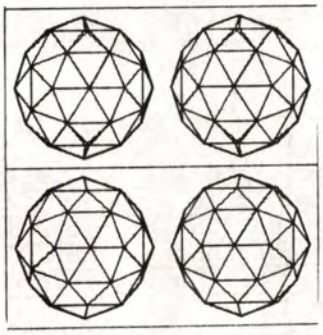


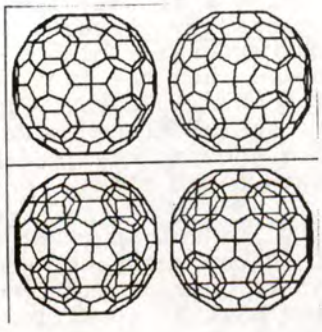


My reason for getting into computer graphics was to draw pictures of regular four-dimensional objects called regular polytopes. Computer graphics helps me study the symmetries of these objects which in turn helps with understanding of four-dimensional space and eventually with the kind of mathematical theory which has been useful in particle physics.

Here are some pictures of these objects made by a digital plotter. I have versions of this program running on the microcomputer system discussed above.







Of course this is just the beginning. There are many more mathematical uses for such a system than the ones I have had a chance to implement in the limited amount of time I have been using the system.

REFERENCES:
1. Gottlieb, Mark, "Hidden Line Subroutines for Three-Dimensional Plotting", Byte, May 1978, Vol. 3, no. 5.

MICROCOMPUTER PROGRAM CORRECTNESS

W. D. Maurer, Professor
Department of Electrical Engineering and Computer Science
George Washington University
Washington, D. C. 20052

Introduction

Many people wonder why, since computer science and mathematics are so obviously similar, there is not more mathematical flavor to computer science. Why are we not proving more things about the programs that we write? It is true that there are some branches of computer science in which theorems are proved, but these theorems do not seem to have too much to do with the practical uses we make of computers.

In this paper we will show that, as has been known for at least twelve years, it is, indeed, possible to prove things about computer programs, and we will apply the methods by which we do this to the specific task of proving things about microcomputer programs. In doing so we will make a number of extensions of known methods of program proving.

Correctness

The most important thing we would like to be able to prove about a program is that it works -- rather obviously.

We say that a program is correct (this is a technical term) if it does what it is supposed to do. This, of course, requires us to formulate a way in which we can express what a computer program is supposed to do.

Some programs calculate the values of functions. If we have a program to calculate the square root of a number, we might try to cite the definition of square root as an indication of what this program is supposed to do. The trouble with such a definition is that it is not general enough. Some programs make complex modifications to tables, and these are not describable in terms of a single function, although we might describe them in terms of several functions, each of which specifies the new value of some variable. Even this amount of generality is not enough, though, in the case of programs which modify data structures. Consider, for example, a program which inserts a new element on the front of a list. It is not necessary for us to know what position in storage the new element takes up -- just that it is associated with a pointer to the old first element of the list. Yet this information is precisely what we would need to know if we had to describe the effect of the insertion program in terms of functions giving new values of specific variables in terms of their old values.

It is always possible, however, to describe the behavior of a program in terms of something that must be true when the program is finished. That "something" is called a final assertion. In addition, many programs have what is called an initial assertion, which is something that must be true before the program starts. This is because there are programs that don't work correctly under all conditions -- only under certain initial conditions that must be satisfied.

A program is correct if the following is true:

Suppose the initial assertion of the program (if it has one) is true, and suppose we run the program, starting at the beginning. Then:

(a) it will not get into an endless loop;

(b) it will not encounter a run-time error;

(c) when it finishes, its final assertion will be true.

A program that satisfies only (c) is called partially correct. In this case, if the program finishes (which it might not), then the final assertion will be true when it does. A program that satisfies (a) alone is said to terminate; a program that satisfies (b) alone is said to be clean.

Thus we cannot say that a program is correct in the abstract. We can only say that it is correct with respect to a particular initial assertion, final assertion, and place to start. Some programs might have different initial and final assertions in different contexts, and correctness could be proved with respect to each of these.

The Inductive Assertion Method

The standard method for proving programs correct was developed by Floyd (now at Stanford). Floyd generously gave credit to others in anticipating various facets of the method, but it has clearly been through Floyd and his students and other successors that the computer science world has come to know about correctness.

The inductive assertion method concerns itself (in its more modern formulation) with certain key points in a program. There are many ways to choose these key points, but the simplest is as follows: Look at the transfers of control (GO TO, jump, branch, or conditional statements); see where they go to (their destinations); and then consider only the destinations of the backward transfers (that is, when we go from one point in a program to another one which precedes it). These destination points of backward transfers are what we call the intermediate assertion points of the program.

With each of these points we have to associate an intermediate assertion. This is something that must be true whenever we pass this point in the program.

Once we do that, we must now be concerned with the control paths which go from one assertion point (initial, intermediate, or final) to another, with no assertion points in between. These are all forward paths -- that is, they can go only forward, although they can also jump forward. The moment a control path jumps backward, however, it comes to an assertion point (by our discussion above), and this is where the path stops. It should be clear from this that there can only be a finite number of distinct control paths and that they have bounded length (in fact no path can have more statements in it than the program itself). Furthermore, every execution of a program must consist of an execution of a sequence of these control paths. (This is the point of using the destinations of backward transfers as assertion points.)

We now have to be concerned with the verification condition of each control path. This is the statement that:

If the path is started at the beginning,
and if its own initial assertion is true at that point,
and if that path is actually followed to its end,
then there will be no run-time error in the path,
and, when the path is finished, its own final assertion will be true.

Note that a control path always has its own initial and final assertion, because it always starts and ends at an assertion point (initial, intermediate, or final) at which there is an assertion. (In particular, the initial assertion of the program is not the initial assertion of every control path, and the same for final assertions.)

If the verification conditions of all the control paths of a program hold, then that program is partially correct and clean. We will now prove this.

Suppose that we start at the beginning of the program, and suppose that the initial assertion of the program holds. Eventually, we will get to some assertion point, and we will get there via a control path. Because the verification condition of that control path holds, there is no run time error in that path and its final assertion holds. But the final assertion of that control path is clearly the initial assertion of the next control path. This means that when this second control path is finished, again by the same argument (since the verification conditions of all control paths are assumed to hold) the final assertion of that control path will hold, and there will be no run-time error in that path.

We can now clearly repeat this argument over and over again as we run the program. Every time we get to an assertion point, the assertion there will be true, and there will have been no run-time error so far. This means that, if we ever get to the end (which we might not), the final assertion of the program will then be true, and there will have been no run-time error. This is exactly what we need to show that the program is partially correct and clean.

To complete the proof of correctness of a program after using the above method, all we have to do is to show that it terminates. We will defer discussion of that point to a later section of this paper.

The Modification Index Method

The modification index method is not a method of proving correctness; it is simply a method of expressing verification conditions in a logically correct form. This method will now be illustrated by applying it to a control path in a sample 8080 assembly language program. The program which we will use in this example computes

X = X + Y + Z

and it is as follows:

```
LDA   X
MOV   C,A
LDA   Y
ADD   C
MOV   C,A
LDA   Z
ADD   C
STA   X
```

In order to use the modification index method, we first convert each assembly language instruction above to its algebraic equivalent. We list the program again, with the algebraic equivalents at the right. (The question of whether these are really the proper algebraic equivalents will be taken up in a minute.)

```
LDA   X        A = X
MOV   C,A      C = A
LDA   Y        A = Y
ADD   C        A = A + C
MOV   C,A      C = A
LDA   Z        A = Z
ADD   C        A = A + C
STA   X        X = A
```

Looking at the algebraic equivalents, we see that:

(1) The A register gets five different values. Let us call these A1, A2, A3, A4, and A5.

(2) The C register gets two values, which we will call C1 and C2.

(3) The variable X gets a new value, which we will call X1. There is also the original value of X, which we shall call X0.

(4) The variables Y and Z do not change their values, and we shall call these Y0 and Z0 respectively.

With these changes, the new algebraic equivalents are as follows:

```
LDA   X        A1 = X0
MOV   C,A      C1 = A1
LDA   Y        A2 = Y0
ADD   C        A3 = A2 + C1
MOV   C,A      C2 = A3
LDA   Z        A4 = Z0
ADD   C        A5 = A4 + C2
STA   X        X1 = A5
```

Now let us suppose that this is a control path, with an initial and a final assertion. Specifically, let's set up the initial assertion as X = 5; then the final assertion could easily be X = 5+Y+Z. The verification condition would now be set up in the following way:

(1) We start by converting the initial assertion X = 5 to X0 = 5, because we are referring to the initial value of X by X0.

(2) The final assertion X = 5+Y+Z is converted to X1 = 5+Y0+Z0. Remember that Y and Z do not change their values, which are Y0 and Z0, but X does change its value, and it is the new value, or X1, to which we are referring here.

(3) We can now read off the verification condition from all the information which we have. It is:

```
if X0 = 5
   and A1 = X0
   and C1 = A1
   and A2 = Y0
   and A3 = A2 + C1
   and C2 = A3
   and A4 = Z0
   and A5 = A4 + C2
   and X1 = A5
then X1 = 5+Y0+Z0
```

Such a condition may be greatly simplified by eliminating hypotheses of the form $\alpha = \beta$ and substituting β for α in further hypotheses and conclusions. This process, carried out cumulatively from the start of this verification condition to its end, may be illustrated as follows:

ELIMINATE	AND SUBSTITUTE	PRODUCING
X0 = 5	5 for X0	A1=5
A1 = 5	5 for A1	C1=5
C1 = 5	5 for C1	A3=A2+5
A2 = Y0	Y0 for A2	A3=Y0+5
C2 = A3	A3 for C2	A5=A4+A3
A4 = Z0	Z0 for A4	A5=Z0+A3
X1 = A5	A5 for X1	A5=5+Y0+Z0

leaving us with the reduced verification condition

```
if A3 = Y0 + 5
   and A5 = Z0 + A3
then A5 = 5 + Y0 + Z0
```

Notice that β is substituted for α only when β is a simple variable. We may now further reduce this verification condition by identifying variables (such as A3 above) which appear on the left of the = and also only once on the right. For such a variable β, we substitute β for α on the right, no matter how complex β may be, and eliminate the hypothesis $\alpha = \beta$. The result in this case is

```
if A5 = Z0 + (Y0 + 5)
then A5 = 5 + Y0 + Z0
```

which is obvious.

In a variable like A3 above, the modification index is 3. This is always a single character (a digit or a letter) which is appended to the name of a program variable, in this case A, to produce the name of a verification variable, in this case A3.

Representations of Instructions

In the proof above, we represented MOV C,A as the assignment C = A (as it might occur in BASIC, for example). This is valid, but it is not quite valid to represent ADD C as A = A + C, as we did. There are two reasons for this:

(1) ADD C is not only an assignment to the A register of the 8080; it is also five more assignments, one to each of the status flags. Suppose we denote these by ZS, SS, PS, CS, and AC. Then each of these variables will also get two new values in the above sequence of instructions; we can call these ZS1, ZS2, SS1, SS2, and so on.

As an example, consider ZS, the zero status flag. If we follow A = A + C by ZS = ZERO(A), where the function ZERO is defined by ZERO(0) = 1 and ZERO(n) = 0 for all $n \neq 0$, then this faithfully represents the new value of ZS. We have not done this in the case above, not only to make things simpler, but because, in fact, the assignments to the five status flags are in fact unnecessary in this case.

Suppose, however, that they were necessary. That is, suppose that we had an instruction such as JZ KAPPA which jumps to KAPPA if the zero status flag has the value 1. In such a case, we would have to ask: Does this path actually go to KAPPA at this point? If it does, then, since we can always assume that the path is followed to its end (see the definition of a verification condition), we can include ZS = 1 as one of the hypotheses of our verification condition. If the previous assignment to ZS was of the form ZS = ZERO(e), as it usually will be, then the hypothesis ZS = 1 can be replaced by e = 0, and the hypothesis ZS = ZERO(e) then becomes superfluous.

On the other hand, if the path does not go to KAPPA but continues with the next instruction in sequence (as a path can do), then we can assume that ZS = 1 is not true (otherwise we would have gone to KAPPA). This means that we can assume ZS = 0, or (using the terminology of the preceding paragraph) $e \neq 0$, as one of our verification condition hypotheses.

If we set ZS = α and then later ZS = β (as above, where we had two ADD C instructions) and never use the setting of ZS to α, then there is no reason to have the two variables ZS1 and ZS2 as described above. We have a verification condition hypothesis ZS1 = α', where α' is derived from α by appending to each variable in α its current modification index, we may write α' = AMXE(α), where AMXE (Append Modification Indices in Expression) is a fundamental verification-oriented function. When we come to ZS = β we can simply throw out ZS1 = α', if ZS1 has never been used, and our next hypothesis can be ZS1 = AMXE(β) rather than ZS2 = AMXE(β).

(2) Setting A = A + C is not correct if there is overflow. There are several ways to deal with this problem. One is to define + as meaning addition modulo 256. That is, A + C is taken to mean the result of adding A and C in the usual way (assuming $0 \leq A \leq 255$ and $0 \leq C \leq 255$, which we can always do on the 8080) and then subtracting 256 if the result is greater than 255.

If we do this, the above proof goes through with no change. That is, we have proved that our sequence of eight instructions on the 8080 calculates X = 5 + Y + Z (modulo 256) if X was initially equal to 5.

Another way, which we can use if we are trying to show that in fact there will not be any overflow (which is sometimes useful and necessary), is to treat overflow as a run-time error. Remember that in the definition of a verification condition we have to show that there will not be any run-time error.

The condition that no overflow will occur as a result of the ADD C instruction is $0 \leq A \leq 255$ after that instruction takes place, if + means ordinary addition (not modulo 255). Let us look back at the figure near the bottom of the left-hand column on the preceding page. There we found that the first ADD C instruction corresponded to the algebraic assignment

$$A3 = A2 + C1$$

Now suppose we want to prove that overflow will not take place. This means that we must add a new conclusion to our verification condition, namely

$$(0 \leq A3 \text{ and } A3 \leq 255)$$

Similarly, since there is another ADD C instruction, we would have to add another conclusion, namely

$$(0 \leq A5 \text{ and } A5 \leq 255)$$

Of course, if we made these changes in our example, the verification condition would no longer be true. In order to make it true, we would have to supply additional initial assertions in our control path.

Termination

There are many special techniques that can be used to prove that programs terminate. For example, a program with no loops always terminates.

Many programs have only one loop, or, more generally, only one intermediate assertion point. A special method applicable to such programs involves finding an integer expression which always decreases as we go around the loop and is never negative. Such an expression is called a loop expression. A program with a loop expression always terminates. Let us prove this.

Suppose the program does not terminate; that is, it goes into an endless loop. We have assumed that there is only one intermediate assertion point, so this point must be in the endless loop. Every time we pass this point, our loop expression has a new value. Let us call these values v_1, v_2, v_3, and so on ad infinitum.

All the v_i are integers (by what we have assumed); they are non-negative (by what we have assumed), and, since the loop expression always decreases as we go around the loop, each v_i is less than the one before it. But this is simply impossible. Think a minute: if the first v_i is 10 (say), then the next few can be 9, 7, 6, 3, 2, ... but eventually one must be negative. (Remember that the v_i are strictly decreasing; we cannot have two the same.) Even if the first v_i is 1,000,000, eventually some v_i will be negative (in fact, the 1,000,001st v_i must be negative). So our hypotheses are impossible, and the program cannot go into an endless loop.

To prove that the v_i cannot be negative, it is enough to have an assertion at the intermediate assertion point which implies that the loop expression is never negative, and then to prove all the verification conditions as before. (Remember that this not only proves that the program is partially correct and clean; it also proves that every assertion will always be true at its corresponding assertion point, provided the program was started with its initial assertion valid. This is exactly what we need here.)

To prove that the loop expression actually decreases as we go around the loop, we again use our modification indices. Suppose, for example, that the loop expression is N-I. At the beginning of the path, this may be represented by N0-I0 since we are talking about the initial values of N and I. Now suppose (as an example) that N does not change its value in this path, but that I does. If I gets exactly one new value, this is I1, and N-I becomes N0-I1. Now all we have to do is to add to our verification condition a conclusion that says that the final value N0-I1 is less than the initial value N0-I0. Thus our conclusion is

$$N0-I1 < N0-I0$$

which reduces to

$$I1 > I0$$

In fact, N-I is a commonly used loop expression, particularly in loops from I = 1 to N. There can always be an assertion $1 \leq I \leq N$ in such a loop, and this implies that N-I is never negative. Also, when I increases by one in the loop, the value of N-I decreases by one.

For programs with more than one intermediate assertion point, there is the general method of loop expression sequences. At each assertion point there is a sequence of integer expressions, none of which are ever negative at that point. The condition that the expressions must decrease as we go around a loop is replaced by the following condition. Suppose that, at the start of a control path, we have an assertion point with associated sequence A,B,C,D,... and at the end of that path, we have an assertion point (possibly, but not necessarily, the same one) with associated sequence W,X,Y,Z,... where A,B,X,Y, and so on are expressed in terms of verification variables (with modification indices). Then we must have:

```
either A>W
or (A=W and B>X)
or (A=W and B=X and C>Y)
or (A=W and B=X and C=Y and D>Z)
or .....
```

(We say that the first sequence must be lexicographically greater than the second.)

The proof, in this case, that the program must terminate resembles the proof above. This time each v_i is a sequence of values of expressions in a loop expression sequence. Every time an assertion point is passed, we evaluate the sequence we find there to get a

new v_i. The fact that the conditions we have assumed produce an impossible situation (so that the program must in fact terminate) is still true but a little bit harder to see informally. Mathematically, we are using the fact that sequences of non-negative integers, in lexicographical order, form a well-ordered set.

Self-Modification

One type of error whose absence we must prove is that of the program which overwrites one or more of its own instructions. This is done as follows.

Let ICA (instruction constancy assertion) be the assertion that every instruction word of the program (that is, every memory byte that contains an instruction, or part of an instruction) has the same contents it had when the program started. Obviously, when the program starts, ICA is true. We want to prove that ICA remains true throughout the entire running of the program, as part of our proof of correctness.

To do this, we can assume, at each instruction, that ICA is true, and then prove that it is still true after that instruction is executed. Of course, if ICA is true, then, in particular, the instruction we are executing has not been overwritten, a fact which we obviously need. We must now show that this instruction cannot overwrite itself or any other instruction. For some instructions (load, add, subtract, etc.) this is obvious. For others, such as a store instruction, it is still obvious if there is only one place that this instruction can store data, and if that place is not, itself, an instruction word.

There remains the case of instructions which store data in an indexed location, and the like, such as MOV M,C on the 8080. For such instructions, we must have assertions that tell us that the index (the HL register, in this case) is within some range, and that this range does not overlap any instruction words. Sometimes (though not always) the range is just a single memory byte, as when we have just loaded the HL register with the address of some simple variable.

A generalization of this method allows us to prove the correctness of programs which actually do modify themselves. Most microcomputer programs do not do this, however.

A Complete Worked Example

We shall now prove the correctness of a program to do Euclid's algorithm on the Motorola MC6800.

Let us first explain Euclid's algorithm, which finds the greatest common divisor (GCD) of two numbers. Suppose the numbers are 16 and 10 (for example). The divisors of 16 are 1, 2, 4, 8, and 16. The divisors of 10 are 1, 2, 5, and 10. The common divisors (that is, the ones which are divisors of both 16 and 10) are 1 and 2. The greatest common divisor is therefore 2.

Euclid's algorithm operates by successive subtraction and crossing out the highest remaining number. That is:

```
 16  10        16-10 is 6
 16  10   6    (Cross out the 16)
-16- 10   6    10-6 is 4
-16- 10   6   4  (Cross out the 10)
-16--10-  6   4  6-4 is 2
-16--10-  6   4  2  (Cross out the 6)
-16--10--6-  4   2  4-2 is 2
-16--10--6-  4   2  2  (Cross out the 4)
-16--10--6--4-  2   2  At this point the
                       process stops, be-
```

cause the last two numbers (2 and 2) are the same; and 2 is the answer.

Now let us write a 6800 program to do this. We are going to keep the two numbers we are currently subtracting in the A and B registers. Initially, these two numbers are M and N, where we are calculating GCD(M,N); so we start by loading A with M and B with N. Now we have to compare A and B to find out whether we want to subtract A-B or B-A. If A=B, of course, the process stops. If A > B, then we want to subtract A-B and we want to cross out A, which will be the largest number. The easiest way to cross out A in this case is to give it the new value A-B. On the 6800, this can be done in one instruction, namely SBA.

On the other hand, if A < B, then we want to subtract B-A and cross out B. We can do this by setting B = B-A, just as before, but there is no one instruction to do this. Instead, we store A in memory, and then subtract that from B.

Whether we set A = A-B or B = B-A, as soon as we have done this, we want to loop back to the point at which we are comparing A and B. Our algorithm may therefore be expressed as follows (here each instruction is given with its algebraic equivalent, and we define ZERO(n) = if n=0 then 1 else 0 and CARRYUS(n) = if n<0 then 1 else 0 -- the carry for unsigned subtraction):

```
    LDAA M      A = M
    LDAB N      B = N
L1: CBA         Z = ZERO(A-B),
                C = CARRYUS(A-B)
    BEQ  L4     if Z=1 go to L4
    BLS  L3     if C=1 or Z=1 go to L3
    SBA         A = A-B
    BRA  L1     go to L1
L3: STAA T      T = A
    SUBB T      B = B-T
    BRA  L1     go to L1
L4:
```

Further conventions here are that T is a temporary cell, Z is the zero status flag, and C is the carry status flag.

The steps in proving correctness here are as follows:

(1) The final assertion is A = GCD(M, N) (also B = GCD(M, N), if we wanted to, but we don't); this is where the answer is left -- in the A register.

(2) The initial assertion is M>0 and N>0, because you can't take the GCD of quantities that aren't positive. Note that if the program had checked M and N for being positive, we wouldn't need any initial assertion. As it is, the program won't work unless M>0 and N>0, so this has to be the initial assertion.

(3) The only backward transfers are the BRA L1 instructions, so L1 is the only intermediate assertion point.

(4) What is the assertion at L1? In order to find this out, we first have to know, informally, why this program works. (Correctness proof is a way of formalizing our informal ideas as to why a program works. It is not a magic method of finding out why programs work when we don't really know why they work ourselves.)

Suppose we had started with the numbers 10 and 6 instead of 16 and 10. It should be obvious that the process, in this case, will be exactly the same, and, in particular, the answer will be the same. This must mean that GCD(16, 10) = GCD(10, 6), or, in general, that

$$GCD(I, J) = GCD(J, I-J)$$

But if this is true, then it gives the key to why Euclid's algorithm works. In fact, we have, by the above formula,

$$GCD(16,10) = GCD(10,6) = GCD(6,4) = GCD(4, 2) = GCD(2,2)$$

and GCD(n,n) = n for any n > 0, because n divides n, and no number larger than n can divide n.

Thus we see that GCD(A,B) is an invariant of our 6800 program, and the easiest way to say that is to say that GCD(A,B) = GCD(M,N) -- the GCD of our original two numbers. In addition, we must have A>0, B>0, M>0, and N>0, because otherwise GCD(A,B) or GCD(M,N) would not be defined. Thus the assertion at L1 -- the condition that is always true whenever we get to L1 -- is

GCD(A,B) = GCD(M,N) and A>0
and B>0 and M>0 and N>0

(5) Now let us look at the control paths, and their corresponding verification conditions:

(a) The first control path starts at the beginning of the program, where the assertion is M>0 and N>0; it goes through the instructions LDAA M and LDAB N; and then it comes to L1, where the assertion is as above. The verification condition is therefore

```
if M0>0 and N0>0
   and A1=M0 and B1=N0
then GCD(A1,B1)=GCD(M0,N0)
   and A1>0 and B1>0
   and M0>0 and N0>0
```

and this clearly holds.

(b) The second control path starts at L1 and takes the branch to L4 from the BEQ L4 instruction. The assertion at L1 is as before, and the assertion at L4 is A=GCD(M,N). The verification condition is therefore

```
if GCD(A0,B0)=GCD(M0,N0)
   and A0>0 and B0>0
   and M0>0 and N0>0
   and Z1=ZERO(A0-B0)
   and C1=CARRYUS(A0-B0)
   and Z1=1
then A0=GCD(M0,N0)
```

We have ZERO(A0-B0) = 1, so A0-B0 = 0, giving A0 = B0 and GCD(A0,B0) = GCD(A0, A0) = A0. So this clearly holds.

(c) The third control path starts at L1, takes the branch to L3 from the BLS instruction, and then does the last three instructions of the program and comes back to L1. The assertion at L1 is as before, and therefore the verification condition is:

```
if GCD(A0,B0)=GCD(M0,N0)
   and A0>0 and B0>0
   and M0>0 and N0>0
   and Z1=ZERO(A0-B0)
   and C1=CARRYUS(A0-B0)
   and Z1=0
   and (C1=1 or Z1=1)
   and T1=A0 and B1=B0-T1
then GCD(A0,B1)=GCD(M0,N0)
   and A0>0 and B1>0
   and M0>0 and N0>0
```

We have C1=1 or Z1=1; but since in fact Z1=0, we must have C1=1, or CARRYUS(A0-B0)=1. Thus A0≤B0; but Z1=0 implies A0≠B0, so A0<B0. This, combined with B1 = B0-T1=B0-A0, implies the conclusion B1>0. We also have GCD(A0,B1)=GCD(A0,B0-A0)=GCD(B0,A0)=GCD(A0,B0) (by another obvious property of GCD) = GCD(M0,N0). The rest of the above verification condition is obviously true.

(d) The fourth control path starts at L1 and does the instruction at L1 and the four instructions immediately following L1, finally branching back to L1. The assertion at L1 is as before, and the verification condition is therefore

```
if GCD(A0,B0)=GCD(M0,N0)
   and A0>0 and B0>0
   and M0>0 and N0>0
   and Z1=ZERO(A0-B0)
   and C1=CARRYUS(A0-B0)
   and Z1=0
   and (C1=0 and Z1=0)
   and A1=A0-B0
then GCD(A1,B0)=GCD(M0,N0)
   and A1>0 and B0>0
   and M0>0 and N0>0
```

We have CARRYUS(A0-B0)=C1=0, so that A0>B0 and A1=A0-B0>0. Also, GCD(A1,B0)=GCD(B0,A1)=GCD(B0,A0-B0)=GCD(A0,B0)=GCD(M0,N0). The rest of the above verification condition is obviously true.

Thus the above program is partially correct and clean.

(6) Finally we consider termination. In this program the value of A sometimes decreases, and the value of B sometimes decreases, as we go around the loop. Neither one always decreases; but either one or the other always does. So let us use A+B as our loop expression. This is an integer expression; it is never negative (since A0>0 and B0>0 from the above); and it always decreases as we go around the loop. Specifically, in the third path above, we have to show that A0+B1<A0+B0; while, in the fourth path, we have to show that A1+B0<A0+B0. But both of these are obvious.

Thus the above program terminates, and thus it is correct since we already know that it is partially correct and clean.

It should be noted that CARRYUS is different on the 8080. In fact, on the 6800, CARRYUS(n) = if n<0 then 1 else 0, whereas on the 8080, CARRYUS(n) = if n<0 then 0 else 1. Thus, for example, 4-3 produces carry on the 8080 and 4-5 does not, while the reverse is true on the 6800.

Notes

Floyd [1] first popularized the inductive assertion method. Generalizations of Floyd's methods of proving termination were given by Manna [2]. Recently an alternative method, the intermittent assertion method, has been developed by Manna and Waldinger [3]. The term "partial correctness" is due to Manna [4].

The modification index method is described in [5] and [6]. A more complete description of self-modification considerations for machine language programs appears in [7].

Many further extensions of these methods are possible. A general discussion of what can and cannot be done in proving programs correct may be found in [8].

Acknowledgments

The research herein reported was partially supported by National Science Foundation Grant MCS76-24280 and by National Aeronautics and Space Administration Grant NSG-1472.

References

1. Floyd, R. W., Assigning meanings to programs, Proc. Symp. Applied Math. 19 (Mathematical Aspects of Computer Science), American Mathematical Society, Providence, R. I., 1967, pp. 19-32.

2. Manna, Z., Termination of programs represented as interpreted graphs, Proc. 1970 Spring Joint Computer Conference, pp. 83-89.

3. Manna, Z., and R. Waldinger, Is "sometime" sometimes better than "always"? -- Intermittent assertions in proving program correctness, Communications of the ACM 21, 2 (Feb. 1978), pp. 159-172.

4. Manna, Z., The correctness of programs, J. Computer and Systems Sci. 3, 2 (1969), pp. 119-127.

5. Maurer, W. D., A new method of generating verification conditions, Proc. 1977 Conf. on Information Sciences and Systems, Johns Hopkins Univ., Mar. 1977, pp. 128-133.

6. Maurer, W. D., The modification index method of generating verification conditions, Proc. 15th Annual ACM Southeast Regional Conf., Biloxi, Miss., April 1977, pp. 426-440.

7. Maurer, W. D., Some correctness principles for machine-language programs and microprograms, Proc. 7th Microprogramming Workshop, Palo Alto, Calif., Oct. 1974, pp. 225-234.

8. Maurer, W. D., Software systems design and correct software, Proc. IEEE COMPCON77 (Spring), San Francisco, Calif., Feb. 1977, pp. 194-197.

GETTING THE WONDERS OF UCSD PASCAL GOING ON AN S-100 SYSTEM

By Jim Gagne, Los Angeles, California

ABSTRACT: UCSD Pascal is an incredibly rich collection of both machine-independent (thus relatively obsolescence proof) software and the best small computer language around (see the August 1978 Byte). It is relatively easy to get up on an S-100 system that already has CP/M, but it does require some doing for optimal results, especially if you have nonstandard equipment, since the system has originally been configured for the PDP-11 environment. This is your chance to hear how it went from a consumer's point of view.

Since I am still optimizing version 1.4 (received about June, 1978), and hope to have version 1.5 (to be shipped about October 1) running by the time of the Faire, this report is still somewhat preliminary. Yet the machine-specific software required should be identical in either case, requiring little if any modification for version 1.5.

If you don't have an 8-inch disc system with CP/M or similar disc operating system, read no further, because as of right now no other 8080/Z-80 disc systems are supported. And until the P-machine interpreters for other microprocessors become available (an announced intention of both UCSD and Byte), no other chips are supported except the Heathkit PDP-11.

If you have CP/M up and running in sufficient memory (over 48 K), you need nothing more to use Pascal, as a CP/M-compatible loader (filename "PASCAL.COM") is provided on a separate disc (along with last-minute documentation and a file called "PGEN.COM", which is a sort of SYSGEN for Pascal that runs under CP/M). All you do is load the disc (called, appropriately enough, "CPM"), type PASCAL, and go. Yet a number of desirable features are lost if you don't create your own BIOS for Pascal, such as the following:

1. If your terminal (or VDM software) doesn't recognize an ASCII 09 (horizontal tab), you'll lose good-looking output.
2. It is important to have an interrupt-driven keyboard, because when console output whizzes by on your CRT, it won't stop unless your interrupt handler recognizes a "control S" (ASCII 13 Hex) as a "stop all output" toggle (meaning push it once, the output stops; push it again and it goes), and somehow communicates that to your console output routine. Similarly, a "control F" (ASCII 5) is useful to get your console to "dump" or ignore long writeouts.
3. The Pascal system comes with a nifty screen-oriented editor that is of roughly the same power as The Electric Pencil, except that it lacks true right justification and a few other Pencil goodies, while it has some new and desireable features (eg, it always tries to put your cursor in the middle of the screen, where you can see what you're doing). While there is also a line-oriented editor, the screen editor is highly addicting. Unlike the Pencil, which is extremely finicky about what hardware you have and

where it is, the Pascal version works with any good CRT terminal that can be made to do the following under software control: home the cursor, erase from present cursor position to the end of the present line, erase to the end of screen, and move the cursor left, right, up, and down without destroying text. SETUP, a customizing program which binds your particular terminal's control characters and other parameters to the Pascal system, makes the going easy. A somewhat hidden "gotcha" is that you also have to bind into the system a routine for moving the cursor to an X,Y position before the screen editor will work.

4. If your present BIOS hangs up when a disc is not present, you'll wait a while if one drive is empty while in use. Your BIOS needs to just return a "1" in register A if a particular drive isn't READY.

5. It is particularly nice to save things like disc directories and other usually console-only material on a hard copy device if your usual console is a CRT. As it is now, Pascal has no "control P" toggle to make the printer follow the console output, as does CP/M. But this is easy to add.

6. One useful addition to my Polymorphics VTI software has been a "page" mode, where new material is written from the top of the screen once the last line at the bottom was completed. In addition, upon reaching page bottom, the page routine calls the input routine to flash the cursor and let you know that the output has stopped and the system is waiting. When you're ready, just hit any character and the page will fill up again. Nice for scanning text, dumps, etc.

7. Along these lines, I have a "slow down scroll" mode in my BIOS, where a timing loop set by the value of the front panel sense switches is called before the screen will scroll. Also nice for scanning text.

8. Although I have yet to check it out with UCSD, I'm fairly certain that neither the cold boot nor the warm boot code in the BIOS is used, which is only a problem if you've put needed initialization routines there. The solution is to insert them in PBOOT, described next.

9. Once you've played with Pascal under CP/M, you'll want to get Pascal up with its own boot, so you can use the specialized BIOS features described above. A booter consisting of a modification of your CP/M booter (you need read only a portion of one track, not two) called PBOOT is stiched together with PINIT (a second booter, which loads the system interpreter) and your new BIOS is stitched together under DDT in much the same way as second-level CP/M generation. There is ample documentation for this process. Put your UART and other system initialization routines in PBOOT, as there should be plenty of room. The whole system of PBOOT, PINIT, and the new BIOS is then placed on track 0 of your Pascal discs by PGEN.

10. Other differences your BIOS should take into account:

a. I am not aware that there is any specific location for rebooting, and it appears that hitting your RESET button does nothing useful at any time.

b. Any BIOS variables should be stored within the BIOS itself, as I do not believe any reserved area is available for your use, such as 40 thru 4FH in CP/M.

c. Blissfully, Pascal is very forgiving of nonstandard (more than 512 bytes) BIOS sizes, and tells you how to handle them in READ.ME, a documentation file on the CPM disc. You really have all the room you need, without having to waste one or more kilobyte of memory as in CP/M.

A PORTABLE COMPILER FOR A PASCAL-LIKE LANGUAGE

Mark Green
2328 Coldstream Dr.
Burlington Ontario
Canada
L7P 3T3

Introduction

In this paper we look at the problem of portability, and some of its solutions. A program or set of programs is said to be portable if it can be made to work on a machine other than the one it was developed on, in significantly less time than it took to develop it. That is, if I took a week to write a program on an 8080 based system and someone can get it working on a 6800 based system in a half a day then it would be called portable.

The advantages of portability are fairly obvious. From a commercial point of view, if you develop a program and want to sell it, the more portable it is the larger the market will be. Also if a group of people are developing a program and portability is kept in mind from the start, then program developement can take place simultaneously on several different machines.

The disadvantages of portability are not quite as obvious. Portable programs are not as efficient as non-portable ones. They run slower than non-portable ones, and sometimes take up more memory. Portable programs are rarely able to take advantage of special hardware or features on a particular machine. Thus you may not be able to access your widgit controller from a piece of portable software.

In general, portability is a desirable property for a program to have, but it does have its price.

In the next section we will look at several ways of achieving portability. The remaining sections of this paper will present a case study of a portable compiler for a Pascal-like language.

Approaches to Portability

In this section we present three ways of producing portable software.

The first way is to write the program in a standard high level language. If I write a program in Basic it is a fairly easy task to get the program to work on another machine, especially if it has the same Basic. This is the easiest way to produce a portable program, but there are several things to watch out for.

The biggest problem that occurs is that there are usually small differences between high level languages in going from machine to machine. These differences are usually associated with how the language deals with machine level features. Using Basic as an example we can come up with the following trouble areas. The Peek and Poke statements cause trouble because

the addresses you use may not exist on another machine or may do something other then what you intended. Usually Peek and Poke are used for I/O or configuring pieces of system software, on another machine I/O devices may be in different places and the systems software may not be the same. Similarly the use of machine language functions causes the same sort of troubles. If the program is to run on another kind of machine then the machine language won't be the same, therfore, the function is useless. If one of these features must be used then there is only one escape, document its use carefully. Every place where the feature is used should be plainly marked. What the statement is doing and how the same effect can be achieved should also be noted. In general any features that your Basic has but others don't should be steered away from.

There are several problems with this approach to portability. The first is you have to have a language processor (either compiler or interpreter) for the language that the program is written in. Most micros have a Basic interpreter so for Basic this isn't a problem. But all these Basic interpreters don't accept the same version of Basic. If a program written in a high level language is going to be portable then that high level language must be standardized. There is one way around this problem, good documentation. Good documentation means there is a written description of the program outlining the algorithm and data structures used. Also the program should be heavily commented. These comments need not be in the copy of the program that is used, but a commented listing should be included with the program.

Another problem with using high level languages is that there are no standard languages available on micros for some applications. One example of this is artificial intellegence (AI). For a large range of AI programs Basic is very awkward at best. Thus it seems that some other portability method would be best for the programs in this area.

The second way to achieve portability is by the use of abstract machines. This technique is also known as interpretive languages (IL) and has been presented in [1,2]. When using this technique you design an abstract machine that is well suited to the program that you are going to write. In its instruction set you include instructions to do operations that will frequently appear in your program. If you are going to be doing a lot of array processing then you would include instructions to store and load from arrays. Once you have your abstract machine designed you write your program in its machine language. In order to execute this program you have to write a program to simulate your abstract machine. This simulator or interpreter will execute the instructions in your abstract machine program. The key to portability here is that it should be easier to write the simulator or interpreter than it is to write the program. Once the simulator has been written for a particular machine your program can be moved to it without any trouble or extra work.

The advantages of this technique are that you can design a special machine for the problem at hand. This machine will be better suited for the problem then a general purpose language. Since each instruction in the abstract machine represents a high level operation the program will not be very large. In general these programs will be several times smaller than equivalent programs written in machine language or assembler.

The disadvantages of this technique are that a simulator

must be written for each machine that the program is to run on. Also since each abstract machine instruction must be decoded in software the program will run slower than the equivalent program in machine language.

This technique is quite popular for the production of portable compilers. The compiler is written in the abstract machine language and it produces code for the abstract machine. Thus both the compiler and the code produced by the compiler are portable.

The third method is related to the above method, but instead of producing code for an abstract machine, macro calls are produced. A macro is a string replacement mechanism. The program would be written as a sequence of macro calls, each of which is expanded into a number of machine language or assembler statements. The macro calls may have parameters which are included in the statements that are produced. The following is an example of a macro for adding two bytes together, and its expanded 6502 code

```
        ADD A,B

         macro


        LDA A
        CLC
        ADC B
        STA A
```

expanded 6502 code

As can be seen this macro is expanded into four assembler statements, and the parameters A and B are put in the right places in these statements.

The program that does the macro expansion is known as a macro processor. The macro processor takes the sequence of macro calls, looks up their definitions, and from these defintions produces the expanded code. The following is an example of what a macro defintion for our macro ADD might look like

```
MACRO ADD P1,P2
      LDA P1
      CLC
      ADC P2
      STA P1
      ENDM
```

The parameters P1 and P2 are replaced by the actual parameters when the macro is expanded.

If we write our program as a sequence of macro calls then all we have to do in order to move it to another machine is to write a new set of macro definitions for that machine. This assumes that a macro processor is available on the other machine.

Of the three techniques this one produces the fastest code, but it requires a pass through the macro processor. The code proceduced by this technique is not as good as hand produced machine code, but it is not too bad. At the present time there are no general purpose macro processors available on a wide range of micros. Thus until such macro processor exist this is not a usable technique.

Designing an Abstract Machine

In the previous section we said that the abstract machine should be designed with the problem in mind, but how do we do this? As an example we present an overview of the abstract machine that was designed for the Micro Pascal compiler. Micro Pascal is a Pascal-like language that has essentially all the control constructs of Pascal. The simple data types that are available are byte, integer (16 bits), and char. The structuring mechanisms that are available are the array and strings of characters.

Since Pascal is a block structured language with recursive procedures, activation

records [1,3,4] are useful for data organization. Instead of having the compiler deal with activation records we build them into the abstact machine. To accomplish this all the program data is stored on a stack, and the procedure call and return instructions create and remove activation records. The display is built into the abstract machine and all store and load instructions go through the display. This movement of activation records into the abstract machine has two advantages. Since the compiler does not have to deal with them it is both simpler and shorter. Address calculations and activation record management is done in the simulator which is faster than the compiler producing code for these tasks.

While on the subject of address calculations lets look at how array and string elements are accessed. For multi dimensional arrays calculating the address of an array element is a long process. If the compiler produces code for each element reference then the code will be long and slow. Therefore, we have instructions for loading and storing individual elements in the array. There are separate instructions for loading and storing integers and bytes in an array. The subscripts for the element are stored on the top of the stack. At the start of each array the dimensionality of the array, and the range of each dimension is stored. Thus when an array reference instruction is encountered the simulator picks up the address of the array from the instruction and the subscripts from the stack and performs the appropriate operation.

As another example of the types of instructions lets look at the arithmetic instructions. There are five arithmetic instructions for bytes and integers, the four usual arithmetic operations and negation. In the instruction set these 10 instructions are grouped together, first the byte ones and then the integer ones. Both the byte and the integer instructions are in the same order. This simplifies the job of the compiler, since once it determines the operation it just adds five if the arguments are integers, and does nothing if they are bytes. Each of these instructions takes its arguments from the top of the stack, therefore, these instructions are one byte long.

From these examples we can draw four guidelines for designers of abstract machines. These guidelines are:

1) The machine should be oriented towards the applications in mind. It should provide the operations necessary for the application, and do any of the dirty bookwork. An example of this is the activation record management mechansims of the above abstract machine.

2) Each of the instructions should perform a whole meaningful operation. If you find that a certain sequence of operations is being repeated constantly then they are a good candidate for a new instruction. Examples of this are the array referencing instructions.

3) The instructions should be grouped together by purpose. This makes it easier for compilers to produce code for the abstract machine. An example of this are the arithmetic instructions.

4) The instructions shouldn't be so powerful that they do most of the work in the program. The reason for using the abstract machine was so the program would be portable. If the instructions implement most of the program then there will be more work involved in writing the simulator than in writing the program.

As can be seen from the above guidelines each abstract machine instruction should be more powerful than a hardware machine instruction but at the same time it shouldn't implement most of the program.

Writing a Simulator

Now that we have covered the design of an abstract machine lets look at how we write a program to simulate it. We will assume that the abstract machine program has already been loaded into memory. Each instruction is one or more bytes long, but the first byte determines the type of instruction. We will need a program counter for our abstract machine, lets call it IP. At the start of execution IP will point at the first instruction to be executed.

The main loop of our program will perform the following tasks, incrementing IP, getting the next instruction, and calling the routine to execute that instruction. Since the main loop is executed once for every instruction executed it must be as fast as possible. The first byte of each instruction uniquely determines the type of instruction, therefore, we can use it as an index into a table of addresses of instruction processing routines. There will be one entry in this table for each instruction in the abstract machine. The main loop of our program will then get the next instruction, use it as an index into this table to get the instruction routine address, and jump to this address. In this routine the processing required to execute the statement is performed. At the end of the instruction processing routine there is a jump back to the main loop. The following is an outline of such a program:

```
LOOP  : IP = IP + 1
        INDEX = CODE[IP]
        ADDRESS = TABLE[INDEX]
        GOTO ADDRESS

TABLE : ADDRESS OF LOAD
        ADDRESS OF STORE
        ADDRESS OF JUMP
          .
          .
          .
LOAD  : DO LOAD INSTRUCTION
        GOTO LOOP
STORE : DO STORE INSTRUCTION
        GOTO LOOP
          .
          .
          .
```

When the load instruction (a zero) is encountered the first byte will be used as the index into the table. The first item in the table is the address of the load processing routine, thus control will be transfered to it and a load will be performed. At the end of the processing routine we jump back to the main loop to get the next instruction. This loop continues until a stop instruction is encountered and the program stops.

Since IP is incremented before the instruction is fetched the initial value of IP must be the address of the first instruction minus one. Similarly when a jump is encountered IP is set to the jump address minus one. Thus the processing routine for the jump instruction would be

```
JUMP  : GET ADDRESS
        IP = ADDRESS - 1
        GOTO LOOP
```

If the abstract machine has less than 256 instructions the remaining locations in the table should contain the address of an error routine. This is to catch the simulator as soon as a problem develops. If this is not done an illegal instruction could cause the simulator to go off track making it hard to find where the error occured. A helpful error routine would print the current value of IP along with any internal variables to make

the debugging easier.

This technique is the fastest one that I know of for simulating an abstract machine. The one disadvantage it has is that the table of instruction processing routines takes up 512 bytes. The version of this routine that I have programmed for the MOS Technology 6502 takes 28 cycles per instruction executed, which appears to be the minimum possible. If anyone finds a faster way of doing this I would like to hear from them.

Summary

In this paper we have looked at the problem of portability, and presented three solutions to it. We then looked at a particular piece of portable software and the techniques that were used in it. The design and implementation of abstract machines was also discussed.

Acknowledgements

I would like to thank the Computer Systems Research Group at the University of Toronto for supplying a healthy environment for this work. I would also like to thank Dr. N. Solntseff for reading several drafts of this paper. Last but not least I would like to thank my wife for correcting my spelling and grammar mistakes.

References

[1] Bauer F.L. and J. Eickel(ed.) Compiler Construction An Advanced Course, Springer-Verlag, New York, 1976.

[2] Cluff J. Defining LIL, A Little Interpretive Language, Byte vol. 2, no. 10, p.30, 1977.

[3] Gries D. Compiler Construction for Digital Computers. Wiley, New York, 1971.

[4] Aho A.V. and J.D. Ullman Principles of Compiler Design, Addison-Wesley, Reading, Mass., 1977.

VIDEOBRAIN AND THE APL/S LANGUAGE

First Pass At Everyman's Computer

TED HAYNES
VideoBrain Computer Company
2950 Patrick Henry Drive
Santa Clara, California

Abstract

If you like to speculate about what home computers will really be like when everyone has one, you had better hurry up. Speculations are becoming hypotheses and hypotheses are beginning to be tested in the homes of consumers. This talk will address the first set of hypotheses submitted for testing - the VideoBrain Home Computer that has been on the market since this past February. The talk will look at how well the hypotheses seem to be working out and go on to discuss the specific question of what languages the consumer will find optimal.

VideoBrain Hypotheses

1. Home computers will be ubiquitous
2. The greatest demand will be in small computers with both graphic and sound capability retailing for under $500.00
3. The consumer will use his computer for:
   - Education
   - Financial Management
   - Entertainment
   - (not home and appliance control)
4. The consumer will demand that well designed, easy to use, and meaningful software be readily available.
5. The best way to sell everyman's computer is the same way you sell most of everyman's other appliances like microwave ovens and television sets, through department stores and other conventional retailers.
6. The consumer will want to be able to start with an inexpensive computer and upgrade from there.

APL/S Hypotheses

1. Consumers will demand a superior language - one that is clear, concise, is easy to write and debug and faciliates the use of good programming concepts. This implies structured programming using the control words of PASCAL.
2. Rather than a general language, consumer will value application languages that facilitate certain uses. The uses with the greatest perceived value will involve sophisticated number crunching in such fields as financial analysis (e.g., stock, portfolios, tax estimation), science and engineering, statistics, and education (e.g., simulations, learning to program). This implies the ability to handle groups of numbers easily in one dimensional arrays.
3. Modern languages for small computers should be specifically designed for small computers using proven language innovations - rather than a scaled down version of a 15-year old language designed for big computers (i.e., BASIC). An illustration of the advance in language design from BASIC to APL/S is given below:

BASIC

```
10  READ N
20  MV=0
30  FOR I=1 to N
40  READ SHS,PRC
50  MV=SHSxPRC+MV
60  NEXT I
70  PRINT MV
100 DATA 3
110 DATA 100 53.75
20  DATA 50 220.50
130 DATA 75 87.00
RUN
22925
```

APL/S

```
SHS = 100 50 75
PRC = 53.75 220.50 87
+/(SHSxPRC)
22925
```

AN INTRODUCTION TO APL/S

A Modern Computation Language for Personal Computing

ROBERT G. BROWN
777 S. Mathilda Avenue, D148
Sunnyvale, California 94087

Abstract

The purpose of this paper is to introduce APL/S and some of the philosophy behind the design.

APL/S is a problem solving language for use on a personal computer. APL/S is a modified subset of APL with additional structured programming control figures.

APL/S has been implemented on the Video-Brain computer in a removable cartridge with 13K ROM and 1K RAM.

Summary Of APL/S Features

- Structured control figures
  - IF formula THEN block ELSE block ENDIF
  - WHILE formula DO block ENDWHILE
  - Subprograms
- Built in functions
  - +; -;x;÷;*;!;,;LOG;EXP; MAX;MIN; MOD; AND;OR;NOT;LT;LE;EQ;GE;GT;NE; SIN;COS;TAN;ABS;FLOOR;CEILING; RANDOM;INDEX;ARRAY;SIZE
- Floating point
  - 6 to 7 digit accuracy
  - 10^{77} dynamic range
- One dimensional arrays
  - Dynamic allocation
  - Any scalar formula as subscript
- Extension of all scalar functions to arrays
  - Element by element
- Reduction operator
  - With any two argument scalar function
- Character data output
- Immediate mode execution
  - No "PRINT" statement required
- Program trace
  - Displays source statement and results
- Cassette tape storage for workspaces
  - LOAD
  - SAVE
- Bargraph display
  - 7 bars, resolution ±64
  - 16 colors
- Workspace commands
  - LIST
  - ERASE
- Statements are Syntax checked at entry.
- Stop key
  - Freezes scrolling
  - Interrupts a program
- Line oriented program Editor
  - Insert
  - Delete
  - Modify
  - Next
  - Previous
  - Intelligent display (shows program window)

Immediate Mode

In immediate mode whatever is keyed is immediately evaluated and the unassigned results, if any, are displayed.

Key In	Result displayed	Side effect
1+2	3	None
1 MAX 5	5	None
A←÷4	None	Variable A is assigned the value .25
10	10	None

The immediate mode is ideal for experimenting with new functions. Particularly when coupled with array operations, a great deal of useful work can be done in immediate mode.

Arrays in APL/S

An array is an ordered aggregate of numbers where each component element is identified by its position within the aggregate. For example, let A←7 11. Then A is an array of size two, where 7 is the first number and 11 is the second. Because order is significant, B←11 7 is not equal to A, even though B is made up of the same numbers as A, and is the same size. (only one dimensional arrays are considered).

To be concrete, I will show how an array is entered at the keyboard and how it is displayed.

In an APL/S program, the function which causes data to be read from the keyboard is "KEYB". To read in the array A the program statement is:

```
A←KEYB
```

the keyboard entry is prompted;

```
?
```

would appear on the screen. Keying in 7 11 <NEXT>; the screen would be:

```
? 7 11
```

This is read in and returned as the evaluated result of KEYB. Normal assignment statement action then assigns the 7 11 to A. The array A has been read in.

To display the array A, the APL/S statement is:

```
      A
```
The screen will show
```
      7 11
```
The array A has been displayed.

Why use arrays in a programming language?

Even the most primitive machine language is universal in the sense that it can be used to compute whatever is computable. The reason for any higher level language feature is to cater to human needs. Arrays and operations on arrays are defined because with arrays it is easier to:

1. Think about certain kinds of problems clearly.
2. Write the program correctly.

Thinking about the problem

Abstraction is perhaps the single most important method people use for problem analysis. It allows us to (temporarily) ignore the details. An array is a direct language embodiment of this principle. Instead of thinking of the individual numeric values -1,000,000;-500,000; 100,000;500,000; 1,000,000; 1,500,000 we think of "the cash flows". We think of the aggregate.

Mental operations on the aggregate

An example of this type of operation is "income less outgo is cash flow". That is, we do not think of each period as a separate, different case:"cash flow in period 1 is income in period 1 less outgo in period 1, cash flow in period 2 is....." Operations on arrays provide a direct language equivalent of thinking of operations on the aggregate.

To be concrete, the series of user program statements:

```
A = 7 11
B = 2 x A
C = B + A
A
B
C
```

would result in the screen showing:

```
7  11
14 22
21 33
```

Individual elements

For other purposes, we think of individual elements. For example, "the cash flow in the first year will be negative because of high start-up costs and relatively low sales" or "the cash flow in the last year should reflect salvage value". Note in particular that we think in terms of position (first or last) in the collection.

In a language, subscripting provides the equivalent of thinking about elements by their position.

Subscripting is done using parenthesis. For example, with B = 14 22,

```
B(1) = 14
B (1+1) = B(2) = 22
```

Reduction

Another common mental operation on collections of numbers is to add them up. For example, "net present value is the sum of the discounted cash flows". Notice that to think of it in terms of "net present value is the discounted cash flow for period 1 plus the discounted cash flow for period 2 plus...." is to risk getting bogged down in detail. In a programming language, the bogged down approach is to write a loop and use subscripting. The language equivalent of the more powerful mental operation is called reduction (because it reduces an array to a single value).

With B = 14 22, the sum of the elements in B (or reduction by addition) is:

```
+/B = 14 + 22 = 36
```

Any scalar two argument function can be used with reduction. Some of the more frequently used are:

OR/(A LT 0)	result is 1 if any element of A is less than 0.
MAX/A	result is the largest element in A
OR/(5 EQ A)	result is 1 if 5 is one of the elements of A
+/(A x B)	result is the inner product of A and B.

Here is an immediate mode calculation using array operations. The overall resistance of n parellel resistors is:

$$\frac{1}{\sum_{i=1}^{n} \frac{1}{R_i}}$$

If the resistances are in an array R, then the APL/S expression for the overall resistance is: (NOTE: ÷ means reciprocal when used with only a right argument).

```
÷ +/÷ R
```

Programs

The syntax for programs is straightforward.

```
Program: = PROGRAM identifier⊢ block
           ENDPROGRAM
block: = [Statement⊣]...
statement:= { IF formula ⊢ THEN block
              [ELSE block] ENDIF
            | WHILE formula ⊣ DO block
              ENDWHILE
            | Simplestatement }
```

within a block, statements are executed one after another, in sequence.

The IF and WHILE statements have the usual interpretations, acting on formulas which evaluate to true (1) and false (0).

Because each block has a definite terminating keyword, there is no need for Begin...End pairs as in ALGOL or PASCAL.

The statement terminators ⁊ are automatically created at statement entry in response to the <NEXT> key signaling the end of a logical line.

Writing a program

Two important principles in writing programs are:

1. For each abstract solution to a part of the problem, produce a separate concrete realization in the program. Although this is not completely feasible in any current language, I think its fair to say that to the extent it can be done, it helps produce correct programs.

2. Use a concrete representation which is as suggestive as possible of the abstract concept.

These two principles are themselves a bit abstract. I will illustrate their application with some examples.

Compute the average

Abstract solution concepts. Get a group of numbers; add them up; divide the total by the size of the group; display the average.

Program 1. Without arrays.

```
PROGRAM AVERAGE
  TOT = 0
  N = KEYB
  I = 1
  WHILE I LE N
  DO TOT = TOT + KEYB
     I = I + 1
     TOT ÷ N
  ENDWHILE
ENDPROGRAM
```

In program 1, the solution concept "get a group of numbers" has no direct representation. Instead, the programmer has had to translate to "find out how many in the group, then get them one at a time". Same with "add them up". Furthermore "getting and "adding" have been intertwined in the WHILE loop. The "group of numbers" does not have a representation. Two extra variables 1 and N, had to be used. Each of these examples are of the type of discrepancy between solution concept and programmed realization which can lead to incorrect programs.

Program 2. With subscripting

```
PROGRAM AVERAGE
  DATA = KEYB
  TOT = 0
  I = SIZE DATA + 1
  WHILE 0 LT I = I - 1
  DO TOT = TOT + DATA (I)
  ENDWHILE
  AVE = TOT ÷ (SIZE DATA)
  AVE
ENDPROGRAM
```

In program 2, all the abstract solution concepts except "add them up" have a direct realization. Only one extra variable has to be introduced. The "size of the group" is realized in the language construct SIZE DATA.

Program 3. With reduction.

```
PROGRAM AVERAGE
  DATA = KEYB
  TOT = +/DATA
  AVE = TOT ÷ (SIZE DATA)
  AVE
ENDPROGRAM
```

Program 3 shows a clean correspondence to the abstract solution concepts. It contains no extra variables or loops.

Find prime factors

The final example is an APL/S program to compute the prime factors of a number. It is equivalent to a BASIC program given by Kemeny and Kurtz in their book BASIC Programming.

Abstract solution concepts. While the number obtained from the keyboard is a non zero integer, find its prime factors. To find the prime factors of a number N, start with a trial factor, F=2.

If the value tried divides N evenly, then add it to the list of prime factors and replace N by N÷F, otherwise advance F to the next odd integer. Keep this up as long as F squared does not exceed N. When finished, indicate that the number is prime if no factors were found, otherwise display the prime factors in the list and the remaining prime factor, N.

```
PROGRAM FACTOR
  WHILE 1
  DO N=KEYB
     IF (N EQ 0) or (N NE FLOOR N)
     THEN EXIT
     ELSE FPF
     ENDIF
  ENDWHILE
ENDPROGRAM
PROGRAM FPF
    FS = ARRAY 0
    F = 2
    WHILE FxF LE N
    DO
        IF N MOD F EQ  0
        THEN FS = FS, F
             N  = N÷F
        ELSE F = F + 1 + (F NE 2)
        ENDIF
    ENDWHILE
    IF SIZE FS EQ 0
    THEN "IS PRIME"
    ELSE "HAS FACTORS"
    ENDIF FS,N
ENDPROGRAM
```

The main loop in the FACTOR program is ended by the EXIT statement.

In the FPF program, the statement FS=ARRAY 0 initialize the array of prime factors to the empty array, an array with zero elements. When a prime factor is found it is concatenated onto the array of the factors by the statement FS = FS, F.

After execution of the factor program the screen would show:

```
FACTOR
?12
HAS FACTORS
2 2 3
```

Summary

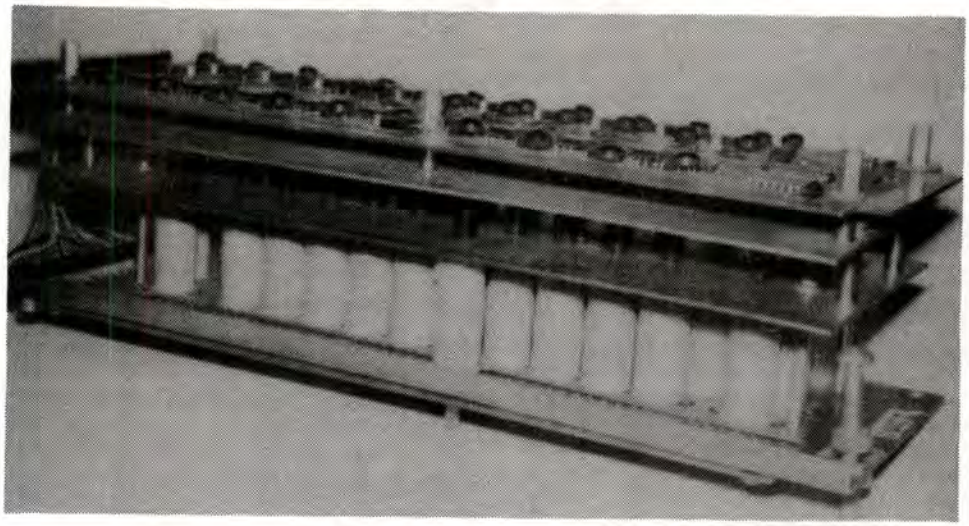
APL/S is a problem solving language based on two widely accepted language facilities: APL array data and structured programming control constructs. It can be readily expanded in either direction.

As a first language it encourages the development of sound programming technique. This will be of benefit regardless of what other languages may be learned later.

The experienced user will benefit from a language which helps him to quickly construct well structured programs.

Acknowledgement

The author would like to acknowledge the assistance of D. Chang, P. Holzmann, F. Wang, C. Cheng, A. Rodgers, and the management of the VideoBrain Computer Company in the production of APL/S.



NOT FLAKY — Lexitron put these solenoid banks over IBM Executive typewriters and until recently used them as printers on $17,000 word processing systems. California Inudstrial recently advertised the units.

Bring your own typewriter:

WHY CAN'T WE HAVE A $49 CORRESPONDENCE-QUALITY PRINTER?

Factories pour out millions of electric typewriters a year. People already have these in their homes, or can get them at low cost. Work performed on these machines, even the cheapest, looks better than that from about any new, $1,500-or-less printer that can be connected to a computer system. What is the possibility of connecting an electric typewriter to computer?

That question has bothered me since the day some years ago I read in an electronics magazine an article about a marvelous new kit that would allow me to own a computer.

Consider the possibilities: A person could walk into a computer or electronics store and buy his ready-made computer. As the clerk is writing up his order, he could point to a $49.95 blister-packed device on pegboard, and say, "And dammit, I'll take one of those, too. Will it work with my SCM portable?"

In order to get the project moving, I assembled a typewriter adapter from rather fancy, and expensive, solenoids. That experience is written up in the article that follows, "Simple Converter...", reprinted from Low-Cost WORD PROCESSING.

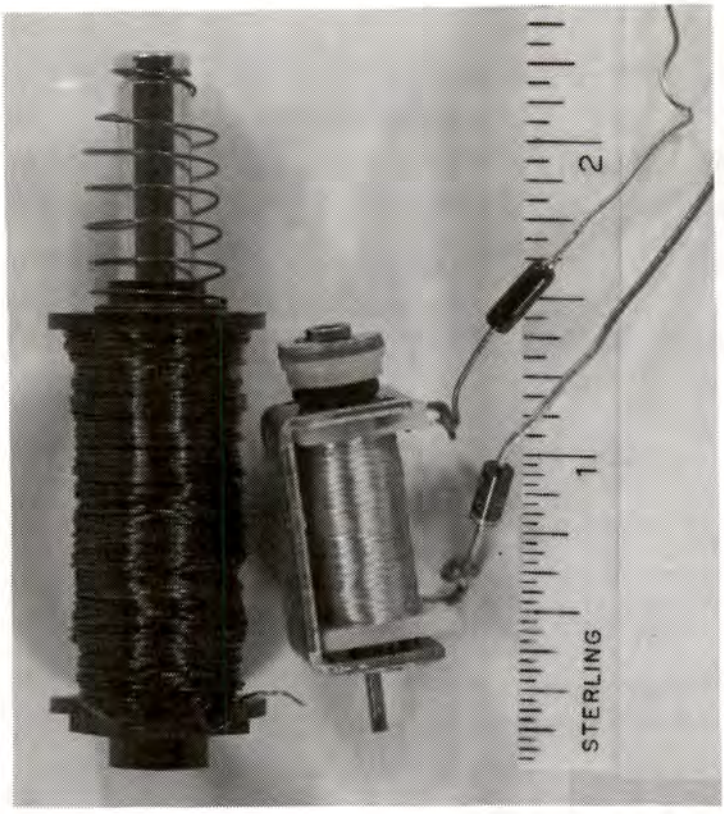
That hardly was the end of the challenge, however. The Hi-G coils, at $130 per set, are obviously too expensive for such a project. There must be product available from someone.

All solenoid manufacturers we could uncover were making fancy, expensive units. Yet inexpensive solenoids find their way into many everyday products. Toys and door chimes come to mind. Door chime manufacturers were queried. A little persuasion, stretching out over a year, and we were sold one set of solenoids, similar to one pictured on the opposite page, for $1 each.

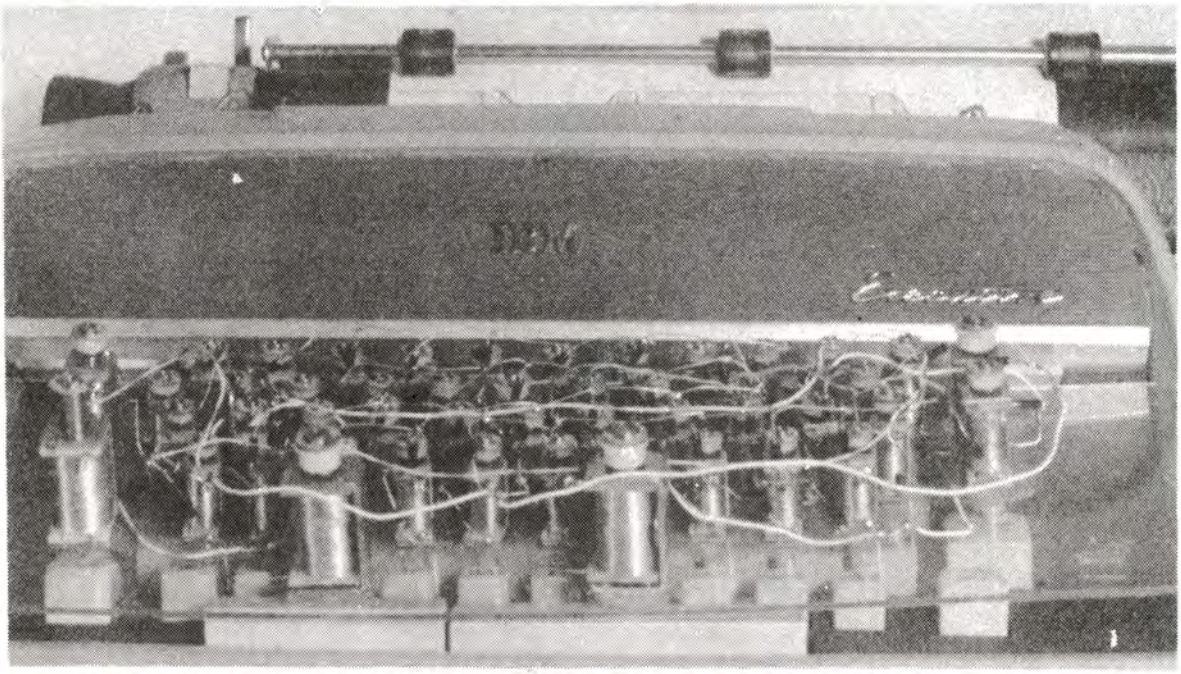
The $50 (or $98) typewriter interface is possible. With such a large market for such a device, it would seem reasonable to assume that eventually someone will market it.

Low-Cost WORD PROCESSING, $12.95 for 12 issues, is published by nonprofit Computer Information Exchange, Inc., Box 158, San Luis Rey CA 92068.

Bill McLaughlin



SOLENOIDS COMPARED — Dollar door chime solenoid, left, is compared to $2.45 Hi-G, to which diodes have been soldered.



COMPLETED TYPEWRITER INTERFACE-Word Processing's solenoid bank is seen in place atop proportional-space IBM Executive model C, which has market value of about $200-350. Headlines of this issue are Executive blowups, and all type in previous issues was set on this machine.

SIMPLE CONVERTER MAKES ELECTRIC TYPEWRITER INTO COMPUTER PRINTER

It can be done. Your electric typewriter, no matter what kind you have, can be used as a printer for your computer system.

If your typewriter produces letter-quality printwork, your computer will be able to do the same.

Your typewriter does not have to be modified. We rejected the idea of hanging the solenoids under the machine, since this presents difficult mechanical, servicing, and reliability problems. So the mechanical interface simply sits on the keyboard and pushes keys.

No, the unit will not break printer speed records, but it probably will be as fast or faster than most Selectric interfaces, since these are usually done improperly, at half speed, with attendant greatly-accellerated wear. Selectric is capable of 15cps operation. An inexpensive Royal 2000, which in our area discounts for $220, can type 13cps–5.5cps faster than an improperly-interfaced Selectric.

At the outset, let us be skeptical that anyone can operate this interface at the theoretical speed limit of a typewriter. Seven-and-a-half to 10cps would seem reasonable, however.

We might also say that it may not be smart for someone to go out and buy a typewriter for this project. We aren't saying you shouldn't do that, either. What we do say is that if you have a typewriter that is in good condition, or reparable, so that it produces good work, this project very well could produce a better printer than you could buy for many times the cost. We also note that used office electrics sell for $80 or $90. Some companies give them to their employes.

Is it fast enough? Read "Speed not as important as printer quality", elsewhere in this Word Processing Letter, and do some serious thinking about required throughput, and about your operating system.

The interface requires only seven of the eight lines normally available on a parallel port. Of these:
-Three feed the one-of-eight row demultiplexers
-Three, the column demultiplexer
-And one the shift signal
The left-over line can be a strobe, if you want one.

All typewriter character and function keys except Shift are arranged in a seven-by-seven diode-separated matrix, illustrated in figure 4. This arrangement was judged to be the easiest to wire. One spare position, row 7 column 7, allows you to operate a signal light, whistle or bell.

All typewriter keyboards are not alike. Manufacturers especially like to move special characters, and the figure "1" around the periphery. These differences should be accommodated in software, not through wiring changes. So do not take seriously the placement of those wierd keys. An IBM "Standard" in this office has fractions where period and comma should be. Were I interfacing it, and I probably will, I would wire it the same as the Executive, figure 5, and correct the odd keys in software.

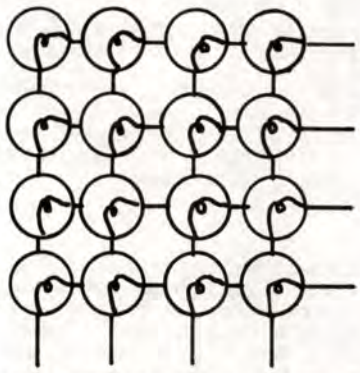


FIGURE 1–Lamp matrix illustrates how keyboard solenoid bank operates. Current between any possible combination of one column and one row lead causes lamp at intersection of those two wires to light. Without diode isolation, however, adjacent lamps glow objectionally.

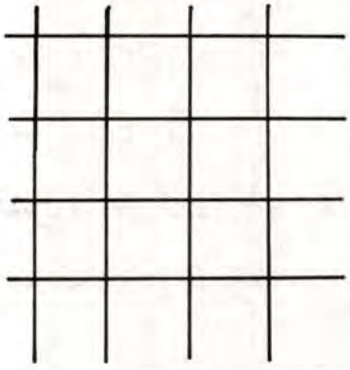


FIGURE 2–Column and row grid is simply wires which do not connect. One side of each filament connect to a row, other side connects to column.

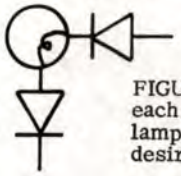


FIGURE 3–Diodes connected to each side of filament isolate lamps so that none glow while desired light burns.

Theory of operation is illustrated, figures 1-3. We have arranged an array of 16 Xmas bulbs, four rows of four bulbs. One side of each filament is connected to a horizontal wire, called a row. The other side of each filament is connected to a vertical wire, called a column. In the case of this 16-lamp array, there are four rows and four columns. If you connect a current between any one row and any one column, one light will burn brightly, and several will

glow. The bright light will be at the intersection of the energized row and column.

For our application, having just most of the current go to the proper solenoid might not be good enough. At worst, more than one key at a time would depress. At best, we would have unnecessary chattering, and possibly excessive loads on the drive circuits.

The solenoid matrix is identical, except it is seven by seven, accommodating up to 49 magnets.

Solenoids selected were the smallest we could find that would operate IBM "C" model typewriter keys. These are the Hi-G series "S" size "A" 24V coils, driven directly from the 120VAC line, no transformer.

Powering 24V coils directly from the 120V line may seem a mistake, but that 24V rating is for continuous-duty operation, and we will be energizing the coils only for short duration, perhaps less than 1 per cent of the time. Under 10 per cent duty cycle, the coils are rated for up to 61.2VDC. Now, we aren't exactly feeding them DC either, since our diodes are forming half-wave rectifiers. We're pulsing them at 60cps, which also is not bad practice.

SOLENOIDS:

We chose the Hi-G series "S" coils. Size "A" is only .365" x .5" x .843". This leaves ample room between units for mounting, wiring and admiring your handiwork. Larger coils would

FIGURE 4–Solenoid matrix accommodates keys of IBM Executive typewriter, shown below. Rows 3 through 6 are picked straight from central portion of keyboard, separated by dotted lines. Remaining columns of keys are moved up into rows 1 and 2, without changing order. Machine functions and one special character are on row 7. Shift is not matrixed, since it must operate simultaniously with all character keys.

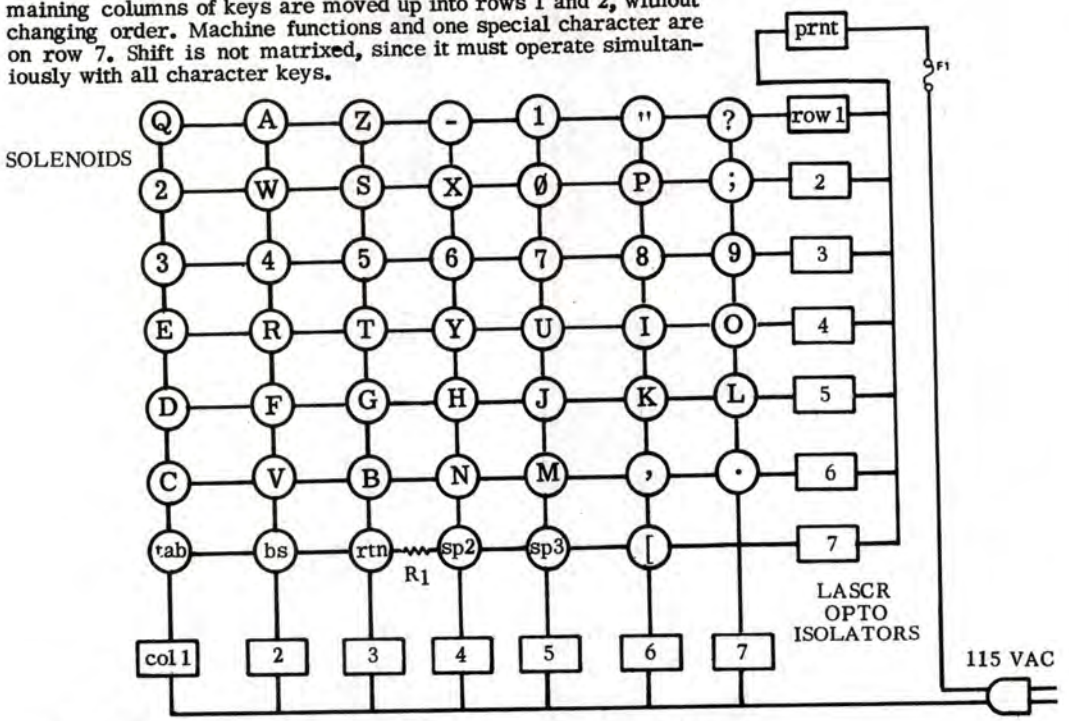
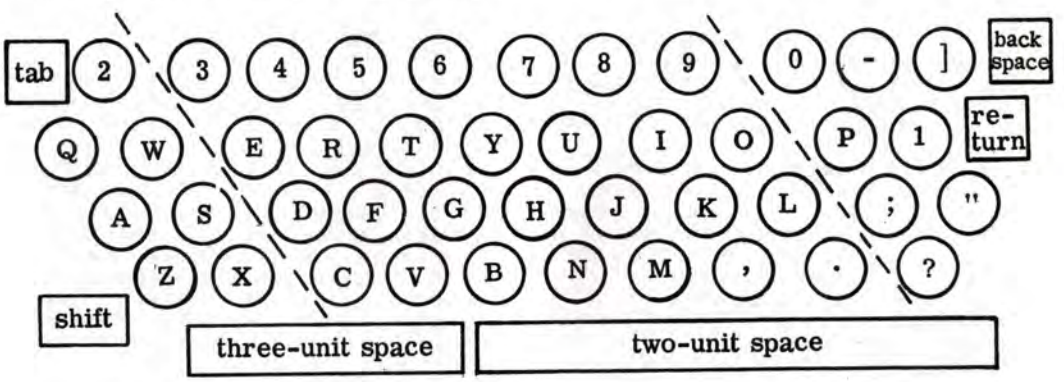


FIGURE 5–Here is IBM Executive C model keyboard, without showing alternate-case special characters.



cause difficulties.

Size "A" coils are used for all character keys, slightly-larger Size "B" for three machines: tab, back space and return. Size "C", approx. 3/4" sq. by 1 1/4" high, is used for space and shift.

Can such tiny coils wield the power that is mightier than the sword? Yes. Because even the coils over the most frequently-used letters sit idle most of the time, we can force a lot more power through them than most people would think.

By experimentation, we've found that Executive keys require about 2.5 oz of force to strike. They also must travel about 2.2". According to the manufacturer's charts, we could not get that minimum stroke and force out of Size "A" coils, but can get it from Size "B", which is not much larger. However, we do get what we want out of Size "A", probably because of voltage and pulsing, and we use them.

We bought 24V coils for all but shift and space. At 24V, the coils do not deliver anything near the power needed to depress a key. But 24V is only the continuous-duty rating. At 61.2VDC it delivers many times the force. Duty cycle must be limited to no more than 10 per cent. We can live with that. Maximum on time must not exceed 5 sec. No problem.

We are turning these solenoids on and off with Light Activated SCRs (LASCRs), little six-pin DIPs that have built-in LEDs which shine on the SCR, turning it on and off. Using these, we can control up to 25W with a 10ma TTL-level signal. No transformer is needed, yet we get 2,500V isolation from the line. But this puts 120V on our 24V coils. Can they take it? As we pointed out, the manufacturer gives his blessing to 10 per cent duty cycle and 61VDC operation. With our diodes and SCRs in the circuit, we're delivering pulsating 120V. Since each pulse is accompanied by a rest period, our coils don't mind it any more than they would 61V–but we probably get more power!

When selected, a character solenoid is pulsed for 8.33msec. every 16.66msec. Response starts within one pulse. We estimate no more than 100msec. is needed to depress the key. At the 13 character/sec. maximum rate of the Royal 2000, each character would require 76.9 msec to print. Most users would be happy with 7.5 characters/sec., a period of 133.3msec. We assume the key must be depressed less than half the print cycle, so coils need be energized somewhere between 40 and 70msec, or 5 to 7 pulses.

There are two kinds of typewriter keys: character, and machine functions. Characters are what you read, and functions make the machine do things, like space, tab, return and shift. Don't think that is just four keys, however, since there are usually at least two kinds of space, called "space" and "back space". Executives have proportional space, and offer the typist both a two-space (normal between-word spacing, also called "el" space, because it is the width of a lowercase "l", and three-space or "en" space, width of lowercase "n". On Executive, back space is a single unit of spacing: you must backspace twice for narrow letters (f,l,i,t,j, and I), three for most lowercase, all numerals and "S", four for "w" and most caps, and five for "m", "M", and "W".

All characters require about the same keying force, on IBM typebar machines about 2.5 oz, for about 2.2" travel. Machine function keys all require more force, but usually less travel. Return requires about 3 oz, space and back space 4 oz, shift 4.25 oz, and tab 5.5 oz. Travel is approximately 1.3".

In discussing this project, several computerists have asked if we have zero-voltage turn-on and off. With our use of SCRs, we do indeed have zero-voltage turnoff. If we want to have zero-voltage turn-on, we can either do it with the optional Print LASCR, figure 5, or replace that with a more-expensive solid-state relay which has that feature. To get zero-voltage turn-on, we could AND the Print signal with the unused power-line half-cycle. To do that, you must shrink 120V to TTL. Anybody have a neat and inexpensive way of doing that? How about zeniers?

That Print LASCR is optional. If implemented, you need provide it a print or strobe signal. No problem, we're using only seven I/O lines. The computer will give us that Print signal at no cost.

Zero-voltage turn-on might indeed be a worthwhile feature. Without it, we might be putting noise on the 120VAC line, and our computer might not like it.

Besides the optional Print LASCR, a mechanical breaker, also in series with the 120V supply, could provide fail-safe solenoid burnout protection.

R1 has been inserted in row 7 to make the three Size "B" coils the same impedance as the Size "A" ones. We figure about 70 Ohms, and about 1.5 W. At the durations and duty cycles we are using, 1W parts should be more than adequate, but worriers could install 2W.

PARTS:

We used the following from Magnetic Corp. Hi-G subsidiary, 96 Granby st., Bloomfield CT 06002:

42 SAA2724D1 size "A" solenoids, at $2.45 ea.
3 SBA3724D1 size "B" solenoids, at $3.89 ea.
3 SCA37120D3 size "C" 120VDC, at $4.71 ea.

From Tri-Tek, 7808 N 27th av., Phoenix AZ 85021:

16 (one optional) H74C1 LASCRs, at $2.15
16 SKT-0800 8-pin wire-wrap sockets, at $.38

From anywhere: inexpensive Vector or OK Tool wire wrap kit, 70 Ohm 1W resistor, 100 diodes, contact sement and spagetti.

CONSTRUCTION:

To make wiring easiest to understand, have seven colors of spagetti. One way of achieving that is to paint it, using a kiddie modelers' paint set. We suggest the BBROYGBVGW numbering scheme of resistors. Skip black, or zero, and use brown through violet to denote row/columns 1-7. Add black bands to ends of one set, either column or row, so you can instantly identify wires.

Board layout is easily accomplished, on Plexiglass, with grease pencil. First cut the board so that it fits the keyboard, then mark key positions. Straighten up the markings, using a "T" square, then drill holes for solenoid pushrods. We considered a drill press indespensible, but then we may be just extra fussy.

Once a Plexiglass board has been prepared, you might give some thought to transferring

its dimensions to steel. The extra weight of steel would be of advantage in giving the magnets a solid base of operation, and it could be thinner, thus gobbling up less pushrod length.

Pushrod length is going to be a problem to everyone. The solenoid people don't really give you any. You have to trim the Nylon tip, the side of it with a hole, so that your solenoids have enough motion to trip typewriter keying. Before you do that, you can simulate the cut by just not screwing the tip all-way down on the push rod.

Solenoids can be held in place with contact cement, the stuff that has held the Formica in place for 25 years on the bar at Ethel's San Luis Rey Cafe.

For software considerations, we include a table of hexadecimal and binal codes for each key. Note that it ignores Shift, since special characters move all around the keyboard, as far as standardization is concerned. Thus your software tables will be twice as long, one table for Shift and the other for Unshift.

WELL, IT'S A START:

HERE'S TYPEWRITER SOFTWARE

Here are the six-bit codes for all keys on the IBM Executive typewriter.

Shift, not shown on this table, is bit-seven. Bit eight is not needed, but Word Processing will probably use it as a strobe or print signal.

In sriting a software conversion routine, or a PROM, start by making a full ASCII listing. You'll probably be working in hexadecimal. You will find this in an ASCII/Baudot or other conversion program. This will really be just a list of hex numbers from Ø to 7F (decimal 127). You will follow that hex-order list with a list of identical length that gives the typewriter-interface code equivalent of each of the ASCII codes, in ASCII order.

ASCII starts with 32 machine codes, then 32 numbers and special characters, then at hex 4Ø "@" is followed by the capital letters. Thus "A" is the 66th ASCII character.

In the 66th line of the second 127-entry table, you will put your interface's hex code for "A".

character	binal	hex	decimal
A	001 010	12	22
B	110 011	63	99
C	110 001	61	97
D	101 001	51	81
E	100 101	41	65
F	101 010	52	82
G	101 011	53	83
H	101 100	54	84
I	100 110	46	70
J	101 101	55	85
K	101 110	56	86
L	101 111	57	87
M	110 101	65	101
N	110 100	64	100
O	100 111	47	71
P	010 110	26	38
Q	001 001	11	17
R	100 010	42	66
S	010 011	23	35
T	100 011	43	67
U	100 101	45	69
V	110 101	62	98
W	010 010	22	34
X	010 100	24	36
Y	100 100	44	68

character	binal	hex	decimal
Z	001 011	13	19
Ø	010 101	25	37
1	001 101	15	21
2	010 001	21	33
3	011 001	31	49
4	011 010	32	50
5	011 011	33	51
6	011 100	34	52
7	011 101	35	53
8	011 110	36	54
9	011 111	37	55
.	110 111	67	103
,	110 110	66	102
:	010 111	27	39
?	001 111	17	23
quote	001 110	16	22
hyphen	001 100	14	20
bracket	111 110	7	6
tab	111 001	71	113
B.S.	111 010	72	114
return	111 011	73	115
sp. 2	111 100	74	116
sp. 3	111 101	75	117
un-assigned	111 110	76	118

$49 QUALITY PRINTER, Cont.:

Speed usually not as important as print quality; systems to blame for printer's apparent slowness

Sometime, who knows when, people may start thinking rationally about printers for small systems.

Mention a printer somebody hasn't heard of, and he will reply, "How fast is it?" Printers are evaluated mostly in terms of speed, not print quality.

Selectrics, which if interfaced properly zip out about 185 words/minute, are rated "slow". A recent magazine article tells about the ideal solution: a $600-$700 20/40/80-character/line printer that spits out 40 to 80 readable characters a second (or 160 unreadable ones), all-caps, on narrow silvery paper. If used for correspondence (and it couldn't be), it could print 272 4" x 11" letters in a little over two and a half hours, on a $2.50 roll of electro-sensitive paper.

Why is this considered a sensible choice for a printer? Because if you have your computer set up as most people do, and you ask for a printout on something, you've got to sit there and wait until it finishes printing. The computer can only do one thing at a time, and unfortunately it is involved in printing.

'WATCHED POT' EFFECT

It is the "watched pot" effect. Put something on the stove and it will be burned up before you get back to look at it. Second time around, you stay with the pot, and it now takes forever even to boil.

So it is with home computer systems: tasks which require the operator to wait for completion are intolerably slow; those which either involve the operator, or allow him to do something else, are at least tolerably fast.

The average small computer system will probably not require ten pages of printing in four hours operation. Slow speed devices can meet those needs if printing does not hog the system.

Two solutions, other than the popular "fast printer" reaction, are: multitasking and multiprocessing.

Either multitasking or multiprocessing, or both, are cost effective alternatives to a faster printing device.

Multitasking, to our knowledge, is now marketed only to 6800 buffs, who have a choice of either the over-$100 TSC package, or the $55 RT/68 Mikbug-replacement ROM from Microware.

Few tasks actually tie up a small computer with work. Most of the time your computer is waiting either for input or output. Multitasking simply frees the computer to do something instead of waiting to output the next character to the printer.

Three or four years ago, when the then-superchip, 8080, was $350 a copy, multiprocessing seemed an expensive way to go. The other Sunday we were rapping with Don Smith of Jade Co., and were amazed to see he was selling something more powerful than an 8080, for $12.95. It was a 6502, or 6502A, we don't recall which. "Twelve-ninety-five!" I exclamed, recalling that $350 8080. "Yes, I pay $5 for them," Don replied.

In order to save a $5 processor, people are spending hundreds of dollars more than they should for printers, and getting print quality so bad it has to be retyped before it can be shown to anybody.

It is a case of people having solutions before they understand the problem. The problem with having to wait for a slow printer isn't the slow printer, but the system that makes you wait for the beast. When working with systems, we should look at the whole system occasionally, instead of always focusing on components.

SLOWER IS 'FASTER'

An example: a number of years back we had a Friden Flexowriter. Then, through a good deal, got a Selectric-based Dura Mach 10. We were using these machines for their punch-tape buffer, the fact that they took the typist "off line". The punch tape could be corrected, and the typing would go out-of-house in perfect shape.

Before Dura, the 100-wpm Flexowriter was "gee whiz" fast. Once the 185-wpm Dura was up, however, the Flexo was intolerably slow.

Both machines have been long gone. Recently, a Friden Justowriter was acquired. That machine prints slower than the Flexowriter (when justifying), yet seems tolerably fast. What has happened? Ye olde scribe slowing down?

No, we just don't have to wait for Justo like we did for Flexo. Flexowriter and Dura were single machines, Justowriter is paired. You have a puncher and a reproducer. Instead of waiting for the reproducer to finish printing a tape you have prepared, you can go back to the puncher and continue working. Thus a slower printer seems faster.

THE $250 LINE PRINTER

For $250, someone I know purchased a broken-down line printer that, if working, could probably run/ruin $100 worth of paper in a single evening. In the couple years I have known him, this guy has probably not written enough to fill 25 sheets of paper. In event he ever gets his computer working, which is questionable, and also gets his line printer working, which is doubtful, and gets a room big enough for the VW-sized printer, again doubtful, what makes anybody think that merely having a computer will make him prolific enough to produce more than a page or two of copy per day?

Looked at from a systems approach, we ask how much printing is to be done, and what quality is desired. If quantity is less than a half-dozen pages per hour, speed need not even be considered.

AN ASSOCIATIVE MEMORY FOR THE S-100 BUS

Sydney M. Lamb, PhD
Semionics Associates
41 Tunnel Road
Berkeley, CA 94705
(415) 548-2400

Abstract

The associative memory--also known as content-addressable memory (CAM)--allows data to be accessed by content, in contrast to conventional memory (RAM), in which data are accessed only by location. Since a CAM is logically more complex than a RAM, it is more expensive to construct, in fact so much more expensive that few CAMs have actually been built.

A simplified design has now been developed which lowers the cost of CAM by two orders of magnitude, to a level only moderately higher than that of RAM. It is called Recognition Memory, or REM for short. It is significant for the personal computing field that the first version of REM to be produced is designed for the S-100 bus.

The REM board has a capacity of 4K bytes, and it operates with either the Z-80 or the 8080. Besides the recognition capability, or content-addressability, it has two other important functions: multiwrite, the ability to write data into multiple locations with a single instruction, and bit-masking, which allows individual bits or specific bit patterns in memory to be operated upon. Adding this combination of features to an ordinary computer converts it into a very powerful machine known as a Content Addressable Parallel Processor (CAPP) (1). Among other things, a CAPP can perform parallel addition and subtraction and parallel searches for minimum and maximum values.

The REM board adds 17 instructions to the instruction set of the microprocessor for various types of recognize, multiwrite, and auxiliary operations, without any changes to the CPU. The instruction codes for these operations are contained in certain high order address bits, which are not needed as address bits for REM operations since all memory blocks of REM are accessed in parallel.

The REM boards can also be used as ordinary RAM when desired--examples of such use are for loading data into REM and for reading records which have satisfied a given set of recognition functions.

A typical CAPP configuration consists of an S-100 microcomputer with 16K or 32K of RAM (for programs, buffers, and so forth) plus 16K to 56K of REM. REM boards may be assigned to multiple memory banks for use as RAM, but all REM banks are accessed in parallel when operating in REM mode.

Applications of REM include pattern recognition, information retrieval, compiling and interpreting, natural language processing, and code compression (to increase effective capacity of disk and other storage devices including REM itself).

References

(1) Caxton Foster, Content Addressable Parallel Processors. Van Nostrand Reinhold, 1976.

THE MAJIC WAND - A COMPUTER DISPLAY IN A PEN

by
Robert A. Freedman
P.O. Box 1136
Lawrence, Mass. 01842
(617) 683-4659

Abstract:

This paper will introduce a simple, elegant and innovative way of displaying micro-computer output data to a viewer. The Majic Wand(TM) is a Retinal Display(TM), a device which projects an image directly onto the retina of the eye. It eliminates bulky and expensive CRT and display panels by writing an image directly on the retina of the eye, using a scanning technique previously unexploited for this purpose.

A pen sized Majic Wand is demonstrated as an example of the whole family of interesting display applications that can be realized using the Retinal Display technique.

A Majic Wand that can display a line of up to 100 alpha-numeric characters at a component cost of under $10, is described.

The Retinal Display is shown to be the most compact and economical computer display available to date, and naturally suitable for personal computer use. Many of the innovative variations of the display are also discussed.

Introduction:

Essential to every personal micro-computer is a means of transfering data between the computer and the user, a human interface. Typically this has meant a keyboard and a CRT video display. The CRT is a high bandwidth output and has very high redundancy. It gives you more information than you really need to interact with a computer. The CRT is also heavy, bulky, frasile, power hungry, and expensive.

For those applications where the CRT is not appropriate such as for a truely portable personal computer, the question can be asked, "What is the minimal bandwidth display hardware that will provide a visual transfer of information to the user, sufficient to support an effective dialog with a computer?" A dot matrix display of at least one line of text is probably the smallest useful display. But even that can cost too much, and require too much power using 7*5=35 diodes per character. Also, displays of this type are frequently multiplexed to save logic and power. If the diodes are not all on at the same time, why have them? Instead of illuminating all of the diodes some of the time, why not illuminate a few diodes most of the time? Thus one column of diodes mechanically scanned accross a row should perform as well as a matrix of diodes that is scanned electrically. Also, rather than continually redisplay the line of text in case the viewer might want to look at it, why not display the text only on demand? When these ideas were first tested in 1974, it was found not only that they worked, but that a surprisingly few scans were needed to read the text. Refreshing a display at a rate below the "critical" flicker frequency (25hz) is a heretical approach that actually works well. It results in a display with very small bandwidth, and minimal redundancy but makes an excellent human interface. The term Retinal Display was chosen to describe the technique of painting an image on the retina by mechanically scanning in at least one axis. The Majic Wand is only one kind of retinal display.

Definition of Retinal Display:

A Retinal Display can be defined as any device which exhibits the following characteristics:

1. The image is written directly on the retina of the eye. Light passes from the emitter, through the eye's lens, directly onto the retina, with no intermediate display screen being used. This would only scatter available light energy, reducing brilliance and efficiency.

2. The image is stored only on the retina. Whereas the liquid crystal display presents a complete image, and a CRT phosphor has a persistence of anywhere from micro-seconds to minutes, the retinal display makes use only of the eye's persistence of vision. The persistence of the retina varies inversely as a function of the amount of light entering the eye. Thus a dark adapted eye will store an image longer than one exposed to high ambient light levels.

3. The image is presented spatially as a sequence of image fragments that are accumulated over time and assembled on the retina to form the image. The image is constructed over time, rather than flashed onto the retina all at once as with a movie frame.

4. The Retinal Display consists of a minimum number of point source emitters which are mechanically scanned through the field of vision, and modulated with brightness information.

Principles of Operation:

The Majic Wand is a very elementary form of a Retinal Display. The Majic Wand consists of a linear array of Light Emitting Diodes (LED's) mounted at the end of a Wand (pen). To display a single line of text, 8 LED's can be used. As the Wand is waved through the field of vision, vertical slices thru the alpha-numeric characters are sequentially displayed as from a column oriented character generator rom. The line of text appears to float in space in front of the viewer. In most cases only one sweep is necessary to to read the message. In cases where the data is complex, or obscure, multiple sweeps may be used. Also if a very long line is being displayed, say 120 characters. One sweep can be used to read the first part of the line, and another sweep for the end of the line. Linearity of the sweep can be controlled by varying the velocity of the Wand by WRIST ACTION, interactively modifying the display parameters for optimum data transfer. Spatial registration of successive sweeps is not important as the visual system is highly adaptive and compensates so apparently distorted (non-linear), or misregistered sweeps are easily readable.

Advantages of the Majic Wand:

The Majic Wand is compact. It can fit into a pocket as would a pen. In fact, the wand could contain a pen, or vice-versa (sort of poor man's hard-copy). The wand can be unobtrusive in use. A flick of the wrist can provide a transient display of data unnoticed by surrounding persons. (Envision university exam rooms full of wrist flicking computer hackers.) Or, combined with Ascii spouting Walki-Talkis, the Majic Wand could faciliate a really Portable data terminal.

The Majic Wand uses very little power. Its duty cycle is less than 50%, and it consumes no power except when being waved, and even then, its AIR COOLED. But most importantly, the Majic Wand costs little more than the price of some Light Emitting Diodes, and some minimal interfacing.

Computer interface:

The Majic Wand can be interfaced to any micro-computer with only one eight bit parallel output port, and a one bit input port. The output port is used to drive the eight LED's. Actually, only 7 LED's are used to display 5 x 7 dot matrix characters. The input bit is used to synchronize the display of the line with the beginning of the sweep. This is accomplished by means of an inertial switch which senses the reversal of direction at the end of a sweep, and triggers the micro-computer, after a suitable delay, to display the line of text. The inertial switch consists of a diode cantilevered at one end from the wand. The other end is forced into contact with a wire, also soldered to the wand, by inertia, when the wand reverses direction. The diode was chosen because it is gold plated, has the proper mass, and costs only 5 cents.

Multi-line displays can easily be built, requiring a latched output port for each line of eight rows of dots.

Technical Considerations:

The Light Emitting Diodes used for the Wand should be the brightest available. They should be driven with the highest current consistent with the power dissipation of the diode. Since the duty cycle is low, the peak current can be several times that of the specified current, for greatest efficiency. The Wand is best viewed with a dark contrasting background. The computer program that controlles the Wand should have a time delay from the moment the sweep reversal is sensed, to allow the wand to accelerate before the next sweep.

Variations:

The basic Majic Wand consists of 8 LED's displaying one line of text when waved by hand. Many variations are possible. Multiple line displays may be built using 8 * (# lines) LED's. Graphical displays can be built with a linear array of any number of LED's. The graphical display can be either in Polar, or cartesian coordinates, depending on how it is scanned.

Rather than waving the wand by hand, various mechanical schemes may be used to scan the display. The wand may be put on a metronome device, and waved at a fixed rate. The wand may mounted on the edge of a disk or drum and spun continuously. An array of 128 LED's may be mounted on the face of a disk and used as a Plan Position Indicator (PPI) for radar.

Instead of moving the wand past the eyes of the viewer, the Retinal Display works equally with moving the viewer's eyes past the wand.

The Majic Wand uses a nominal 0.1" spacing between each LED. The size and spacing between LED's can be scaled up or down depending on what LED's are available and the application.

The Majic Wand was not practical before the advent of both micro-computers, and LED's, however, it is possible to use Strobe lights instead of LED's and make a display of incredible intensity suitable for displaying advertising from an airplane.

Other Applications:

One of the most interesting possible uses for the Majic Wand is in advertising. Imagine a pole, an Ad-pole, standing by the side of the road. If you look at the pole, all you see is a vertical column of lights. However, if you drive by the pole at highway speed, you will see spread out in bright red letters:

"DRINK DR. PEPPER" or "VOTE for HARRY" etc.

This is Dynamic Advertising. The Ad-pole takes up little room compared to standard billboard signs. It does not obscure the scenery, and is environmentally inoffensive. The only thing it can pollute is the mind of the viewer with subtle subliminal suggestions that are difficult to avoid.

The Ad-pole is economical to construct and maintain. A single chip micro-processor can be programmed to display a variety of advertising messages according to a time schedule. Thousands of them could line the highways of America each containing a micro-computer programming us.

A variation on the Ad-pole is to mount the Wand on the side of a car. Cars going in the opposite direction on a highway will see the message. With a keyboard in the car, one can create graffiti on wheels.

The ultimate implimentation of the Retinal Display is to mount some very small LED's on a pezio-electric crystal and embed it with some optics in an eyeglass frame for use as a head-up display for a truley personal computer terminal. The user would see data hanging in space in front of him-self superimposed on a portion of his normal field of view.

Conclusions:

The Retinal Display is a technique for displaying data without an intermediate display screen. The Majic Wand is a simple inexpensive implimentation of this technique which is particularly appropriate for use with personal and portable micro-computers, and is being used to pursue some of the commercial applications mentioned above. In addition, the Majic Wand is a unique and interesting novelty of itself, which can provide a great deal of interest and amusement to the personal computer owner or hobbiest.

Anyone interested in exploiting the Majic Wand for commercial purposes is advised to contact the author regarding licensing of this technology.

DOUBLE DENSITY RECORDING ON FLOPPY DISKS

A COMPARISON OF TECHNIQUES

by JEFFERSON H. HARMAN

Director of Research and Development

PerSci, Incorporated

West Los Angeles, California

Abstract

This paper is a comparison of three commonly used double density codes, MFM M^2FM and GCR. The reasons these codes are required, relative strengths and weaknesses of the codes and write precompensation are discussed.

Introduction

Man is forever trying to get something for nothing; or at least more for less. In his continuing search for more storage for less money, the computer system designer is going to double density codes on floppy disks. The use of these codes doubles the capacity of the floppy disk, almost for free. However, all methods are more susceptible to noise and density effects than the single density codes.

Mechanization of Recording

In order to understand why simply changing the code can double system capacity, we need some understanding of what happens in saturated magnetic recording. The read write head consists of a coil on a ferromagnetic core. This core has a gap of non-magnetic material which is brought into contact with the media. (Or very near in flying head hard disks.) Current in the coil magnetizes the media as it passes under the gap.

The coil responds to a change in the magnetic flux of the media. This is accomplished by moving the media. In addition to motion, a change in the magnetic field from the disk is required to produce an output. In other words, the read head can only detect where transitions in polarity occur. Therefore, the information must be encoded in the location of the boundaries, and the read system must operate by locating these boundaries. This is done by detecting the peaks of the read signal.

Self Clocking Codes

The simplest code is to write a transition for each one, and no transition for each zero. This is NRZ. This code is also a most efficient code, having only one bit per transition. Unfortunately, this code does not contain a clock. In systems where several tracks are read in parallel, schemes guaranteeing at least one transition on one of the tracks (usually also checking parity) can be used to clock the code - as in seven or nine track tape drives. This code is not used in floppy disks because only one track at a time is read.

Double frequency code; also known as FM, frequency doubling, and other aliases, consists of adding a clock to NRZ such that the data transitions are ideally centered between clocks. This code is self clocking, is easy to encode and decode, but requires two transitions per bit. This code is the most widely used single density code for floppy disks.

Density Effects

If one could write magnetic flux reversals as close together as needed, then requiring two transitions per bit would be no disadvantage. However, life isn't that simple. As transitions are packed closer together on the media, density effects begin to shift the location of the transitions. This shift interferes with the ability to distinguish clock from data, and is the limiting factor on capacity. Of course, single density floppy disks are operated at the practical maximum capacity for FM recording.

Since the floppy disk is a disk, the density changes from inner to outer tracks. Density is highest at track 76 and lowest at track 00. At high density the output level is lowest, and peak shift is greatest. At low density the output stays near zero for relatively long periods, producing inflection points that with the addition of noise could be erroneously detected as peaks. Density ranges from 6400 transitions per inch near track 76 to 3600 transitions per inch near track 00.

Peak shift has two causes. The first, due to density effects, causes peaks to shift apart. Pulses which are close tend to move away from each other when bounded by pulses which are not so close. In FM or MFM this effect can be as great as 300 ns. This effect can be compensated by precompensation. The second effect is asymmetry, which is due to poor write circuits, poor

read circuits, or poor head design. Read and write circuits can be made essentially symmetrical, but heads can be expected to produce ± 100 ns of asymmetry.

In order for a code to be more efficient it must reduce the transition rate to pass data from two transitions per bit, while at the same time providing clock information. Three codes; MFM M^2FM, and 4 for 5 GCR have been successfully used to double the capacity of floppy disk drives. This has been done at little hardware cost and with good reliability.

Modified Frequency Modulation (MFM)

MFM is the oldest of these codes. MFM adds a clock to NRZ, much like in FM. However, the clocks are only added between successive zeroes. Since the highest frequency in FM occurs when clocks are inserted between successive ones, the maximum transition rate is half that of FM for the same data rate. The minimum frequency in MFM would be half the frequency for FM at the same data rate. Therefore, doubling the data rate would not increase the transition rate over FM.

Although MFM data at 500 KHz requires no higher transition rate than FM at 250 KHz, the time window for determining if a transition is clock or data is half that of FM. The practical window is the theoretical window minus the peak shift effects and speed variation. The theoretical window for MFM is 1 µs peak to peak. Peak shift can be 600 ns peak to peak due to pulse crowding, plus 200 peak to peak due to asymmetry. This leaves 200 ns. Speed variation can take another 50 ns, another 50 ns for the ability of the phase locked loop to track the data, another 10 ns for vaiations in logic delays, another 40 ns for noise; and the window has gone to zero for MFM--although FM at half the data rate still has 1 µs to spare.

Practical MFM systems use some form of write precompensation to reduce the effect of peak shift. Practical margins of ± 100 ns are obtained in this manner. Error rates far less than 1 soft read error in 10^9 bits have been demonstrated. However, systems using MFM are more susceptible to problems from system noise, dirty heads, worn diskettes, and speed variations that FM systems.

MFM recording, in fact all of the double density codes to be considered, must be synchronized with a phased locked loop. This is because some of the clock pulses are missing. FM, of course, can be synchronized with a one shot. However, a phased locked loop gives demonstrably better data reliability.

Modified Modified Frequency Modulation (M^2FM)

In order to improve the margins of MFM it was modified; hence M^2FM. The modification was to add clock pulses between successive zeroes only if a clock was not inserted between the previous successive zeroes. In this manner a clock pulse is never crowded. However, the longest time between pulses has increased to 5 µs for a 500 KHz data rate.

Since the clock pulses are not crowded, peak shift on clock pulses will be less than on data pulses. The decoding window can be widened for data and narrowed for clock. This will increase the margin by the amount the window is skewed. Use of write precompensation distorts this. With the write precompensation used at PerSci, it was found that M^2FM worked best with a symmetrical window and worked as well but no better than MFM. The longer time between pulses would appear to make M^2FM more susceptible to noise at outside tracks, but this has not been a practical problem.

Group Controlled Recording (GCR)

The third method of doubling diskette capacity through a code is 4 for 5 GCR. This approaches the problem from an entirely different perspective.

If five bits of code are used to encode four bits of data, then those patterns containing long strings of successive zeros may be eliminated. Practical codes having no more than two successive zeros are in use on floppy disk drives and on tape. The code bits occur at 1.25 times the data rate, hence this is a 1.25 transition per bit code instead of a one transition per bit code.

The minimum time between transitions in GCR is 1.6 µs, the maximum is 4.8 µs. GCR code is not quite as bad as M^2FM for shouldering effects at track 00. Peak shift due to pulse crowding is worse than MFM or M^2FM since pulses only 1.6 µs apart are bounded by pulses 4.8 µs apart. However, the ideal window is 1.6 µs wide, so GCR is much more tolerant of peak shift. Practical windows of 400 ns have been achieved without write precompensation. With write precompensation this improves to 600 ns.

Write Precompensation

Write precompensation consists of examining data to be written and shifting the transition in time in a direction to compensate for expected peak shift. The precompensation may be made as elaborate as the controller designer desires. The more accurate the compensation the better the

timing margins.

PerSci recommends compensating six patterns of encoded MFM or M^{2}FM. Four of the patterns require 300 ns of compensation, and two of the patterns require 100 ns.

Encoding GCR simply requires grouping data into 4 bit "nibbles" and via a ROM, matrix, or whatever, translating it into corresponding 5 bit code groups. Decoding consists of translating the code groups into corresponding data nibbles. A code group, usually all ones, is reserved for synchronization.

Summary

Double density recording can be reliably accomplished with all of these techniques. The final choice of which is to be used in a system is usually decided by such factors as desirability of interchange with company X, who is using one of the methods already; availability of an off the shelf controller; availability of LSI controller chips; or such factors as management prejudice and customer preference.

Although all of the techniques work, drives and diskettes must be maintained cleaner and in better condition than with FM, that most tolerant of codes.

TYPICAL 4 FOR 5 GCR ENCODING

DATA	CODE
Sync Mark	11111
0000	11001
0001	11011
0010	10010
0011	10011
0100	11101
0101	10101
0110	10110
0111	10111
1000	11010
1001	01001
1010	01010
1011	01011
1100	11110
1101	01101
1110	01110
1111	01111

COMPARISON OF NRZ, FM, MFM, AND M^2FM

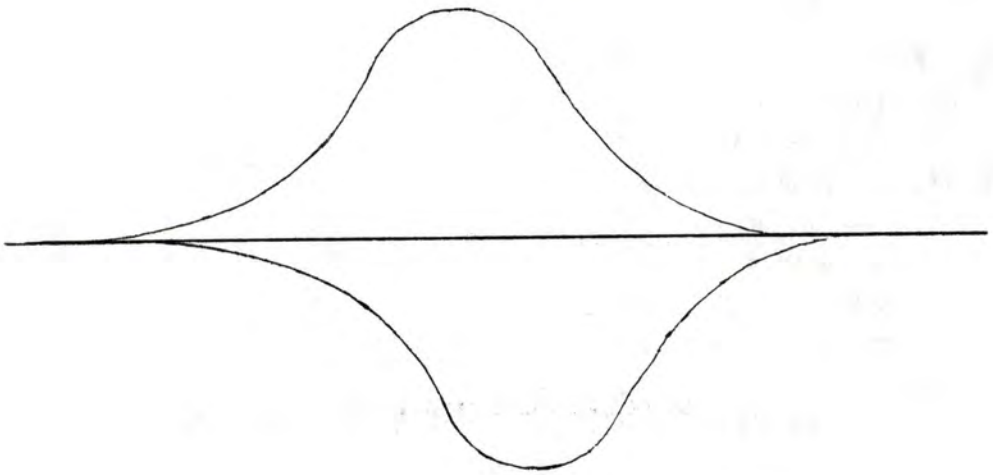
DATA		1			0			1			1			0			1								
NRZ		1						1			1						1								
FM	1	1		1			1		1	1		1	1			1		1	1		1				
MFM		1						1			1						1								
M^2FM		1						1			1						1								

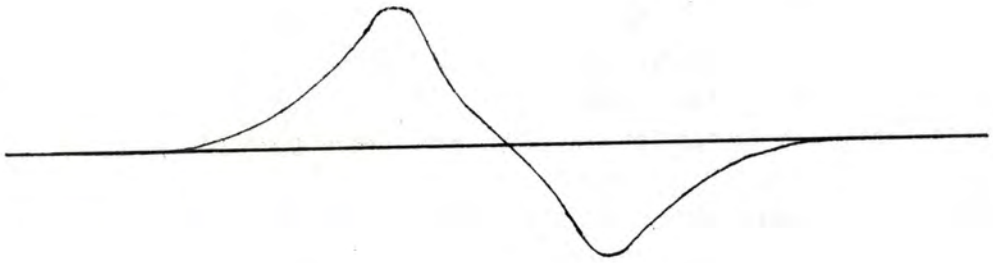
DATA		1		0		1		1		0		1		1		1		0		0		0			
MFM		1				1		1				1		1		1				1		1		1	
M2FM		1				1		1				1		1		1				1				1	
DATA		1		0		0		0		1		0		0		0		0		0		0			
MFM		1			1		1			1			1		1		1		1		1					
M^2FM		1			1					1			1				1				1					

WRITE PRECOMPENSATION

MFM or M^2FM

Precompensation Required	Encoded MFM/M^2FM Bit Being Compensated
300 nsec to right	0 0 0 1 0 1 0
300 nsec to right	0 0 0 1 0 0 1
100 nsec to right	1 0 0 1 0 1 0
300 nsec to left	0 1 0 1 0 0 0
300 nsec to left	1 0 0 1 0 0 0
100 nsec to left	0 1 0 1 0 0 1





BIBLIOGRAPHY

U. S. Patent 3,108,261 Recording and/or Reproducing System - A. Miller

U. S. Patent 3,560,947 Method and Apparatus for Communication and Storage of Binary Information - R. C. Franchini

Proposed American National Standard Recorded Magnetic Tape For Information Interchange (6250 CPI, Group Coded Recording

5 VOLT EPROMS
HOW TO USE THEM IN YOUR MICROPROCESSOR SYSTEM

Bob Greene
Application Manager
Special Products Division
Intel Corporation
3065 Bowers Avenue, Santa Clara, CA 95051

SUMMARY

This paper discusses how a family of EPROMs and ROMs can be used efficiently with microprocessor systems. The evolution of the various pinouts is discussed to provide a common frame of reference which leads directly to the impact of this family of devices on system architecture. Particular emphasis is placed on the pitfalls of bus contention and the microprocessor/memory interface. Finally, an actual printed circuit board layout which embodies all the concepts of the paper is presented.

PINOUT EVOLUTION

As EPROM/ROM technology has evolved, there are often periods of confusion over EPROM and ROM pinouts, as ROM density usually leads EPROM density by a factor of two, but ultimately users want any given EPROM to have a ROM compatible part. As we have seen, after the 2716 16K EPROM was introduced, a new ROM pinout emerged and "triumphed" over an earlier "standard". The reason this ROM pinout change occurred is that as codes stabilize in user's systems and equipment, many users opt for the less expensive ROMs, which are mask programmable devices. At the same time, users often use the highest available density ROM so they combine modular firmware and minimize device count. Of course, many users never do go to the ROM stage with their equipment, preferring to minimize inventory levels and utilize standard designs that can be customized for final equipment configurations, but they always want the capability to do so if desired.

In addition, over the past few years, the development of microprocessors has been intimately entwined with both ROMs and EPROMs.

The 1702A and its ROM counterpart, the 1302, were completely adequate to support the requirements of the 4004 series of microprocessors. In order to support the 5 volt, 3mhz 8085A and 5mhz 8086, it is desirable to use a compatible device such as the Intel 5 volt 2716, whose 450ns access time is compatible with the microprocessor requirements. Some high performance versions of these processors may require selected versions of the 2716 (such as the 2716-1 with t_{ACC} = 350ns, or the 2716-2, with t_{ACC}=390ns) depending on the actual system configuration.

Summarizing these events since the introduction of the Intel 1702A, which was the first EPROM, we can postulate the following hypothesis: at any point in time, the present EPROM determines the pinout for the next generation ROM. And, if the subsequent larger density EPROM is not ROM compatible, the ROM will change. Also, it can be seen that ROMs and EPROMs must evolve along with microprocessor developments - so memory performance does not limit system performance.

The devices which are discussed in this paper represent an extention of the 5 volt compatible family to 32K bit and 64K bit densities, while improving performance as discussed above. It also follows that the pinout for the 32K devices must be derived from the 2716 in order to maintain socket compatibility. This 16K to 32K pinout evolution is shown in Figure 1.

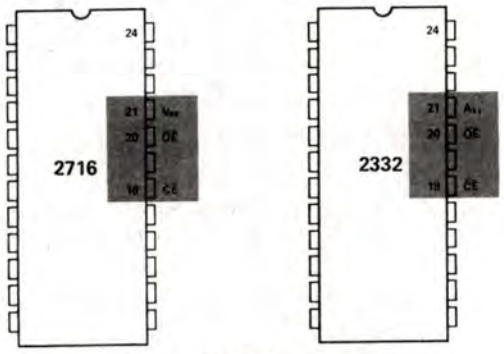


Figure 1. 16K (2716) EPROM Determines 32K (2332) ROM Pinout

SYSTEM ARCHITECTURE

As higher performance microprocessors have become available, the architecture of microprocessor systems has been evolving, again placing demands on memory. For many years, system designers have been plagued with the problem of bus contention when connecting multiple memories to a common data bus. There have been various schemes for avoiding the problem, but device manufacturers have been unable to design internal circuits that would guarantee that one memory device would be "off" the bus before another device was selected. With small memories (512x8 and 1Kx8), it has been traditional to connect all the system address lines together and utilize the difference between t_{ACC} and t_{CO} to perform a decode to select the correct device (as shown in Figure 2).

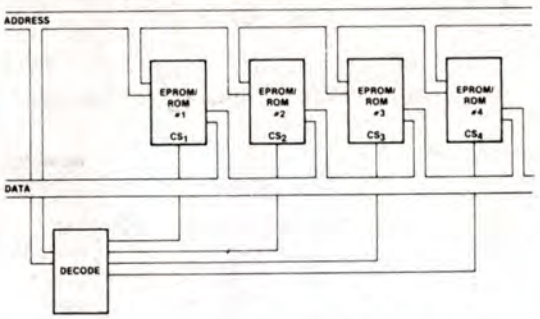


Figure 2. Single Control Line Architecture

With the 1702A, the chip select to output delay was only 100ns shorter than the address time; or to state it another way, the t_{ACC} time was 1000ns while the t_{CO} time was 900ns. The 1702A t_{ACC} performance of 1000ns was suitable for the 4004 series microprocessors, but the 8080 processor required that the corresponding numbers be reduced to t_{ACC}=450ns and t_{CO}=120ns. This allowed a substantial improvement in performance over the 4004 series of microprocessors, but placed a substantial burden on the memory. The 2708 was developed to be compatible with the 808 both in access time and power supply requirements. A portion of each 8080 machine cycle time had to be devoted to the architecture of the system decoding scheme used. This devoted portion of the machine cycle included the time required for the system controller (8224) to perform its function before the actual decode process could begin.

Let's pause here and examine the actual decode scheme that was used so we can understand how the control functions that a memory device requires are related to system architecture.

The 2708 can be used to illustrate the problem of having a single control line. The 2708 has only one read control function, chip select ($\overline{CS}$), which is very fast (t_{CO}=120ns) with respect to the overall access time (t_{ACC}=450ns) of the 2708. It is this time difference (330ns) that is used to perform the decode function, as illustrated in

Figure 3. The scheme works well and does not limit system performance, but it does lead to the possiblity of bus contention.

BUS CONTENTION

There are actually two problems with the scheme described in the previous section. First, if one device in a multiple memory system has a relatively long deselect time, and a relatively fast decoder is used, it would be possible to have another device selected at the same time. If the two devices thus selected were reading opposite data; that is, device number reading a HIGH and device number two reading a LOW, the output transistors of the two memory devices would effectively produce a short circuit, as Figure 4 illustrates. In this case, the current path is from V_{CC} on device number one to GND on device number two. This current is limited only by the "on" impedance of the MOS output transistors and can reach levels in excess of 200mA per device. If the MOS transistors have a lot of "extra" margin, the current is usually not destructive; however, an instantaneous load of 400mA can produce "glitches" on the V_{CC} supply-glitches large enough to cause standard TTL devices to drop bits or otherwise malfunction, thus causing incorrect address decode or generation.

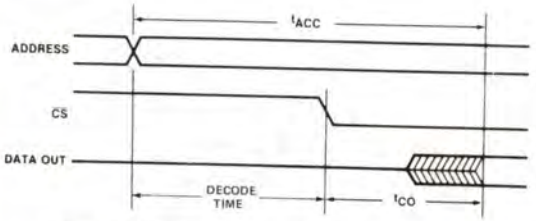


Figure 3. Single Line Control Architecture

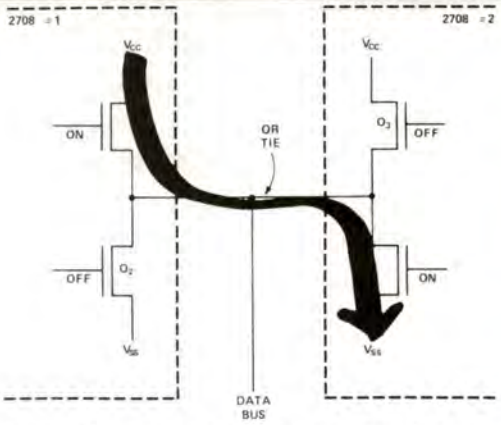


Figure 4. Results of Improper Timing when OR Tying Multiple Memories

The second problem with a single control line scheme is more subtle. As previously mentioned, there is only one control function available on the 2708 and any decoding scheme must use it out of necessity. In addition, any inadvertent changes in the state of the high order address lines that are inputs to the decoder will cause a change in the device that is selected. The results is the same as before-bus contention, only from a different source. The deselected device cannot get "off" the bus before the selected one is "on" the bus as the addresses rapidly change state. One approach to solving this problem would be to design (and specify as a maximum) devices with t_{DF} time less than t_{CO} time, thereby assuring that if one device is selected while another is simultaneously being deselected there would be some small (20ns) margin. Even with this solution, the user would not be protected from devices which have very fast t_{CO} times (t_{CO} is specified as a maximum).

The only sure solution appears to be the use of an external bus driver/transceiver that has an independent enable function. Then that function, not the "device selecting function," or addresses, could control the flow of data "on" or "off" the bus, and any contention problems would be confined to a particular card or area of a large card. In fact, many systems are implemented that way - the use of bus drivers is not at all uncommon in

large systems where the drive requirements of long, highly capacitive interconnecting lines must be taken into consideration - it also may be the reason why more system designers were not aware of the bus contention problem until they took a previously large (multicard) system and, using an advanced microprocessor and higher density memory devices, combined them all on one card, thereby eliminating the requirement for the bus drivers, but experiencing the problem of bus contention as described above.

THE MICROPROCESSOR/MEMORY INTERFACE

From the foregoing discussion, it becomes clear that some new concepts, both with regard to architecture and performance are required. A new generation of two control line EPROM devices is called for with general requirements as listed below:

1. Complete ROM and pin and function compatibility.
2. A power control function that allows the device to enter a low-power standby mode when deselected. This function can be used as the primary device selecting function, independant out of the output control.
3. Capability to control the data "on" and "off" the system bus, independent of the device selecting function identified above.
4. Access time compatible with the high performance microprocessors that are currently available.

Now let's examine the system architecture that is required to implement the two line control and prevent bus contention. This is shown in the form of a timing diagram (Figure 6). As before, addresses are used to generate the unique device selecting function, but a separate and independent Output Enable ($\overline{OE}$) control is now used to gate data "on" and "off" the system data bus. With this scheme, bus contention is completely eliminated as the processor determines the time during which data must be present on the bus and then releases the bus by way of the Output Enable line, thus freeing the bus for use by other devices, either memories or peripheral devices. This type of architecture can be easily accomplished if the memory devices have two control functions, and the system is implemented according to the block diagram shown in Figure 6. It differs from the previous block diagram (shown in Figure 2) in that the control bus, which is connected to all memory Output Enable pins, provides separate and independent control over the data bus. In this way, the microprocessor is always in control of the system; while in the previous system, the microprocessor passed control to the particular memory device and then waited for data to become available. Another way to look at it is, with a single control line the microprocessor/memory communications. By using two control lines, the memory is synchronized to the processor.

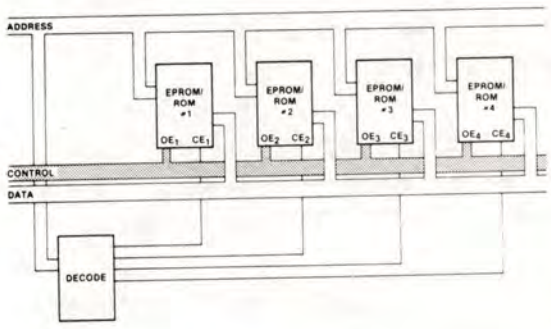


Figure 6. Two Control Line Architecture

TERMINOLOGY

Some of the terminology applied to the functions of the Intel 5 volt compatible family may be confusing or unfamiliar to many EPROM/ROM users, so the various terms are defined here. Actually, the nomenclature was developed by various standards groups and is reiterated here to avoid confusion as we begin a detailed discussion of the devices themselves.

First of all, Chip Enable ($\overline{CE}$) must be defined, as it is the primary device selection pin. By agreed standards, that function which substantially affects power dissipation is called CE. Any memory device that has a CE function has both an active and standby power level associated with it.

Output Enable (OE) is the signal that controls the output. The fundamental purpose of OE is to provide a completely separate means of controlling the output buffer of the memory device, thereby eliminating bus contention.

THE NEW INTEL FAMILY

Figure 7 shows the new Intel 5 volt compatible family of EPROMs and ROMs. In order to take advantage of the modular compatibility offered by the family, the functional compatibility of device pins 18, 19 and 21 must be understood. (Shaded area in Figure 7.)

First, we must examine the compatibility of the two oldest EPROM members of the 5 volt family - the 8K (2758) and the 16K (2716).

Pin 21 (V_{PP}) is normally connected to V_{CC} for ready only applications of both devices, and pin 19 is either at GND (V_{IL}) for the 8K 2758 or connected to A_{10} for the 16K 2716. Further details on either of these devices can be found in Section 9 of the 1977 Edition of the Intel Memory Design Handbook, or Section 4 of the 1978 Intel Data Catalog.

The 32K (4Kx8) devices, which have identical pinouts for both the ROM and EPROM, will now be discussed. Pin 18 is $\overline{CE}$. Pin 19 is A_{10}, while pin 20 is $\overline{OE}$. As was pointed out before, Output Enable is the function which allows independent control of the data "on" and "off" the output bus. As Figure 7 indicates, V_{PP} (the programming voltage for the 2732 EPROM) is now multiplexed with $\overline{OE}$ on pin 20. Pin 21 becomes A_{11}, which is the additional address bit that is required as the density increases from 16K to 32K.

Pin 21 is the only pin that requires any special consideration when designing a system to accept the 8K, the 16K, or the 32K device. With the 8K and the 16K devices, pin 21 must be connected to V_{CC}, while with the 32K and higher density devices, it must be connected to A_{11}. This is easily accomplished by making sure the printed circuit trace links all pin 21s together as though they were an address line and allowing for a jumper that will connect pin 21 to either V_{CC} or A_{11} at the edge of the array (this technique can be seen in the "Printed Circuit Board Design" section and (Figure 8). Connecting the pin 21s together in this manner is acceptable as the read current requirement for V_{PP} is 4mA maximum per device - low enough to be handled by a signal trace, but too high for an address drive to provide directly.

The highest density member of the family is a 64K ROM which is also shown in Figure 7. In order to maintain total compatibility, it is packaged in a standard 28-pin package.

It may seem as though the 28 pin package is not compatible with the rest of the family, but referring again to Figure 7, note that the lower 24 pins are identical

to the 24 pin 8K, 16K and 32K devices. To allow for total compatibility within the family, printed circuit boards must be laid out to accommodate 28 pin sites; a jumper must be included to accommodate pin 21 as shown in Figure 8, and when using 64K devices, CS_2 (Pin 26) must be mask coded active high. This compatibility can also be seen graphically in Figure 9. The upper portion of the figure shows how 24 pin devices are used in the 28 pin site. The two control lines ($\overline{CE}$ and $\overline{OE}$) remain unchanged as discussed earlier, and A_{12}, the next address bit required for a 64K bit device, is connected to pin 2 of the 28 pin site. The lower portion of the figure illustrates the use of 28 pin devices. Address bit A_{12} is already connected to the right pin, and the chip selects (CS_1 and CS_2) are connected to the V_{CC} power distribution grid. This configuration would require that both CS_1 and CS_2 be coded active high.

In some systems, additional logic may be used to implement a "special" decode of CS_1 to allow for ROM to overlap RAM when a system bootstrap program is being loaded; that additional logic should be implemented with CS_1; CS_2 should be coded active high in order to preserve total compatibility.

To summarize, the selection of a 28 pin package for 64K devices has several benefits of importance to present and future system designs:

1. Two line control philosophy (separate $\overline{CE}$ and $\overline{OE}$ functions) is preserved at the 64K level.
2. 64K EPROM compatibility is allowed for by maintaining a pin for the V_{PP} function.
3. The next generation (128K bit ROM) must be in a 28 pin package.

If CS_2 (pin 26) is mask coded to be active high and connected to V_{CC}, and the jumper provision for pin 21 is included on the card as described above, any member of the family can be plugged into the same socket - 1K, 2K, 4K or 8K bytes - without any card modification or redesign. In additional, future devices of higher density will fit in the same pinout.

PRINTED CIRCUIT BOARD DESIGN

The I_{CC} waveform for the 2332 and the 2364 is shown in Figure 10. The supply current, I_{CC}, has three segments that are of concern to the system designer - the standby level, active level and the transient peaks that are produced on the rising and falling edges of Chip Enable. The transient currents must be suppressed by properly selected decoupling capacitors. High quality, high frequency ceramic capacitors of small physical size with low inherent inductance should be used. In addition, bulk decoupling must be provided, usually near where the power supply is connected to the array. The purpose of the bulk decoupling is to overcome the voltage droop caused by the inductive effects of the PC board traces. Eletrolytic or tantlum capacitors are suitable for bulk decoupling. The following capacitance values and locations are recommended for the 2332 and 2364:

1. A 0.1uF ceramic capacitor between V_{CC} and GND at every other device.
2. A 4.7uF electrolytic capacitor between V_{CC} and GND for each eight devices.

A printed circuit board layout for a total array of 16 devices is shown in Figure 11. This printed circuit layout incorporates a power supply distribution system such that the power supply and ground traces on the PC board are gridded both

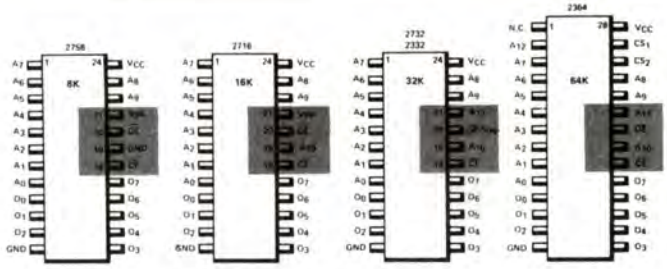
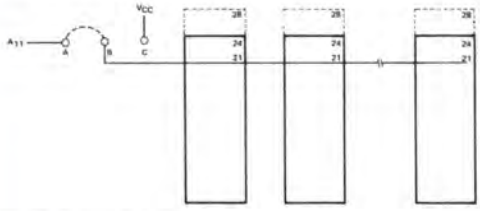
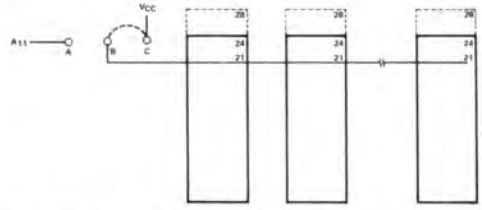


Figure 7. 5 Volt EPROM/ROM Compatible Family



32K Bit Density and Higher



8K and 16K Density Devices

Figure 8. Pin 21 Connections for Various Density Devices

vertically and horizontally at each memory device; this technique minimizes the power distribution system impedance and enhances the effect of the decoupling capacitors. Provisions are included for all address inputs, output enable inputs, data outputs and decoded chip enable inputs. The 0.1uF capacitors referred to above are included for every other device (indicated by the legend C2) while the bulk decoupling capacitor is shown at the upper left-hand corner (indicated by the legend C1). The layout consists of four rows of four 28 pin device sites each and embodies all of the concepts explained above. Note that pins 28, 27, and 26 are all connected to V_{CC}. This requires that when ordering mask programmed 2364 64K ROM, the order must specify that CS_1 and CS_2 be coded active HIGH. The single jumper provision discussed in the previous section is also included at the upper lefthand corner of the array (indicated by A, B, and C). Pad B is connected to Pin 21 of all devices in the array; pad A should be connected to the A_{11} address driver and pad C is connected to V_{CC}. For use with 32K bit or larger devices, a jumper must be installed between pads A and B; for use with the 2716 (16K) or the 2758 (8K), the jumper must be installed between Pads B and C.

A full size (2x) artwork film is included on the last page of this paper. The entire array, or segments of it can be photographed and used directly as part of a system board.

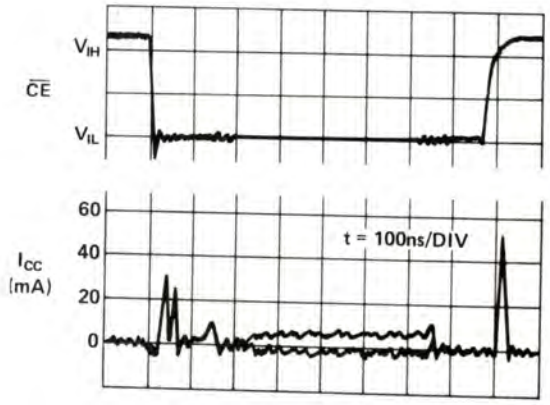
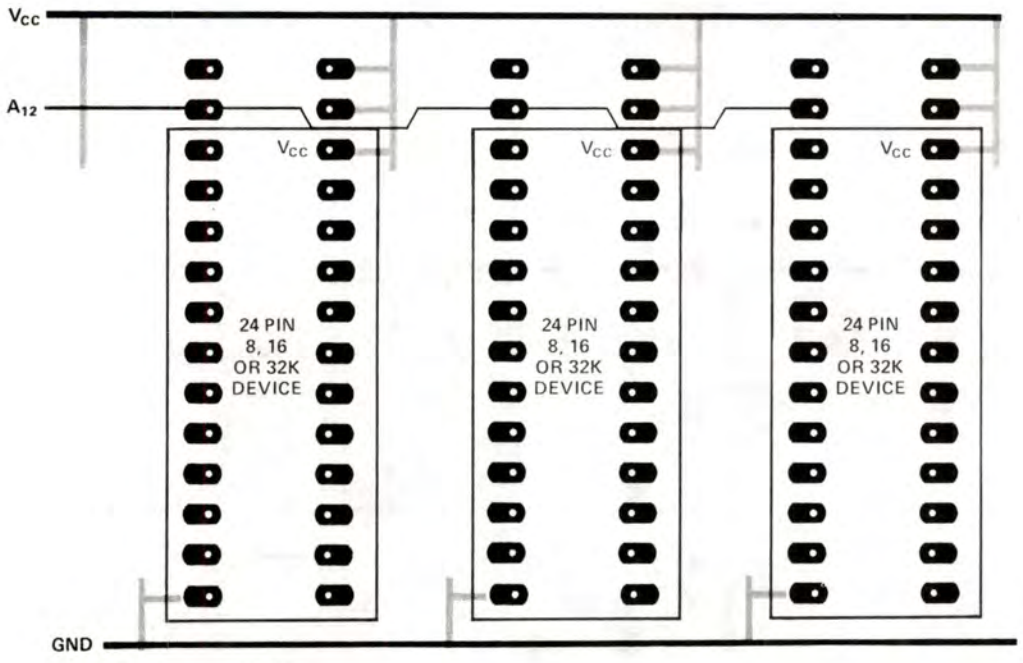
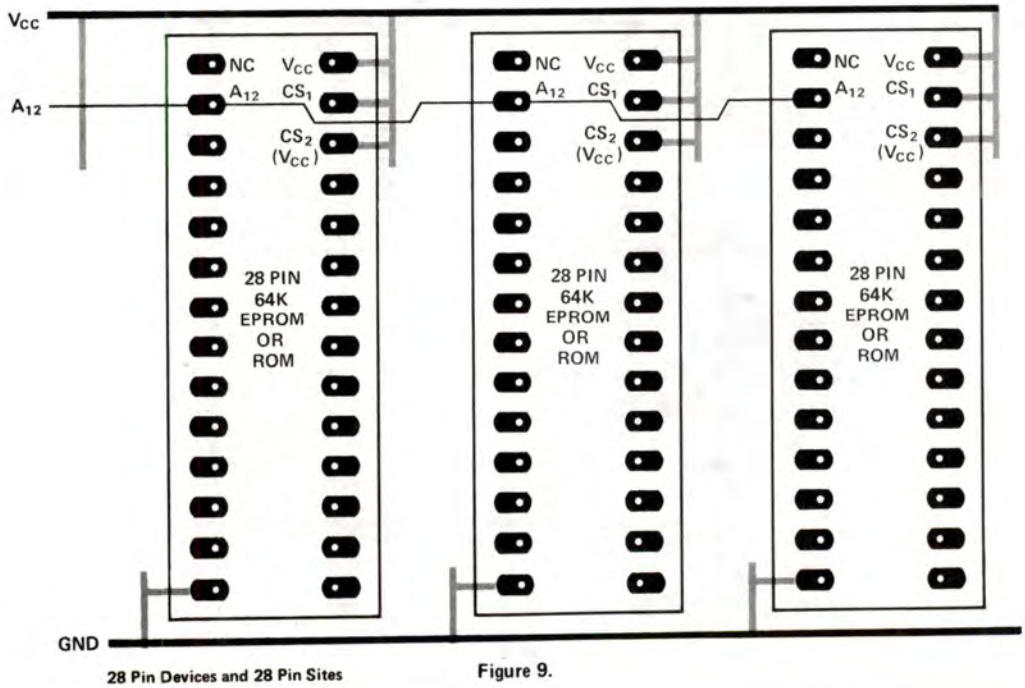


Figure 10. Typical I_{CC} Current vs Time



24 Pin Devices and 28 Pin Sites



28 Pin Devices and 28 Pin Sites

Figure 9.

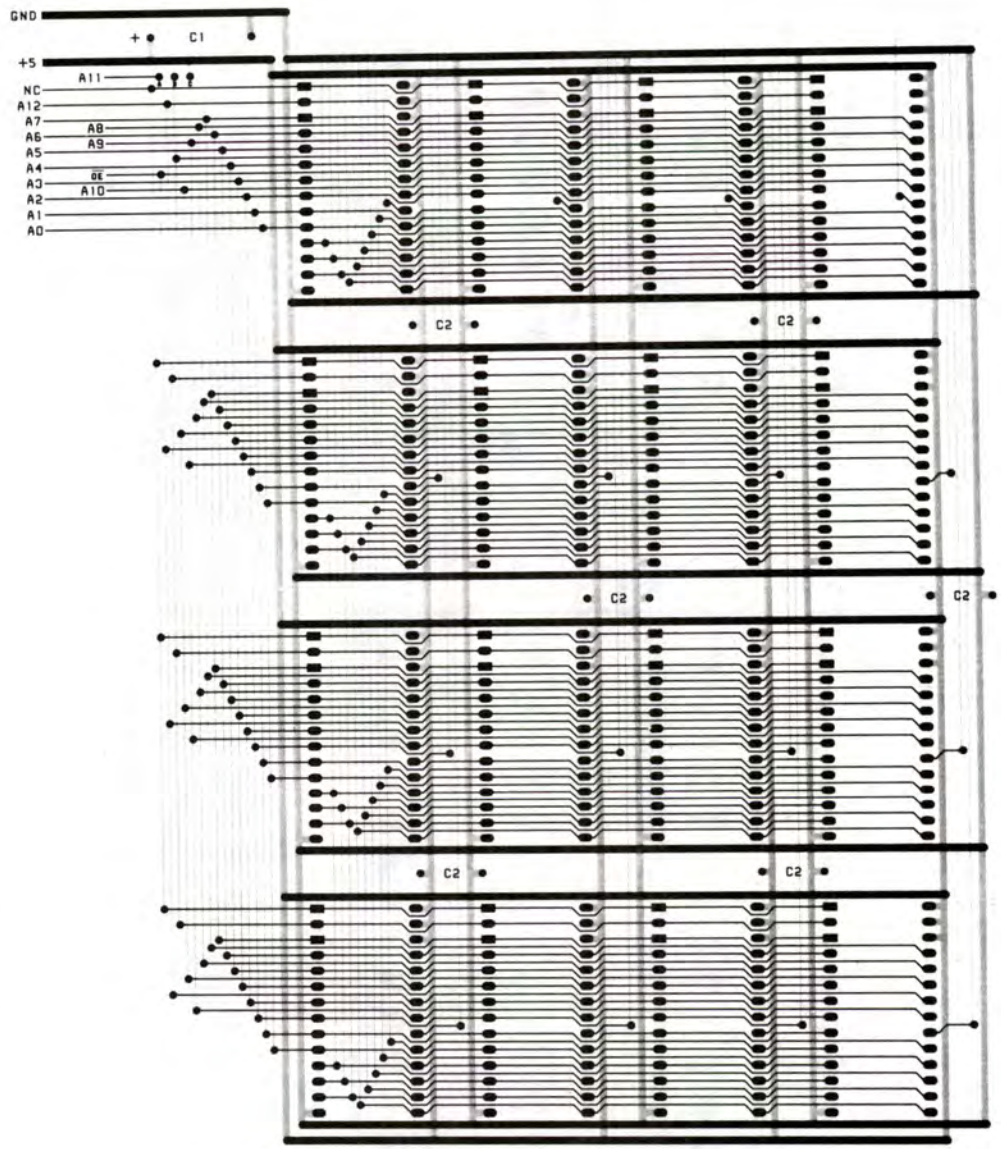

GND
+5
C1
A11
NC
A12
A7
A6
A5
A4
A3
A2
A1
A0
A8
A9
OE
A10
C2
C2
C2
C2
C2
C2

Figure 11.

September 20, 1978

A CBBS MANUAL

Dave Caulkins

Introduction

Early in 1978 two CACHE members (Ward Christensen and Rand Suess) developed the Community Bulletin Board System (CBBS). The CBBS is an electronic mail system; users can enter messages, read previously entered messages, review messages summaries, etc. Anyone equipped with a 110 or 300 baud modem attached to a terminal or computer can access a CBBS using an ordinary voice telephone line. The CBBS provides extensive HELP messages and prompts for the inexperienced user, but even with these aids it is easy to miss useful features or forget how to do things. This manual is entended to provide a comprehensive hardcopy reference to CBBS functions.
At present (August of 1978) there are four CBBS in operation:

City	Phone Number	Operating days/hours
Atlanta, GA	404-458-4886	7 days/week, evenings
Chicago, Ill	312-528-7141	7 days/week, 24 Hrs/day
San Diego, CA	714-565-0961	7 days/week, 24 Hrs/day
Santa Clara, CA	408-246-2805	7 days/week, 24 Hrs/day

This manual is intended for use with the Santa Clara CBBS; features of other CBBSs may differ from those described here. The best way to learn about CBBS operation is to try one; call one of the above numbers with any 110 or 300 baud terminal or computer. When the connection is made send several CARRIAGE RETURN characters; from these the CBBS will figure out character rate and start transmitting the welcome message.

All CBBS message reference functions (Enter, Retrieve, Summary, Etc.) use the message number assigned to each message on entry. The message number is separated from the function character by ";". This delimiter character can be used to form strings of command functions, suppressing HELP messages and speeding up CBBS functioning.

Top Level Functions

The CBBS has a series of functions (or commands); each invoked by typing a single alphabetic character. They divide into three general classes;

1. Introductory Information And System Parameter Setting
   B - Print bulletins (a bulletin is a general message printed automatically upon accessing the system). The B function allows repeating the bulletins in case information was missed when they were printed on system entry.
   C - Case Switch. Charges the CBBS from the default value of uppercase-only to upper/lower case. Successive entries of C will cause the case switch to alternate between the two values. The other switches discussed below (D and P) operate similarly.
   D - Duplex Switch. Changes the CBBS from the default value of full duplex (CBBS echoes each received character) to half duplex (CBBS transmits only the characters it originates).
   G - Goodbye. Exits the user from the CBBS and terminated the telephone connection.
   H - Help. Provides detailed information about CBBS functions.
   N - Nulls. Inserts 1-9 nulls after each carriage return to allow for mechanical devices which may have slow carriage return operation.
   P - Bell Switch. Changes the CBBS from the default value of transmitting the conrol-G character (BEL) to the value for not transmitting it.

W - Welcome Message. The welcome message is the first thing printed after system entry. The W function allows repeating the welcome message in case information was missed on first presentation.
X - Expert Mode. The prompt messages are shortened, speeding up system operation.

2. Message Entry and Retrieval
E - Enter A Message. This function assigns the next sequential message number and guides the user through the steps necessary for entering a 60 character-per-line message with a maximum of 16 lines. The user is asked for the date, the name of the person the message is for, the subject and finally the text of the message. This function is covered is covered in more detail below.
K - Kill a Message. This function allows the user to erase messages that are no longer current.
R - Retrieve A Message. Upon receipt of the message number the CBBS transmits to the user the entire text of the message.

3. Message Summaries (covered in more detail below)
S - Summarize Messages. Starting with the message number given, the CBBS transmits the message number, date, sender, recipient and subject of all messages with numbers equal to or larger than the given message number. As discussed below the S function can also be used to initiate string searches of the date, to, from, or subject fields of messages.
Q - Quick Summary. Starting with the message number given, the CBBS transmits the message number and subject of all messages with numbers equal to or larger than the given message number.

Control Characters

The CBBS accepts a number of control characters that can be used at any time to alter CBBS operation. These are:

DEL/RUBOUT - Erases (and echoes) the last character sent to the CBBS.
CTL-C - Cancels the current printing. This may be used to abort the printing of unwanted bulletins, etc.
CTL-K - 'Kills' the current function and returns to the menu of top level functions.
CTL-N - Sends 5 nulls after each carriage return to accomodate devices with slow carriage return.
CTL-R - Transmits to the user the characters entered after the previous carriage return. This can be done as many times as desired during the printing of a line, before or after DEL has been used. It is particularly useful when a noisy telephone line is encountered.
CTL-S - Stops (or restarts) CBBS transmission of characters. Useful with video terminals to 'freeze' a screen full of data for leisurely examination.
CTL-U - Erases the current input line.

Complex Functions

The Enter and Summary functions are rather more complex than other CBBS functions. I will cover them in some detail.
Enter:
When the E is sent to the CBBS to intiate the Enter function the CBBS responds;
WHAT IS THE DATE, PLEASE (MM/DD/YY)?
The user sends the date.
IF THIS MESSAGE IF FOR EVERYONE, ENTER: ALL
WHO IS THIS MESSAGE TO (20 Chars Max.)?
The user sends 'all'; or the name of the recipient.
MESSAGE SUBJECT (30 chars max)?
The user sends the subject of the message.
WHAT TO BE ABLE TO ERASE THIS MSG LATER?
If the user wants to be able to erase this message he sends 'Y' otherwise 'N'. It is almost always good practice to send the 'Y'

ENTER 4 CHAR PASSWORD (C/R FOR NONE)?
If the user wishes to prevent erasure of the message by anyone except himself or the system operators, he enters a 4 character password which will be required subsequently to erase the message via the 'K' function. If anyone can erase the message, he sends a carriage return. Use of the carriage return provides a convenient 'return receipt' mode of operation: put the phrase 'kill this when you've read it' in your message; when the message vanishes you know it has been read and killed by the recipient.
Y/N: REVIEW STEPS TO ENTER A MESSAGE:?
If 'Y' is sent by the user HELP information about the Enter function is transmitted to the user. If 'N' is sent the CBBS continues:
ENTER THE LINES. 60 CHAR/LINE MAX. BELL RINGS AT 55
-----WHEN DONE, ENTER A BLANK LINE (C/R ALONE)
-----TO ABORT, ENTER ABORT
01?
After each line number (automatically generated by the CBBS) to user sends up to 60 characters of the message. At any point CTRL-R can be used to cause the CBBS to transmit back to the user the contents of the line entered thus far.
Similarly DEL/RUBOUT echoes the previous character sent by the user and removes it. A carriage return terminates the line, causing the CBBS to generate the next line number and wait for the text of the next line.
The message can consist of up to 16 lines.
To terminate message entry, type a carriage return in response to the line number transmitted by the CBBS. It will respond with:
ENTER LETTER FOR FUNCTION:
A=ABORT, C=CONTINUE INPUT, E=EDIT, H=HELP, L=LIST, R=RETYPE LINE, S=SAVE---
(DO THIS WHEN YOU HAVE FINISHED) A, C, E, H, L, R, S:?
These sub-functions operate as follows:

ABORT - Allows entry of a new CBBS function; message text is erased.
CONTINUE - Returns to message entry mode, starting with the next line number.
EDIT - Permits string replacement in the message text.
HELP - Transmits help information.
LIST - Lists the message text from the indicated line number. To list the whole message, send L;1.
RETYPE LINE - Retype the indicated line.
SAVE - Stores the message on the disk; when this is done allows entry of a new CBBS function.

Summary;
A summary of messages in the CBBS can be gotten by sending 'S;' followed by the message number where message summaries are to start. The system will transmit the message number, size (no. of lines), Date, From, To, and Subject fields of each message from the selected message number to the most recent message.
A quick summary can be gotten by sending 'Q;' followed by the starting message number. In this case only the message number and subject fields are printed.
Summaries can also be retrieved selectively on the basis of string searches of the Date, From, To or Subject fields.
String Search Examples:
S;2,D=2/13/78
Any messages subsequent to 1 entered on 2/13/78
S;49,F=Ward
Any messages subsequent to 48 from someone whose name contains the string 'ward'
S;13,T=Suess
Any messages subsequent to 12 to someone whose name contains the string 'suess'
S;23,S=CBBS
Any messages subsequent to 22 with the string 'CBBS' in their subjects.

APPENDIX A

CBBS Function Characters
Alphabetic Listing

Char	Top level (Y/N)	Function	Sub-Function	Description
A	N	Enter	Abort	After entry of a message, the A sub-function causes entry of the message to be aborted.
B	Y	Bulletin	-	Prints the bulletins.
C	Y	Case switch	-	Changes from upper case only to upper/lower case, or changes back again.
C	N	Enter	Continue	After message entry has been concluded allows entry of more lines.
CTL-C	-	Any	-	As with all control chars, can be invoked at any time. Cancels current printing.
DEL	-	Any	-	Erases last char sent to CBBS and echoes it.
D	Y	Duplex switch	-	Changes from full duplex mode to half duplex, or changes back again.
D	N	Summary	Date search	Performs a string search on the DATE field of message.
E	Y	Enter	-	Enter a message.
E	N	Enter	Edit	After entry of a message the E sub-function can be used to replace old strings with new ones.
F	N	Summary	FROM search	Performs a string search on the FROM field of messages.
G	Y	Goodbye	-	Exits from CBBS, accepts comments,terminates phone connection.
H	Y	Help	-	Provides detailed information on CBBS functions.
K	Y	Kill message	-	Erases a previously entered message. If a password was used on msg. entry it will be required to erase the msg.
CTL-K	-	Any	-	'Kills' current function and returns to 'menu' of top level functions.
L	N	Enter	List	Lists the lines of the entered message, starting with the given line number.

Char	Top level (Y/N)	Function	Sub-Function	Description
N	Y	Nulls	-	Inserts 0-9 nulls after carriage return to accomodate devices with slow carriage return.
CTL-N	-	Any	-	Similar to N; inserts 5 nulls after CR.
P	Y	-	Bell switch	Enables (or disables) transmission of BEL character (CTRL-G).
Q	Y	-	Quick summary	Prints the subject field only for all messages with numbers equal to or larger than the given message number.
R	Y	-	Retrieve message	Retrieves and transmits the entire text associated with the given message number.
R	N	Enter	Retype lines	Retypes the indicated line of the entered message.
CTL-R	-	Any	-	Transmits to the user the characters entered after the previous carriage return.
S	Y	Summary	-	Starting with the given message #, the CBBS transmits the message number, date, sender, recipient and subject of all subsequent messages.
S	N	Summary	Subject search	Preforms a string search on the SUBJECT field of messages.
CTL-S	-	Any	-	Stop/start CBBS output.
T	N	Summary	To search	Preforms a string search on the TO fields of messages.
CTL-U	-	Any	-	Erase current input line.
W	Y	Welcome	-	Transmits the welcome message.
X	Y	Expert	-	Invokes expert user mode; prompts are shortened, speeding up system operation.

APPENDIX B
An actual CBBS session

#1 TERMINAL NEED NULLS? TYPE CTL-N WHILE THIS TYPES:

*** WELCOME TO THE SANTA CLARA CO. PCNET ***

*** COMPUTERIZED BULLETIN BOARD SYSTEM ***

CONTROL CHARACTERS ACCEPTED BY THIS SYSTEM:

DEL/RUBOUT	ERASES LAST CHAR. TYPED (AND ECHOS IT)
CTL-C	CANCEL CURRENT PRINTING INCLUDING THIS WELCOME MSG AND THE BULLETINS THAT FOLLOW.
CTL-K	'KILLS' CURRENT FUNCTION, RETURNS TO MENU
CTL-N	SEND 5 NULLS AFTER CR/LF
CTL-R	RETYPES CURRENT INPUT LINE (AFTER DEL)
CTL-S	STOP/START OUTPUT (FOR VIDEO TERMINAL)
CTL-U	ERASE CURRENT INPUT LINE

CBBS SYSTEMS IN USA

CBBS	SAN DIEGO	714-565-0961
CBBS	SANTA CLARA	408-246-2805
CBBS	CHICAGO	312-528-7141
CBBS	ATLANTA	404-458-4886

NEW REV 4 D. C HAYES ON LINE 9-SEP-78. PLEASE LEAVE COMMENTS

MESSAGES PACKED 10-SEP-78 MSG 443 BECAME MSG 299

A NEW CBBS, CBBS SAN DIEGO, BEGINS 24HR OPERATION 27-AUG-78
PHONE 714-565-0961

THIS CBBS WAS CREATED BY WARD CHRISTENSEN
AND RANDY SUESS OF CACHE. THEY ARE OPERATING
THE FIRST CBBS IN CHICAGO: ITS NUMBER IS
312-528-7141.
THE EQUIPMENT FOR THIS SANTA CLARA CBBS WAS
DONATED BY JIM WARREN & BILL GODBOUT.

Y/N: IS THIS YOUR FIRST TIME ON THE SYSTEM?N
WHAT IS YOUR FIRST NAME?DAVE
WHAT IS YOUR LAST NAME?CAULKINS

HELLO DAVE CAULKINS
HAVE I MISSPELLED YOUR NAME?N

LOGGING NAME TO DISK...
NEXT MSG # WILL BE 336
YOU ARE CALLER # 869

FUNCTION: B,C,D,E,G,H,K,N,P,Q,S,W, (OR ? IF NOT KNOWN)
?C
LOWER/UPPER CASE MODE

FUNCTION: B,C,D,E,G,H,K,N,P,Q,R,S,W, (OR ? IF NOT KNOWN)

```
?h
USE CTL-K TO ABORT THE HELP FUNCTION
HELP FILE last updated 8/14/78
Want help in using the system's control characters?y
The control chars DEL/RUBOUT control-R, and control-U are
used when keying in a line.  Control characters CTL-C, K,
N, and S are used when the sytem is typing something to you.
Want the DEL/RUBOUT, CTL-R & CTL-U explanation?y
     DEL (sometimes labled RUBOUT) is used to backspace when
typing, erasing the characters you typed.  The system echos the
characters back to you as it deletes them.
     Control-R retypes the input line back to you.  Use it
when you want to see what the line you are typing looks like,
after having used DEL.
     Control-U cancels the entire line being typed, allowing
you to start it over.
Want the CTL-C, K, N, and S explanation?y
Use these when the system is typing something to you:

     Control-C cancels what is being sent.  Output resumes
at the next information typed, or the next question.  Use
it when you know what the system is going to say/ask, or
otherwise don't want to see more of what is being typed.
     Control-K kills the current function.  It returns you
to the function select menu.  It may be typed alone as the
first character on a line in input mode.  If allowed, it will
echo as:  K  and you must press return to have it take
effect.  If you sent  K when you don't want it, cancel it
with a control-N which sets a flag in the computer telling it to
send 5 null characters at the beginning of each line output.
     Control-S suspends output until some other character
is typed.  Use this if you have a video display, and want
to see something before it scrolls off the top of the screen.
Want help on the E (ENTER) function?y

The E (Enter) function is used to enter messages into the
system.  A message is any information to be conveyed to
1 or more people.  You may leave messages for a specific
person, or for seneral use.  It consists of up to 16 lines,
each line up to 60 characters.  You may assign a password
to be able to erase your own messages.  Using the Enter
function, you key in your message just as if you were
simply typing it at a typewriter.  The system will put
line numbers at the front of each line (like a BASIC program).
After the message has been keyed in, you can edit it (by
retyping any line), list it, save it, start over, or throw
it away.
Want info on erasing your own messages?y
When you enter your message, you will be asked if you want
to be able to erase it later.  If you say no, then only the
system operators will be able to erase it.  However, if you
say yes, then you have 2 options:  1) no password, which
means anyone can erase it (this will usually be sufficient);
2) leave a 4 char password, which means only the system
operators, you, or someone knowing this password can erase
the message.  If you regularly send messages to a particular
person, you should both know a particular password, or
use none.
Want description of the Q/S (summary) functions?y
The S function prints a 2 line summary of each message in
the system, from a given starting message number.  For example,
if you have seen messages 1-20, you would ask for summaries
starting with message 21, to see what is new.  Note that
control-C stops printing only the CURRENT entry.  You must
use control-K to abort the summary function entirely.
```

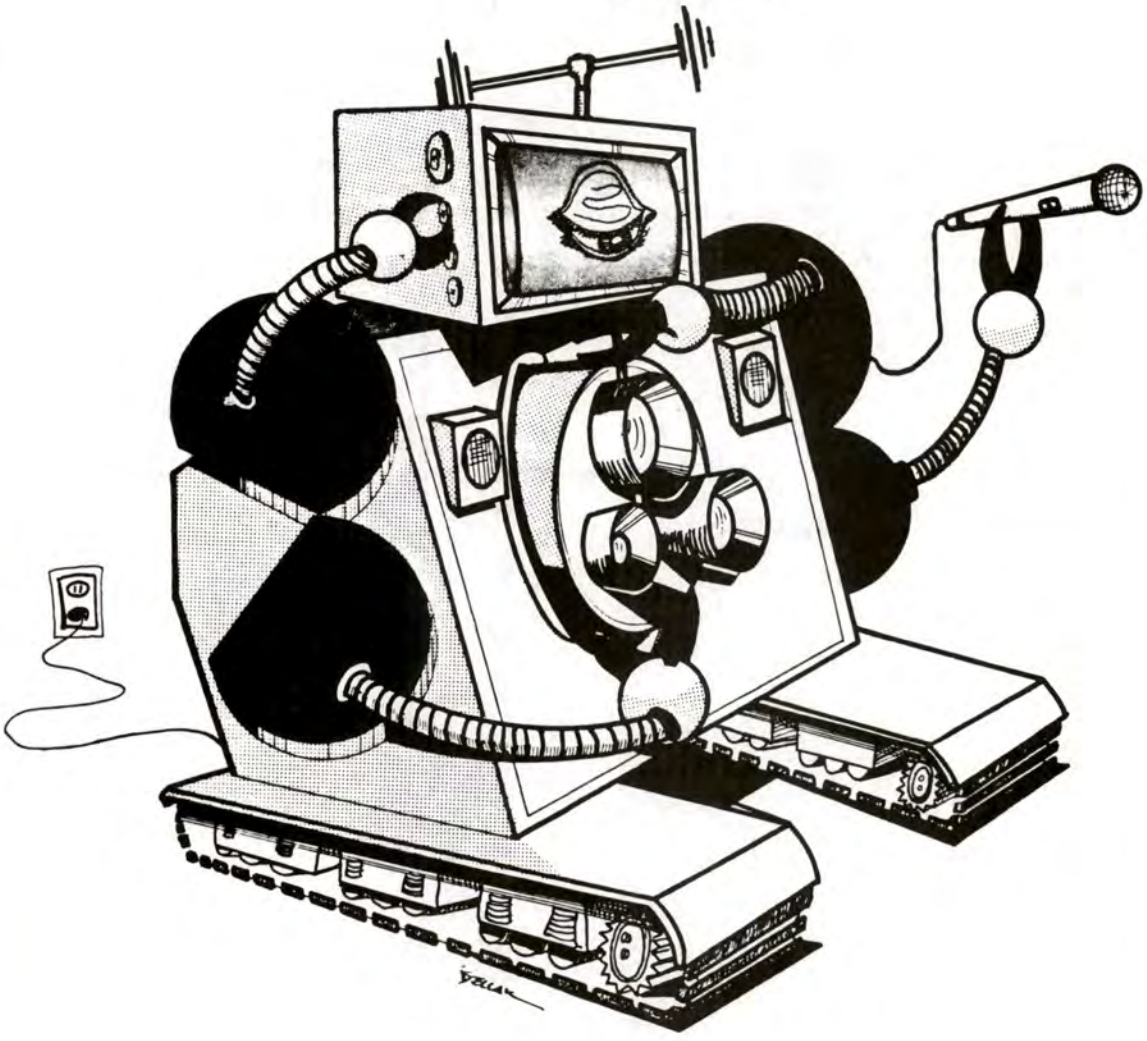
```
The Summary prints:
  Message number
  Number of lines in message
  Date Created
  From
  To
  Subject
The Q function provides a quick summary, (msg #, subject).
Want help about scanning summary for contentsy?y
When you ask for either summary printout, you must always give
a starting message number.  However, you may ask that a field
in the summary header be scanned for a certain value.
For example, any subject containing 'FOR SALE', or any
message from 'SYSTEM OPERATORS'.  To do so, follow the message
number with a comma, the letter for the field to be scanned
(D=date, F=from, T=to, S=subject), an equal sign, then the
value to be scanned for.  Example: 1,T=WARD  (or) 1,S=MEETING
(or) 20,S=FOR SALE
Want description of R (retrieve) function?y
The retrieve function is used to retrieve 1 or more messages
from the system.  You retrieve by message number (use the
S (summary) function to learn what messages you will want
to retrieve).
Want description of the G (good bye) function?y
Typing G when asked for the function, will set you off the
system.  When it does so, it will give you a chance to
leave comments about the system.
Want help about system sending nulls to your terminal?y
Certain terminals require nulls to be sent to them following
a carriage return/line feed, for instance the T.I. Silent
700 series,  The 'N' command is used to set the number of
nulls, to any value from 0 to 9.  However, if you need nulls,
you would want to set them from the beginning.  While the
system is typing, pressing control-N will cause the system
to send 5 nulls after each lifefeed.  Press control-N
repeatedly if it doesn't work at first.
Want to know about B (bulletin) and W (welcome)?y
When you first got onto the system, the systrm typed a
welcome message.  It included some useful information
such as a brief summary of the control characters
accepted by this system.  Then, if there were any bulletin
messages, they printed.  If you missed them (used control-C)
or just want to see them again, command 'W' will retype
the welcome message, and command 'B; will retype the
bulletin if there is one.
Want to know about D function (system stop echoins to you)?y
The D (for Duplex) function switches the system between
Full Duplex (which echoes what you type) and Half Duplex
(which doesn't echo what you type) mode.  Some teletypes
and other devices print immediately when you press a key,
so you don't want the system to echo characters back to you.
Use the D function to switch to or from this mode.
Want to know how to use the system efficiently?y
 1) Minimize connect time: only 1 phone into system.
 2) Use control-C to cancel typing if you know what is
    being typed or don't want to see more.
 3) Use message summary retrieval to see what messages
    you want to see.  For example, if you have seen messages
    1-20, ask for summaries starting at message 21.
 4) USE THE QUICK SUMMARY TO SEE JUST SUBJECTS OF INTEREST.
 5) If you plan on leaving a message, have it well thought out
    to minimize the 'compose' time while you are on the system.
 6) Once you learn the questions being asked, answer
    multiple questions in advance by separating you answers
    with a semicolon (for example: s;1  or  r;23;24;25).
```

```
FUNCTION: B,C,D,E,G,H,K,N,P,Q,R,S,W, (OR ? IF NOT KNOWN)
?s;200,s=cbbs
USE CTL-K TO ABORT, CTL- TO SKIP, CTL-S TO PAUSE.

NO. SIZE --DATE-- --------FROM-------- ---------TO---------

207 05 08/10/78  CLAY RYDER              LOWELL RYDER
SUBJECT:WELCOME TO CBBS

243 15 08/21/78 CHRIS KOEB               SYSTEM OPERATORS
SUBJECT:CBBS IN PHX AZ.

257 11 08/26/78 STAN SKOGLUND            ALL
SUBJECT:CBBS/SAN DIEGO


NO. SIZE --DATE-- --------FROM-------- ---------TO---------

333 03 9/15/78  TERRY TODD               SYSTEMS PEOPLE
SUBJECT:CBBS


      ----END OF SUMMARY----
FUNCTION: B,C,D,E,G,H,K,N,P,Q,R,S,W, (OR ? IF NOT KNOWN)
?g;y
I WILL NOW KEY THIS IN FOR YOU:

FROM:  DAVE CAULKINS

ENTER YOUR COMMENTS.  USE CONTROL-U TO CANCEL (IGNORE) A LINE,
AND DEL OR CTL-H TO BACK UP (AND ECHO) A CHAR.
EACH LINE MAY BE UP TO 60 CHARS LONG (BELL SOUNDS AFTER 55)
ENTER C/R ALONE TO END:
?This was done with a DECWRITER and an/na/a DSM acoustic coupler;
OK - C/R ALONE TO END:
?seems to work fine.
OK - C/R ALONE TO END:
?   12 Sept 78 1200 Tues
OK - C/R ALONE TO END:
?

CHARACTER COUNTS:      204 KEYED BY YOU,      11193 TYPED BY SYSTEM.
FROM THE PCNET:
THANKS FOR CALLING, DAVE

++END OF CONNECTION+
```

The Personal Computer as a Universal Cummunication's Terminal

By: Mark Cummings

ASSISTANT PROFESSOR
BROADCAST COMMUNICATION ARTS DEPARTMENT
SAN FRANCISCO STATE UNIVERSITY
1600 HOLLOWAY AVENUE
SAN FRANCISCO, CALIFORNIA 94132
(415) 469-1787



CHARACTERISTICS OF UNIVERSAL HOME TERMINAL:

ALPHABETIC INPUT AND DISPLAY

NUMERIC INPUT AND DISPLAY

COLOR GRAPHIC INPUT AND DISPLAY

MOTION VIDEO INPUT AND DISPLAY

AUDIO INPUT AND OUTPUT

SOFT COPY STORAGE

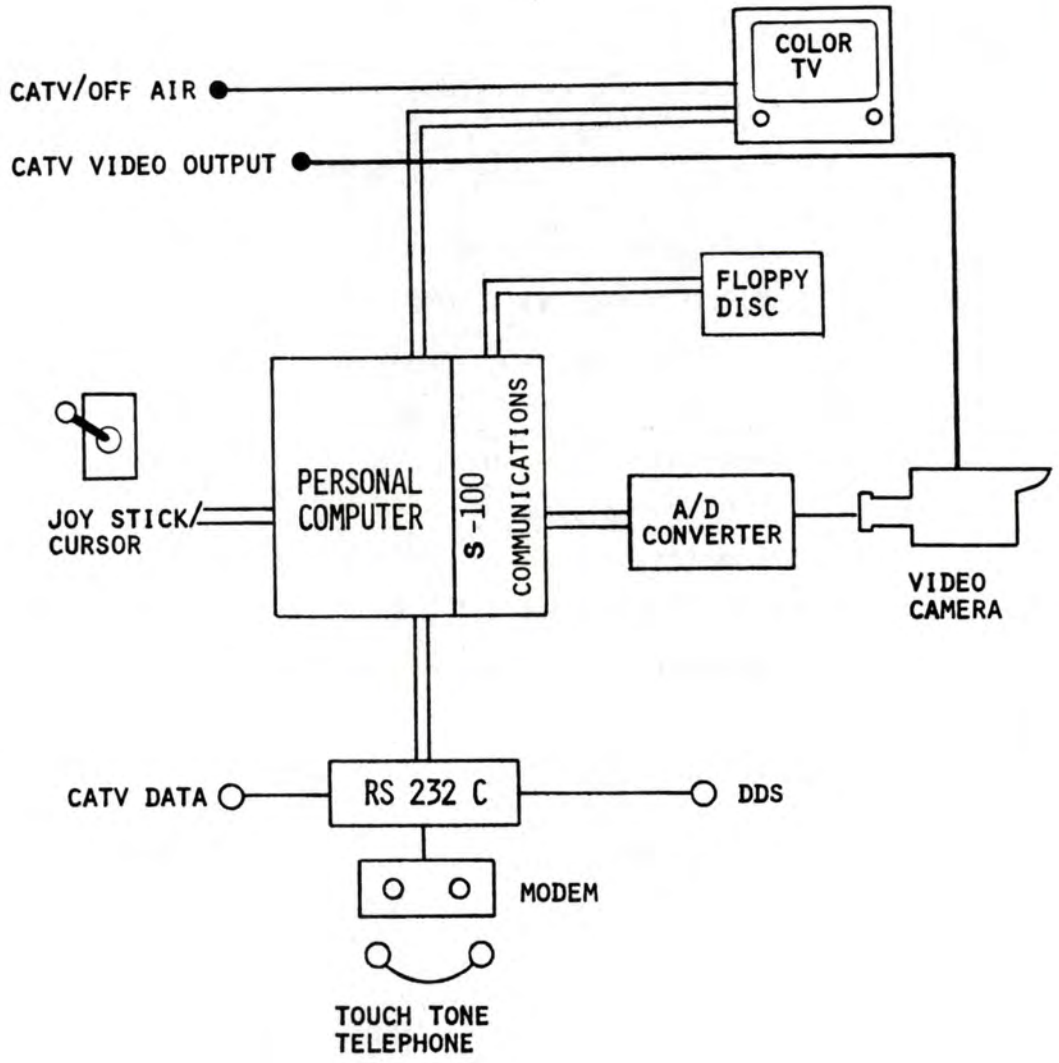
HARD COPY OUTPUT

ACCESS TO COMPUTING POWER

TWO WAY COMMUNICATION CAPABILITY:

- AUDIO
- VIDEO
- DATA

CURRENTLY AVAILABLE DEVICES THAT CAN BE ARRANGED TO PROVIDE THE CAPABILITIES OF A UNIVERSAL TERMINAL:



COST OF OUTLINED SYSTEM:

RETAIL, SINGLE PURCHASE MARCH 1978

PERSONAL COMPUTER.....	$ 1,000.
FLOPPY DISC.....	$ 300.
A/D CONVERTER.....	$ 125.
TV CAMERA.....	$ 175.
RS232C COMMUNICATIONS INTERFACE....	$ 100.
MODEM.....	$ 100.
COLOR TV.....	98% OF HOMES IN U.S. HAVE B&W SETS, 78% HAVE COLOR.

TOTAL COST RETAIL SINGLE PURCHASE.....$ 1,810.00

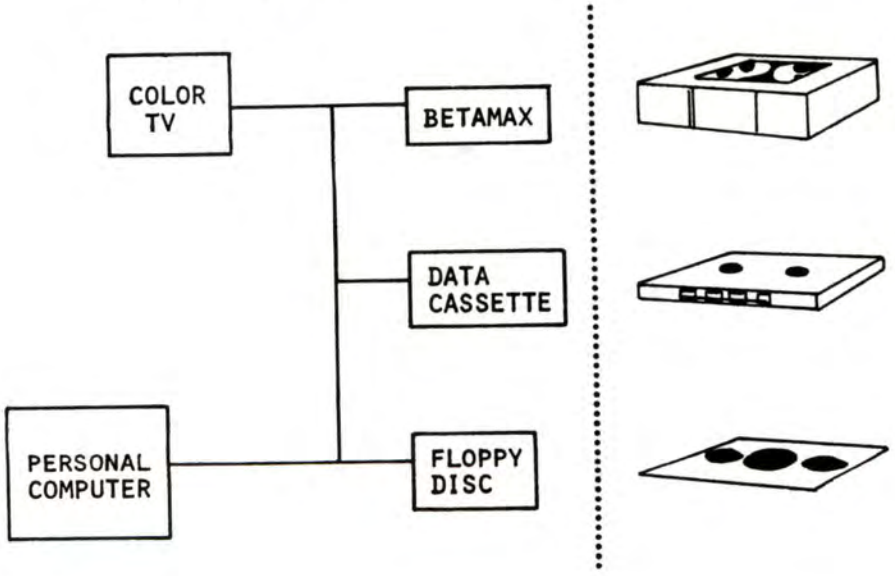
COST IF PACKAGED BY ONE MANUFACTURER.....	$?
COST DECLINE.....	20% / YEAR
COST IN FIVE YEARS.....	$?
COST TO EDUCATIONAL INSTITUTIONS IF PURCHASED IN LARGE QUANTITIES.....	$?

PUBLICATION IN SOFT COPY FORMATS OF:

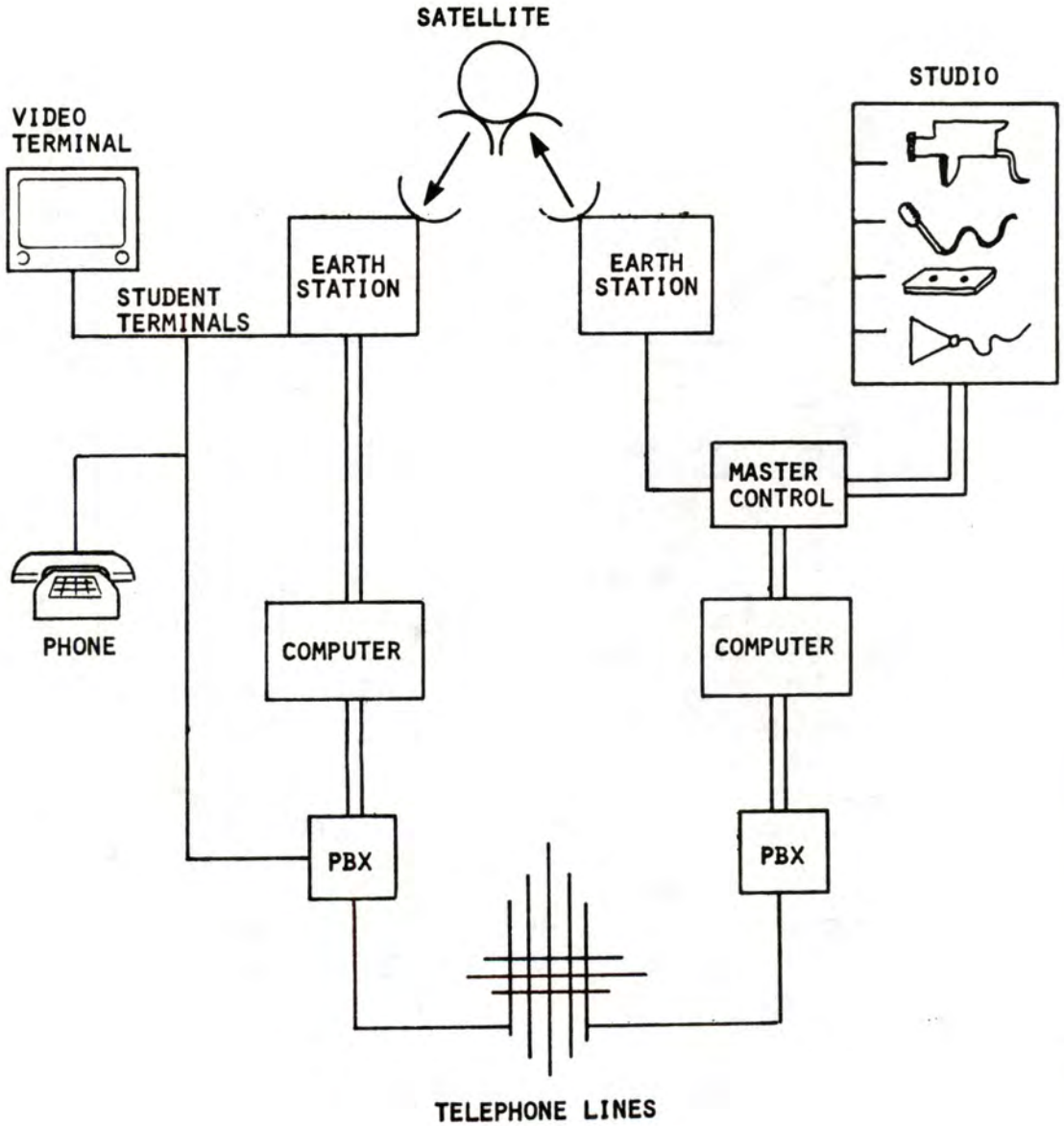
-COMPUTER AIDED INSTRUCTION (CAI)

-GAMES

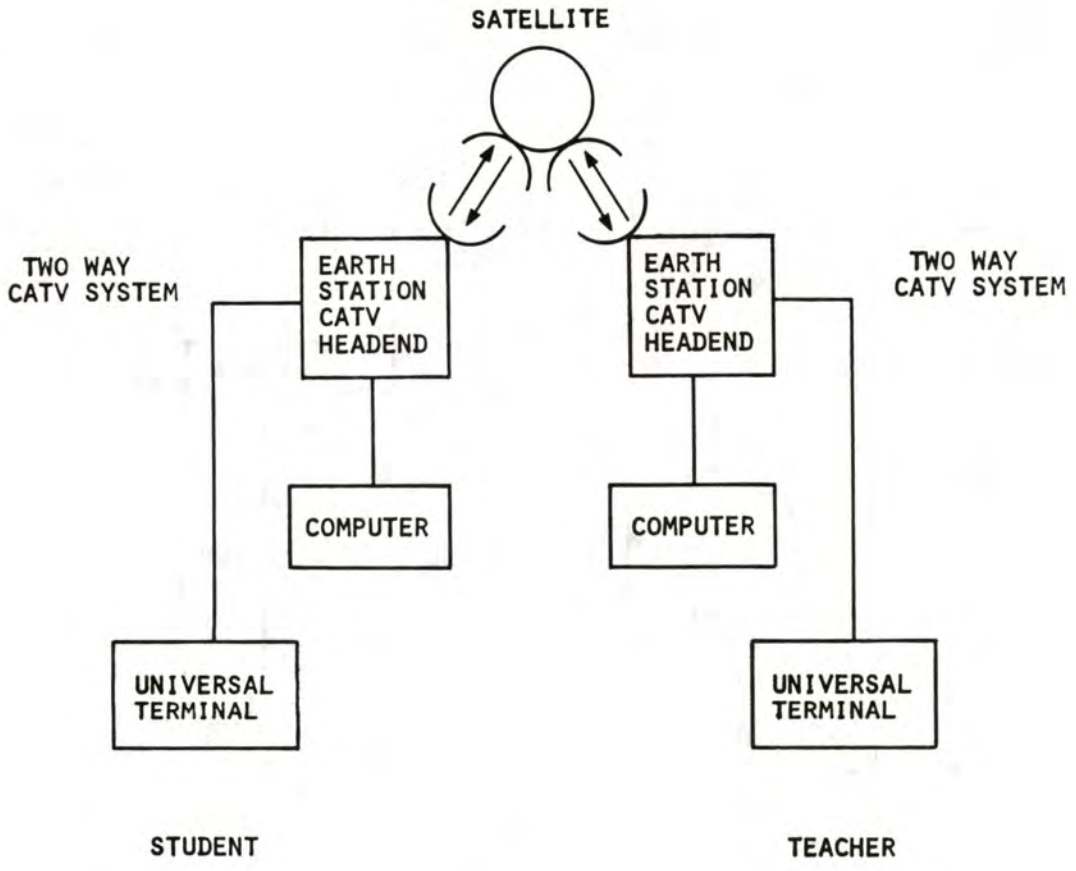
-CULTURAL EVENTS



BASIC SYSTEM



FULLY DEVELOPED SYSTEM



POSSIBLE CONSEQUENCES FOR SOCIETY

* ECONOMIC
  - Democratization of information economy
  - 50% of GNP in the information economy

* POLITICAL
  - Potential for redistribution of power through change in access to information

* AESTHETIC
  - Impact on social paradigm, self image, relationship between man and his environment

* INSTITUTIONAL
  - Given the economics, the benefits, and the goals, what is the optimal configuration of the next generation of mass communications?
  - What structure of public and private institutions might lead to this optimal outcome?

COMPUTER HISTORY: THE EARLY COMPUTER ENVIRONMENT IN SOUTHERN CALIFORNIA

Speakers in this session: Paul Armer, Executive Director, Charles Babbage Institute
Professor Fred Gruenberger, Northridge State University
Professor Henry S. Tropp, Humboldt State University

For those individuals who have become involved in recent years in the Personal Computer development, and equally, for those who have entered the computing environment in the past decade, the parallels between current problems and the problems faced by the small group of computer pioneers will be enlightening, fascinating, and entertaining. Although now the size of the group involved numbers in many tens of thousands, as compared to a group of people too small in number to need a room with the seating capacity of this room at the Convention Center, many of the problems and bottlenecks of thirty years ago are supprisingly similar.

Mr. Armer and Professor Gurenberger were both active participants during the evolution of the digital computer and during its beginning applications in Southern California. Professor Tropp has been active in doing research on computer history since 1971.

Professor Gruenberger address is entitled: "Must Personal Computing Repeat Every Mistake Ever Made?" The title speaks for itself and requires no further elaboration of its content of past blunders and unsolicited advice to current practitioners.

Mr. Armer will describe early gatherings of west coast pioneers and the evolution of PACT, DCA, SHARE, the CPC, the IBM 701, etc.

Professor Tropp will deal with the general environment of this early period and briefly discuss BINAC (predecessor of UNIVAC I), MADDIDA, the IBM 650, etc. and if time allows he will also present material from Dr. Cuthbert Hurd's paper on how the IBM 701 came into existence (Dr. Hurd was IBM's first Director of Applied Research).

Time will be allowed for audience questions and discussion. Also, time permitting, the speakers will discuss some of the current activities in computer history which may be of interest to the audience. (A new Journal, The Annals of the History of Computing; the Charles Babbage Institute, and AFIPS' History of Computing Committee).

SELECTED REFERENCES

Gruenberger, Fred J., " A Short History of Digital Computing in Southern California," RAND Corporation document, P-1599, 26 January 1959. (Also has appeared in Computing News, pp. 145-22/ 145-30, 1958).

Gruenberger, Fred J., "The History of the Johnniac," RAND Corporation, Memorandum RM-5654-PR, October 1968

Gruenberger, Fred J., "18 Symposia," Popular Computing, Vol. 4 (1), pp. 4-7, November 1976

Goldstine, Herman H., "The Computer: From Pascal to Von Neumann," Princeton University Press, 1972

Randell, Brian, "The Origins of Digital Computers: Selected Papers," Springer-Verlag, 1973

Sprague, Richard E., " A Western View of Computer History," Communications, ACM, Vol. 15(7), pp. 686-692, 1972

Tropp, Henry S., "The Effervescent Years: A Retrospective," IEEE Spectrum, Vol. 11(2), pp. 70-79, February 1974

SECOND-SOURCING CPUS: EMULATION, ETHICS, AND ELECTRO-POLITICS

Chuck Hastings, 4890 Hamilton Avenue, San Jose, CA 95130

Abstract

In the traditional computer culture, CPU architectures and instruction sets have been regarded as property and CPU second-sourcing has been considered immoral, not to mention illegal and fattening. In the traditional semiconductor culture, on the other hand, nobody bought your fantastic new chip unless at least one of your hungry competitors second-sourced it, and so second-sourcing became a way of life.

A microprocessor is a CPU on a chip, and microprocessors and microcomputers have suddenly become big business. Hence -- instant cultural collision! While semiconductor firms literally give each other masks and process data in order to persuade reluctant competitors to second-source their microprocessors, they get sued by minicomputer manufacturers for "pirating" and other lurid alleged crimes whenever they pick a minicomputer architecture, rather than a microprocessor architecture, to emulate. Meanwhile, on another front, there are at least a dozen firms around the world which now second-source the IBM 370 line of computers, at some level of "compatibility," and several others are second-sourcing various members of either the DEC PDP-11 family or the Data General Nova family of minicomputers.

Is a manufacturer "stealing" if he emulates another manufacturer's CPU architecture rather than originating one through his own honest toil? Or, on the other hand, is he engaged in a devious attempt to "lock in" his customers if he invents yet another whole new instruction set or input/output interface incompatible with those of any of his competitors? Is a user "trespassing" if he runs software obtained from manufacturer A on a computer bought from manufacturer B who is second-sourcing A's machines without A's permission? Traditional legal and ethical theories haven't always shed much light on issues such as these.

If you don't first ask the right questions you normally don't get the right answers. This paper attempts a fearless foray into some poorly mapped territory, asks some rather off-the-wall questions, and concludes with several sweeping opinions. If you think you might some day want to lay out some bread for a personal computer which emulates the instruction set of some existing minicomputer, you'll need to know this territory, so read on.

"... CPU ARCHITECTURES AND INSTRUCTION SETS HAVE BEEN REGARDED AS PROPERTY ..."

Property Rights vs. Civil Rights

The idea for this paper burst upon me many years ago, when the regional sales manager for one of the major minicomputer manufacturers solemnly assured me -- in answer to my routine question as to whether there was a second source for his company's one-board minicomputers -- that second-sourcing another manufacturer's minicomputer was "immoral." He was quite started when I let out a roar of laughter and uncontrollably broke up for several minutes. I was quite startled that my reaction should have surprised him. We then spent some time cautiously questioning each other, clarifying what we meant, and trying each to understand where the other was coming from, as though we were from two totally different cultures suddenly thrown into collision -- and perhaps we were!

This man thought of a CPU architecture as property. To duplicate this property -- even in the broad sense of a reverse-engineered emulation -- was, in his view, stealing. His company owned and controlled the property, and was not about to permit such duplication. A cash customer who bought a minicomputer could enter upon the property -- run programs -- with his company's blessing. The idea that my employer would really like to see some other

"... AS THOUGH WE WERE FROM TWO TOTALLY DIFFERENT CULTURES ..."

company out there making almost-identical minicomputer boards, which would run the same programs as those which his company's boards would run, blew his mind -- at least, at first.

I, on the other hand, was fresh from an attempt to evaluate about two dozen microprocessors, from as many companies. (This was about four years ago, and many of those microprocessors never did finally make it out into the world as real products.) I was considering his company's one-board minicomputer as yet another alternative to committing my employer to go with some microprocessor which as yet existed only as an "objective specification" (i.e., paper tiger) and to develop a real-time microcomputer around that microprocessor, for a high-reliability application in providing local telephone service. My employer, like virtually all systems houses, had previously gotten burned by depending on sole sources for semiconductors and computer peripherals, and was twice shy of making future commitments to sole sources for anything. Moreover, it was particularly scary to commit to a microprocessor, which not only dictated a pinout and a part performance specification but also by implication rendered non-transportable a lot of expensive real-time software, when nobody was shipping parts yet and only one semiconductor house even had a full-scale development program underway for that particular microprocessor. Thus, I took it as my employer's natural right to try to locate a second source for whatever microprocessor was selected, and furthermore to run any and all software -- whether we wrote it ourselves or acquired it by some other route -- on the second-sourced CPU. In essence, I thought of the right to run software on one's own CPU as a civil right akin to that of driving a rental car up one's own driveway, or to "quiet enjoyment" of a leased residence.

In case you wonder where this rambling anecdote may be leading, consider the current legal hassle between Data General (DG) and Fairchild Semiconductor. DG is probably the world's second-largest minicomputer manufacturer after DEC, and Fairchild is maybe the third or fourth largest semiconductor house. DG's low-end line is the famous Nova family, which the company has produced since it began. DG also has for several years operated their own Silicon Valley semiconductor facility, and reputedly makes a considerable proportion of what they themselves use -- ranging from TTL jellybeans to a one-chip NMOS "MicroNova" microprocessor around which they build various microcomputer products. Fairchild chose the instruction set and architecture of one particular Nova model for their integrated-injection-logic (I^2L, a bipolar technology) 16-bit type 9440 microprocessor, now being marketed under the name "Flame," and somehow never got around to getting DG's permission. DG legally and loudly contested Fairchild's right to do so, and beyond that contested Fairchild's right to run any DG software on 9440-based microcomputers. By now, Fairchild is advertising such microcomputers as well as marketing the 9440 as a semiconductor component for system builders, after some delay during which some Fairchild software got written just in case it turned out not to be cool to be running DG software. However this particular case finally gets settled, it is best understood as a cultural collision, with both companies genuinely convinced that right is on their side in accordance with the ethics of the culture from which they come.

Wry technical note: As a semiconductor component, the 9440 is not particularly compatible with the Micronova. For one thing, it doesn't need nearly as many different supply voltages. As befits a recently-introduced bipolar part, it is apparently quite a bit faster.

DG's frame of mind, in deciding to go after Fairchild, may have had something to do with having sold a license several years ago to another Bay area company to use the Nova CPU architecture and existing software. The company was Rolm, now a major military computer and PBX manufacturer, and the pragmatic basis for the sale was that DG had no interest in selling militarized or ruggedized versions of the Nova and Rolm had no plans to sell any other kind. At the time that Rolm bought a license, it was a tiny startup operation rather than a large and powerful company like Fairchild, and couldn't have afforded a strenuous legal hassle. However, after Rolm paid good money for a license to some things, it certainly would have strengthened the impression that these things were property, which is only a step away from the subsequent conviction that what Fairchild did was "stealing."

Embroiling of Hapless Third Parties

What can happen to a user in a case like this one? On what may he count for reassurance that it won't happen?

If a user U buys a product from manufacturer B which appears to infringe a patent held by a competing manufacturer A, A can sometimes find legally tenable grounds for bringing suit against U as well as against B. This happened a couple years ago in one well-publicized lawsuit over modem patents. And it could even conceivably happen in a case like DG versus Fairchild, where the heart of the case does not concern patent rights, that the minicomputer company might nonetheless attempt to go after third parties who use the semiconductor company's microprocessor in their products. Whether or not a legal action is well founded, it may serve well as a scare tactic to keep customers in the fold who might otherwise switch to cheaper microcomputer boards which run the same instruction set as the minicomputer company's boards, but are based on the microprocessor. It may even -- who knows -- scare off venture capitalists from investing in little companies which are trying to market such boards.

In another interesting situation several years back, effective reassurance was possible since the minicomputer company seemingly turned out not to "own the property" in question after all. By that time, various versions of the DEC PDP-8 minicomputer and its direct predecessors had sold in the tens of thousands of units. Intersil wanted to build a CMOS microprocessor, and was looking for a CPU to emulate on silicon. They considered just two candidates -- the PDP-8 and a popular 8-bit NMOS microprocessor. They chose the PDP-8 primarily on the basis that the resulting CMOS chip would actually be smaller and less complex than for the other microprocessor, plus the expected marketing advantages of the 12-bit PDP-8 word length in a microprocessor. They apparently did not fully realize what a coup it would turn out to be to be the first company to successfully emulate a popular minicomputer in a microprocessor. They did anticipate that DEC would not be altogether pleased with their sincere accolade to the popularity of the PDP-8, however, and accordingly did enough investigation to convince themselves that the PDP-8 architecture (as embodied in various predecessors) had essentially been developed at Lincoln Laboratories before DEC had even been founded, and therefore could be defended as something in the public domain. Even so, Intersil found it prudent to come to terms with DEC before purveying any DEC PDP-8 software to third parties who bought their IM6100 CMOS microprocessor. Having taken these steps, Intersil was in a position to convincingly reassure nervous customers who were apprehensive that they and Intersil would both get sued by DEC.

Total Compatibility and the Fountain of Youth

It is often assumed that compatibility, like pregnancy, is a two-valued parameter -- you is or you ain't. Wrongo. Compatibility has so many qualifiers, ins, outs, variations, and ramifications that it is most prudently regarded as a gray-scale or analog phenomenon. There really is no such thing as Total Compatibility -- there is only compatibility for some particular purpose, such as running one particular piece of code or plugging in one particular peripheral device. "Total Compatibility" is simply marketing hyperbole for compatibility for all practical purposes -- all practical purposes -- and can usually be proved to be embarrassingly non-total by some bright ratfink with a pathologically clever application, which must then perforce be defined by the marketeer as Impractical.

In case you think I am splitting hairs, consider the DEC PDP-11 family of computers. They all do considerably resemble each other with respect to such matters of practical interest as instruction sets and input/output interfaces. It is, however, relatively straightforward to code up a program using two-address instructions which refer to the same general register in both address-specifier fields, in one case using one of the auto-

indexing addressing modes, which will do a variety of different things on the various PDP-11 models, with departures particularly on the older models. To paraphrase George Orwell's famous line in Animal Farm, "All PDP-11s are compatible, but some are more compatible than others." If you want your code to be safely transportable from one PDP-11 model to another, study your DEC handbooks diligently to learn what these impractical cases are, and then avoid using them. There should always be some core set of features, which you can stick to, which operate with sufficiently good compatibility on all PDP-11 models in which you are interested.

Consider also the IBM 360 family of computers -- a quite different case. Here, the manufacturer set up some tough internal administrative controls to ensure both upwards and downwards compatibility (which I'll define in just a minute), since the different models were to be developed by administratively and in some cases physically separated groups of people. In essence, these various groups were each chartered to emulate the same target machine (apparently the middle-of-the-line 360/50), just as if they were all working for separate companies. An excellent degree of upwards-and-downwards compatibility was in fact achieved for the five models intended as mainstream -- 360/30, 360/40, 360/50, 360/65, and 360/75; some of the other models do have departures in various directions. However, even though the front door was locked some incompatibility did creep in the back door by a different route -- optional features. On the 360/65 and larger machines, virtually all features with a serious impact on program transportability were standard, packaged-price items. On the smaller ones, however, one could pick and choose among various options in matters such as memory protection. In the 360/40 case in particular, there were eight or nine of these, with the result that a 360/40 could be ordered in about as many configurations as a hamburger from Wendy's today, and code which accessed these optional features would not necessarily run on an economy-model stripped 360/40 just because it ran fine on one loaded with extras.

Compatibility is best understood from a mathematical logic viewpoint. (PDP-11s, for instance, form a partially-ordered set.) If a piece of code written for model 111 runs on model 222, we say that model 222 is upwards-compatible for that purpose from model 111. Model 111 is then downwards-compatible from model 222 by an inverse definition, just as in formal mathematics subtraction is defined as that operation inverse to addition -- strictly speaking, the statement A - B = C is first introduced as simply another way of stating that A = B + C. If, wonder of wonders, model 111 and model 222 are both upwards-compatible from each other, then they are compatible. However obvious these statements may seem here, what they imply -- and, even more, what they fail to imply -- is frequently ignored, with interesting and often expensive consequences.

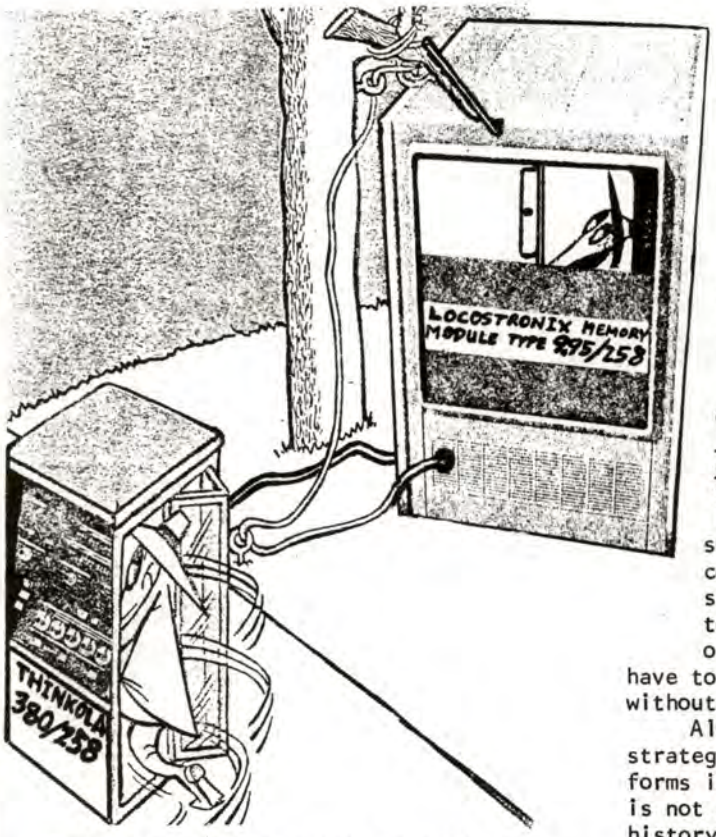
By the way, "compatibility" is probably the most-often-misspelled computer-jargon word in the English language. One sees "compatability" three or four times as often as "compatibility," even in print. Look it up in the dictionary if you think that 40,000 computer gurus can't be wrong. The misspelling may possibly go back to an Ogden Nash one-liner of several decades ago about marital harmony, which goes something like "Incompatibility? No problem -- as long as he has income and she is patable."

Add-On and Add-In Memories and CPUs

Many years ago, some bright entrepreneur noticed that most big mainframe CPUs were connected to their large, impressive memory modules through cables. Aha! If one could put a cheaper, denser, physically smaller, logically larger, more reliable "add-on" memory at the end of a cable having a similar connector, one could replace lots and lots of gigantic impressive memory modules made by the CPU manufacturer and make a lot of money. And this was done, most extensively with IBM 360 and 370 memories, but increasingly with others also. In some cases, where the memory isn't a separate cabinet but merely a module within the CPU cabinet as in the General Electric 435, the second-source vendor (in this case Intel) provides an "add-in" rather than an "add-on" memory. Nowadays, an add-in memory may consist of but a single large circuit card, as does the Mostek add-in memory for the DG Eclipse CPU. Of course, the same business logic has been very well understood from the beginning on the personal computer scene.

It also was noticed long ago that large CPU peripheral devices, notably giant disk files, also connected to CPU input/output channels through cables with standard connectors, providing a similar opportunity. When add-on peripherals became a lucrative business, the CPU manufacturers countered by such moves as pulling the disk controllers into the CPU cabinet so that the interlopers couldn't get to the cable interfaces anymore.

By then, however, some of the interlopers had themselves become large and powerful companies, and their success had inspired some well-financed new startups, and several of these diverse companies attempted to carry the war back onto the enemy's territory by emulating the CPU they were offering add-ons to. Although several of these attempts resulted in financial debacles which almost sank the



"SUBTLE CHANGES IN SIGNAL BEHAVIOR AT CABLE INTERFACES ... HOWEVER INTENDED ..."

interloper company, the strategy ultimately prevailed, and today there are more than a dozen companies worldwide (including five communist-bloc state-owned companies) which essentially second-source several different IBM 370 models, and altogether probably do close to a billion dollars worth of this business a year. A small fraction of IBM's annual sales, to be sure, but still a good piece of change.

White Spy, Black Spy

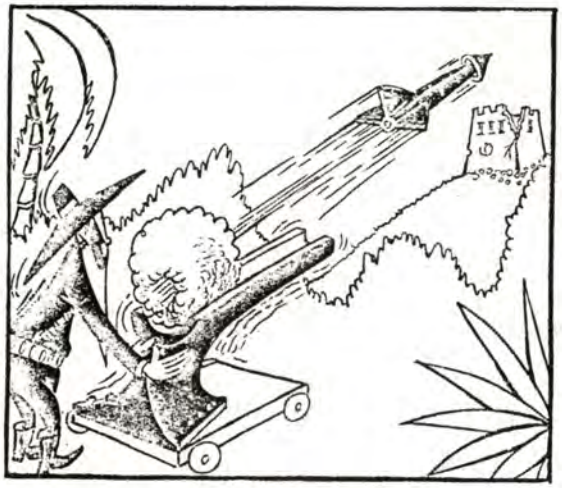
As this second-sourcing business has grown large and successful, a fascinating set of electro-political business strategies has developed along with it.

One obvious strategy is the pursuit of Total Compatibility. In one rather extreme case, a second-source house has emulated an IBM CPU at the microcode level as well as at the instruction level! The rationale is that this capability will make is particularly easy to rapidly track changes made by IBM.

When a company whose CPUs and memory modules are being second-sourced desires to make engineering changes in their products, it may at times be very difficult to distinguish "Our Policy is One of Continuous Improvement" from the counterstrategy to Total Compatibility. Subtle changes in signal behavior at cable interfaces and in operating-system behavior, however intended, can raise havoc in user installations having a mixture of first-source and second-source equipment, unless the second-source manufacturer has been able to anticipate and meet all of the changes. Incompatibility thus becomes a major competitive weapon -- fix things so that when your competitor hooks his box into your customer's system, the whole smash quits working and it's all your competitor's fault. The countercounter-strategy to this one was once described to me by the engineering vice-president of a major add-on memory firm as "You have to hang by a string inside an IBM CPU without disturbing anything."

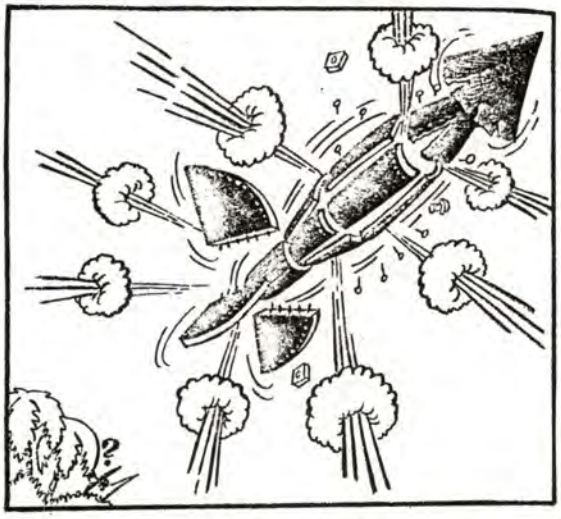
Although the deliberate incompatibility strategy takes on some particularly interesting forms in the computing-equipment business, it is not exactly unprecedented in previous human history. Take, for example, the whole subject of railroad track gauges. The standard American and British gauge is 4 feet, 8½ inches. This not-so-round number was derived in the surest possible way: A powerful major user (the Roman Empire) imposed it on other users (the Britons, whom they conquered a few years B.C.) by decree as a standard for wagon axle widths. The conforming standard wagons then wore standard ruts in British roads for many centuries, and users whose wagons had differing axle widths got a rough ride throughout those many centuries -- or until they conformed. (Sound familiar?) Roughly two millennia later, the first British railway cars were fabricated simply as wagons with steel wheels, and so guess what the track gauge had to be? Elsewhere in Europe, however, many of the non-island countries chose utterly unique gauges for their railroad right-of-way, thus ensuring that any enemy troop trains which came charging across their borders would immediately jump the track. (Not to mention freight trains loaded with goods competing with those of the local merchants.) Closer to home, when Cincinnati built its streetcar system around the turn of the century, a 5-foot gauge was chosen so that the existing interurban railway company in the area could not run its cars over the streetcar right-of-way. The more things change, the more they stay the same.

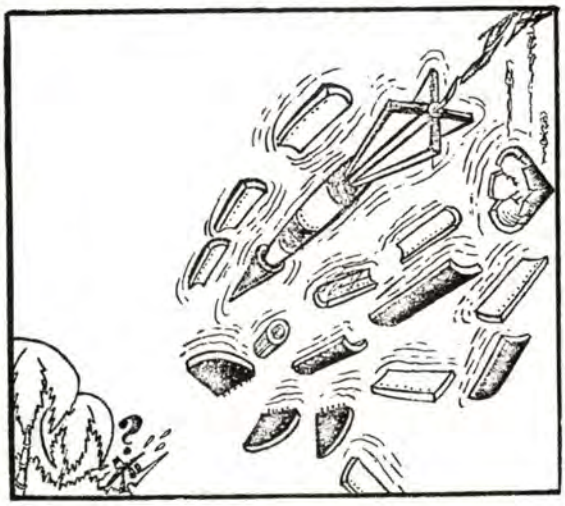
Another intriguing old-time example, having to do with the selective use of upwards

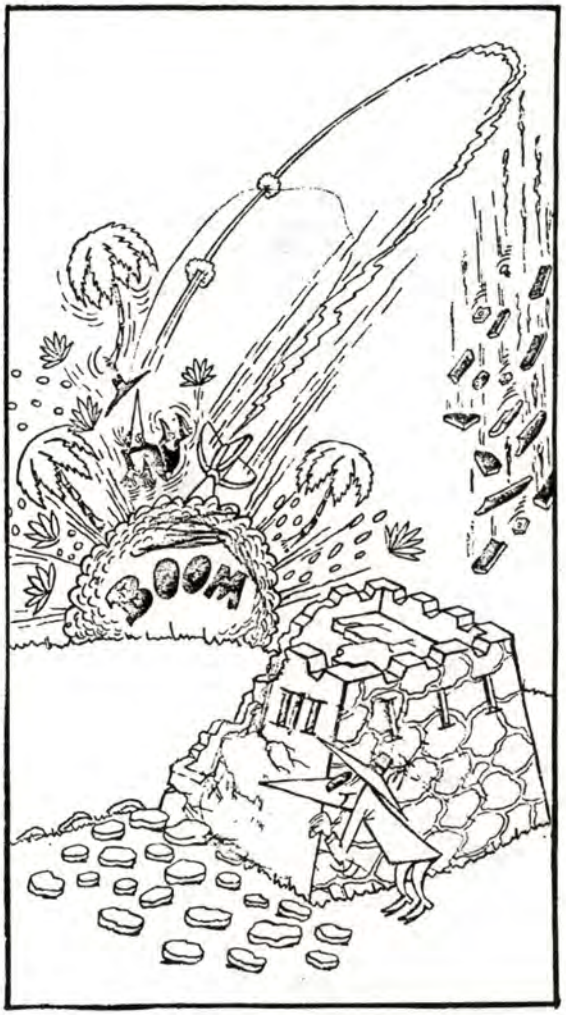


compatibility, concerns field artillery ammunition. Some years ago, the Russians chose to make theirs one millimeter larger in diameter than the existing western-world standard, and their gun barrels likewise. They were still able to shoot off captured enemy-manufactured shells, with no real problems except perhaps for some presumed decrease in accuracy due to increased wobble of the shell in the gun barrel. On the other hand, enemy artillery units which attempted to make use of captured Russian shells were likely to find that these shells didn't quite make it all the way out of the gun barrel before coming to rest and blowing up.

To return to the present-day electronics world, similar games get played at the compo-







"AS OFTEN AS NOT, HOWEVER, THIS DELIBERATE INCOMPATIBILITY GAME BOOMERANGS ..."

nent level. It has been known to happen that a large semiconductor manufacturer G has deliberately introduced a part similar to one from a smaller manufacturer H in an incompatible pinout, citing some rather unconvincing advantages, rather than simply second-sourcing H's part. The obvious intent here is to force systems houses using the part to lay out their boards to G's pinout, and thereby lock H's part out of all those sockets and H out of all those dollars. As often as not, however, this deliberate incompatibility game boomerangs, some other manufacturer second-sources the part in H's pinout, and G's part falls flat in the marketplace until G belatedly comes out with a part in H's pinout after all.

The MAD magazine Spy vs. Spy cartoon series by Antonio Prohias, excerpts from which appear in this text by permission (and with obvious modifications in one case), exhuberantly summarize the wheels-within-wheels booby-trap philosophy which sometimes appears to prevail. However, in Spy vs. Spy, it is usually one of the protagonists who gets blown up. In the real-world second-sourced-CPU and add-on game, unfortunately, it isn't -- rather, it is the customer who simply wants the most cost-effective computing equipment he can buy for his installation, from whatever vendor, and wants it all to play together reliably in place after he gets it.

Standards Activities

Obviously, if instruction sets and interfaces are published in explicit, fully divulged form, and do not get tinkered with after that, many of the devious games just alluded to cease to be effective. Thus, users and second-source manufacturers share an increasingly intense interest in trying to standardize architectural attributes which affect them in some way.

There are hazards to this approach also. The most obvious one is that premature imposition of standards in a technical area will freeze out worth-while future improvements; there is a very legitimate aspect to "Our Policy is one of Continuous Improvement." But there is another one, probably more deadly in practice, that standards inevitably operate to the benefit of some sets of economic interests and to the detriment of others. This eventuality is often foreseen, and the foreseers then dig in their heels and fight like hell to keep any standard from being adopted, or to keep the one adopted toothless. But sometimes it is not altogether obvious who will be the beneficiaries and who will be the losers left out in the cold.

For instance, consider the present move by various forces within the U. S. government to require that all computer procurements over $400,000 in value make use of the IBM 370 input/output channel interface, supposedly to facilitate effective competition among add-on peripherals houses as well as between all of them and IBM. Honeywell has loudly opposed this standard for obvious reasons, perceiving it as a Trojan Horse rumbling through the city gates, and has threatened to withdraw from competing for procurements subject to the standard rather than uprooting their whole input/output architecture and replacing it with the standard. So the standard represents a major cost for those manufacturers like Honeywell who are not compatible with it, unless they are to be locked out and their competitors to be locked in for sales of large government systems. But wait a minute -- why is IBM also rather cool about the standard? Do they want to make some additional changes in the way their input/output channels behave, which they won't be able to do and remain standard? If it isn't IBM's Trojan Horse, just whose is it? Possibly it is Amdahl's. Amdahl is the world's largest IBM 370 second-source manufacturer, and has vigorously supported the standard.

Thus it is difficult to avoid having standards themselves become a competitive weapon, like incompatibility. Get your format specified in, and your competitor's format specified out. When the protagonists are semiconductor firms, read "pinout" for "format."

Even so, the economic logic behind standardization is strong, and attempts to standardize go on. After all, once a standard has taken effect and has been in place for a while, the sharp and admittedly unfair immediate effects on some of the protagonists should die out, and everyone should begin to benefit. Within the last year I have attended some meetings of two quite different standardization committees: one under the auspices of the U. S. Army Electronics Command which is attempting to arrive at a standard architecture for a Military Computer Family (MCF), and one under the auspices of the IEEE Computer Society which is attempting to put forth a comprehensive standard for floating-point number formats and arithmetic in future microprocessors.

Both of these committees seem to have a substantial majority of people sincerely trying to do the right thing. Both of them, however, have tended to make haste very slowly. Any standard which either of them puts forth may tend to make life much easier for some of the agencies and economic interests represented and much more difficult for others, and large amounts of money and even the survival of companies and thus the jobs of some committee members may be at stake!

At one point in time, the MCF committee managed to gather considerable momentum towards standardizing on the DEC PDP-11/70 instruction

set and internal CPU architecture for future military minicomputers, along with cabinetry which bore at least some resemblance to that of the Control Data 480 military minicomputer. At that point there were loud screams from the military-computer divisions of Litton and Westinghouse, which perceived themselves as potential recipients of a Trojan Horse, and the almost-jelled consensus came unjelled. Whatever the recent trends may be, the final outcome is likely to be in doubt for several years. At best, the MCF architecture is likely to become yet one more flavor of military computer in addition to all the others in use, rather than a universal standard.

Generally the MCF effort has seemed to be sane and well-founded, with fairly modest goals and realistic notions as to what the world is actually like, as compared to other such efforts which preceded it. The entire October 1977 issue of Computer magazine (the general-interest magazine published by the IEEE Computer Society) was devoted in essence to MCF, and is worth reading even if you have utterly no interest in khaki-colored computers. This is the Computer issue whose cover portrays heroic-looking eagles against a psychedelic-color backdrop, and if you are at all familiar with some of the earlier military computer architectures which the MCF committee was quietly hoping to phase out, you may react as I did that a much better choice could have been made for the species of large native American bird pictured on the cover. Nevertheless, the papers in this issue range from good to excellent, and the first few have a direct bearing on my subject here from the point of view of one very large consumer of computers.

The floating-point standards subcommittee is still hard at work, and the outcome of its efforts can't yet be predicted. The concepts being discussed at the time I write these words bear some resemblance both to those in DEC's VAX-11/780 supermini and to those in devices in development as part of Intel's 8086 program. However, unlike the MCF committee, this subcommittee is proposing to write a totally new standard which does not exactly correspond to any existing hardware, which reduces the Trojan-Horse aspects somewhat. By the time you read these words, some definitive action may have been taken, but as of now it has not.

The following six maxims are from an excellent paper[1] which should be readily accessible to most of you, and pertain to standards of almost any type as well as to the software standards about which the paper was written:

1. Proposed standards can be very beneficial; Imposed standards will be disaster.
2. Standards are not only meaningless, they are actually dangerous, if their proper use cannot be understood by those affected by them.
3. The most important part of a standard is its applicability clause, which exactly specifies when not to use it.
4. Standardized methods can become standard methods only when accepted in use.
5. Standards exist only to serve the needs of a nonstandard world. Standards cannot create a standard world.
6. Compatibility is more achievable than conformability, and usually is all that is needed.

Several Sweeping Opinions

It's always much easier to point to problems than to figure out what to do about them. Many of the problems I've described are inherent in any situation where information-handling machinery is designed and marketed by human beings with something to gain from its widespread acceptance. They are not unique to one industry, one nation, or one economic system, and they will be with us as long as we continue to use computers. However, my basic subject here is ethics, despite the unfamiliar context; what is it right that we should be trying to do? The particulars which I have discussed seem to me to point to the following generalities:

A. For manufacturers, whatever policies and decisions serve customers better are likely to work out best, both from an ethical viewpoint and for staying in business over the long haul. Customers are what make paydays and dividends possible. If a sufficiently long-term view (probably 3-5 years) is taken, there is virtually 100% correlation between sound marketing practices and following the Golden Rule and other classical Judeo-Christian ethical principles.

B. Conversely, whatever policies and decisions are aimed at manipulating customers are unethical and should be avoided, both because they are wrong and because they will surely lead to loss of market share once the perpetrator gets found out. There have been plenty of examples presented in this paper: playing deliberate-incompatibility games, suing your second sources, and cynical misuse of standards as Trojan Horses.

C. As CPUs become smaller and users almost universally come to understand them as components, the computing world will be be better off with the semiconductor-industry-culture view that second-sourcing is a way of

life than with the minicomputer-industry-culture view that it is immoral. Second-sourcing leads to free interchange of software and add-on boxes, and thus ultimately to greater user satisfaction, more rapid improvement in technology, and thus even to greater national security and better service to Mankind. These considerations may in time give software transportability and plug-compatibility the force of civil rights, transcending the property rights which manufacturers hold -- or think they hold -- to architectures.

D. A CPU architecture which has become very popular, because users have found that it serves them well, eventually becomes bigger than the company which originated it. Users acquire a huge investment in existing software, and prefer to retrofit new hardware to this software rather than rewriting it. If the economic mass of this software is sufficiently large, second sources will appear for all kinds of things which users need, including CPUs. Any attempt by the original manufacturer to stymie this process will, if successful, ultimately cause his CPU architecture to lose out in the marketplace to other architectures with more second sources.

E. In consequence of D, manufacturers should make some real effort to avoid inventing new instruction sets where they are not absolutely necessary. It serves the customer to give him a superior new CPU which doesn't force him to junk or rewrite all of his software. The world needs a new instruction set like it needs a new species of mosquito, or supply your own simile.

F. The most durable standards seem to be those which start out with one company's definitions, spread out to cover much of the world because of high-volume production of something, become accepted in use, become somewhat diluted as slightly mutually incompatible tweaks get made in them along the way to adapt to particular user needs, and then get pulled back together by a broadly-based user standardization effort. The IBM 370 instruction repertoire has by now been through most of these phases and is well on its way to becoming a world standard, even though by now IBM themselves might well like to bring out some improvements which would not be IBM-370-compatible. Much the same thing may also be said of the DEC PDP-11 CPU architecture, the DG Nova architecture, and the Advanced Micro Devices 2901 bipolar microcomputer building block; and in the personal computing world the S-100 bus has already reached the very last stage of this evolution. Fill in your own favorite example; mine is the 4 foot $8\frac{1}{2}$ inch railroad track gauge, defined by the Romans two millennia before railroading began.

G. Customers don't buy hardware -- they buy solutions to problems. Even a large-magnitude improvement in hardware will get a chilly reception if it creates new problems, such as incompatibility; it is worth a large amount of effort and cost to keep the new improved machine at least upwards-compatible from the old one.

H. As to why manufacturers should pay attention to such altruistic-sounding maxims as I have stated, consider these words of Robert Townsend[2], for several years #1 at Avis Rent-a-Car:

> "... money, like prestige, if sought directly, is almost never gained. It must come as a byproduct of some worthwhile objective or result which is sought and achieved for its own sake." #

Proper Thanks

There are several people I can publicly and gratefully thank:

Permission to use the Antonio Prohias Spy vs. Spy cartoons was kindly granted by William M. Gaines, Publisher of MAD magazine, on behalf of the copyright owners.

The European railway gauge anecdote came from a talk by Tom Steel of System Development Corporation to the Minneapolis-St. Paul Association for Computing Machinery chapter some time during the mid-1960s. His subject was software standards and compatibility!

The Cincinnati streetcar and Russian artillery shell anecdotes I owe to Dick Delp of Signetics, who also happens to be Chairman of the IEEE Computer Group Subcommittee on Floating-Point Standards.

There are also many other people whom I am grateful to, but had better not publicly thank. This whole subject is at times a hot potato, about which people express strong opinions sharply at variance with the official positions of the organizations which pay their salaries. I have heard and appreciated many such opinions, as well as some milder ones.

So it is best that I express my appreciation in general terms, without publicly identifying all the people I am thanking!

Anyway, there are people -- and even whole companies -- out there sincerely trying to do the right thing. When you meet one and like their act, tell them so.

Who Dat Say Dat

Chuck Hastings has spent the last half of the 1950s as a programmer, the 1960s as a computer architect (whatever that is), and the 1970s as a hardware designer specializing in superminis. He currently works for Microcomputer Systems Corporation in Sunnyvale, CA. He has previously been with National Semiconductor, Itek Applied Technology, Racal-Milgo, two subsidiaries of United Telecom, Control Data, Honeywell, and TRW, and has moonlighted for various big and little companies in four states. He has a BA in physics and math from Grinnell College, an MA in math from UCLA, some additional EE and computer science courses from the University of Minnesota, and by now a few MBA courses at the University of Santa Clara. He has two U. S. patents in electro-optic mass-memory technology, belongs to ACM and IEEE, and still thinks that computers are a good way to have fun and get paid for it.

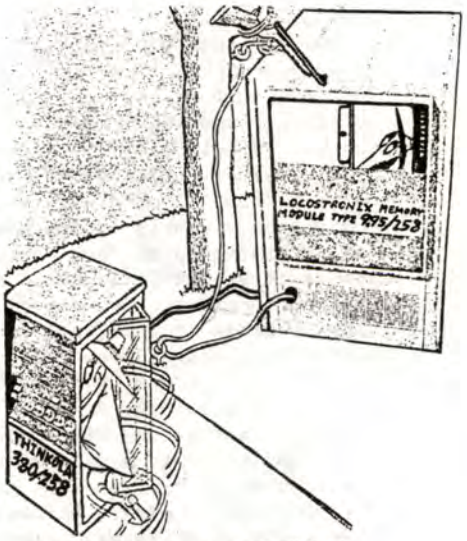
References

1. "Homilies for Humble Standards," Douglas T. Ross, pages 595-600, Communications of the ACM; 11/1976.

2. Up the Organization, Robert Townsend, Alfred A. Knopf, New York, 1970; quote is from page 62.

"... AS THOUGH WE WERE FROM TWO TOTALLY DIFFERENT CULTURES ..."

"... CPU ARCHITECTURES AND INSTRUCTION SETS HAVE BEEN REGARDED AS PROPERTY ..."



"SUBTLE CHANGES IN SIGNAL BEHAVIOR AT CABLE INTERFACES ... HOWEVER INTENDED ..."

AUTOMATED COMPUTER CONTROLLED EDITING SOUND SYSTEM (ACCESS)

William R. Deitrick, Pres., Mini-Micro Systems, Inc., 2101-H W. Crescent Av, Anaheim, CA, 92801

This paper will describe the most revolutionary advance that has been made in Post-Production Sound Editing over the past fifty years.

INTRODUCTION

The project was initiated by Neiman-Tillar Associates, an independent Post-Production Sound Editing company, located in Los Angeles, California, when they decided to utilize computerization to bring Post-Production Sound Editing out of the dard ages after observing the remarkable progress being made in the video area. The initial attempt was a very disappointing experience, but sufficient, albeit painful, knowledge was obtained that enabled a small group of engineers and programmers (Mini-Micro Systems of Anaheim, California) to continue the task to a successful completion.

The sequence of events will be chronicled as development proceeded. Operation of the system, and hardware and software descriptions, will be covered. An abbreviated general description of the capabilities of the system and the hardware follows.

General

The initial concept of ACCESS was that it would be constructed of "off the shelf" electronic equipment (to minimize engineering research and development costs) and would be flexible enough to permit all the required taskes to be accomplished with software. It became increasingly apparent that the tasks to be performed had been woefully underestimated and the availability of the supposedly "off the shelf" equipment had been unbelievably overestimated (not an uncommon occurrence). The system was to be completely digital with a sound effects library digitized and stored on rapid access mass storage (disk drives). This would be feasible if the sound effects could be digitized and compuressed using software to minimize the amout of storage required. The hardware and software to achieve data compression/decompression not only did not exist, it does not today, and probably will not in the foreseeable future! By the time this was discovered, the commitment had been made to produce a system no matter what was required. As it turned out, the subsequent solutions to the problem were more efficient anyway. Hardware was designed and developed to allow real time operation, and software was created to permit the editor to use the machine as a tool to allow him to do what he does best and let the computer do the rest.

Capabilities

ACCESS is presently capable of performing the following:

1. Sound loading
   a. Audio can be loaded into the system from any source and can be controlled manually or automatically. The sound is digitized and stored temporarily and can then be edited manually by executing start/stop commands to establish beginning and ending points or the computer will edit those points using selected parameters. An identifying key and description is entered and the sound is then transferred to permanent mass storage called the sound library which is a removable 200 megabyte disk pack. There can be up to seven of these library packs online (instantly accessible to the computer) at any time.
2. Sound editing
   a. One or two sounds may be retrieved and played simultaneously so they may be modified (volumne, pitch, equalization) and/or synchronized to video tape using SMPTE time code. The system saves all this modification and timing data and inserts it in an assignable list (cue). These cues can then be played automatically in sync with external equipment, e.g., VTR and sprocketed 35mm three track magnetic tape.
3. Data utility programs
   a. These are a myriad of functions generally dealing with administrative or housekeeping tasks such as formatting storage areas detecting code or data input errors, listing of sounds, modifications and cues, program assemblers and editors, system diagnostic tests, etc.

Hardware

The ACCESS hardware is comprised of one 50 megabyte disk drive and up to seven 200 megabyte disk drives, a floppy disk drive, a 300 lpm 132-column printer, a CRT terminal with keyboard, an operations console containing controls, indicators, and displays for manual operation, two microprocessor units with up to 56 kilobytes of memory, two auxiliary memory banks of up

65 kilobytes each, three sound channels containing up to 65 kilobytes each, a two-channel independent fast access memory bus (DMA), and assorted peripheral controllers for interfacing to the external equipment such as video tape recorders, magnetic tape recorder/players, sound amplifiers and level monitors, and SMPTE code conversion units.

Summary

The ACCESS has been used to created sound effects tracks for several feature films and many television shows over the past twelve months. The system has enabled the editor to expand and utilize his creative abilities immeasurably. Production has generally increased fivefold. In addition the system alleviates the physical work formerly required because the editor never even sees tape anymore. The surface of this new technology has only been scratched as the flexibility of the system permits ongoing software development to continually increase its value.

Abstract

ACCESS was developed to enable the Post-Production sound effects editing phase of movie and TV production to utilize "state of the art" electronic equipment and computerization. Present editing methods are performed almost completely manually by splicing prerecorded sound effects on 35mm sprocketed magnetic tape using a movieola to allow synchronization to the picture. ACCESS eliminates the need for manually handling tape and allows electronic synch using SMPTE time code, provides instantaneous availability of sound effects by digitizing the sounds and storing them on magnetic disk packs, and permits creating or modifying sounds electronically using computer control or computer assisted control using a two channel operations console (similar to an automated mixer). The editor can work on any part of the film and can build as many reels as desired and in any order. All entries are error checked by the computer and where possible generated by the computer. Various printouts are furnished such as final mix sheets, sound library contents, etc. The system is completely digital and can be synchronized to SMPTE code or generate it which permits interlocking external equipment for inputting or laying off sound tracks. Production has generally increased fivefold using ACCESS.

Introduction

The Post-Production phase of movie and TV production generally consists of adding sound effects, music, and dialogue clean-up. These tasks are performed today in much the same way as they have been for the past 50 years. This might have been reason enough to cause the creation of ACCESS but there are several others. The advances made in video editing, primarily in video tape, have been spectacular due to the progress being made in the electronics industry. The amount of product required is steadily increasing especially in the TV field. This results in tighter schedules and like everything else, the cost of all aspects of production is going up and time is something one can rarely buy. Sound quality has not been considered an overriding factor in the past particularly in film where all the sound effects tracks, dialogue and music were mixed down to one track and that ended up on an optical stripe on film that yields questionable audio quality. Theatre acoustics and audio sound systems are being improved. This trend is encouraged by productions such as Star Wars. Improvements are being made in TV set sound systems as well. The viewer has been the beneficiary of these advances and is becoming more discriminating (or spoiled) and he will of course, expect more.

Background

A brief description of the normal procedure (not using ACCESS) will assist the reader in recognizing the improvements and advantages realized by using ACCESS. Only the sound effects editing tasks will be described and it should be noted that this is a general description as the procedure will vary among companies.

The editor receives a work print reel of approximately 10 minutes in length (35mm sprocketed). This is then previewed on a movieola and notes taken of the sounds required and the footages needed. The editor then selects a tentative list of sounds from the library index and if he is unfamiliar with the sound it must be pulled so he can audition it or he will have to wait until he receives the transfers. The tape master library may be ¼ inch tape with the sound effects separated by leader and a tone burst. A technician will then take the list and copy the sounds onto 35mm sprocketed mag tape. There will usually be several of these transfers made of each sound. The editor then receives the transfers and b egins the cutting process. One 10 minute reel of picture may require any number of sound tracks (8 is a rough average). The physical distance between sounds on a given track must be adequate to allow the mixer to adjust his board controls if necessary from one sound to the next. If the editor wants to make level changes to the sound he must remove oxide from the tape using a razor blade or a dissolving solution. There is no way he can increase the level however. If the sound he needs is too short he will have to create a physical loop of tape so the sound repeats and he must be extremely careful to

avoid the repetitious characteristics that are very obvious. Synchronizing the sound to picture is literally cut and try. Sound transfers are spliced to leader and the splice continually adjusted until synch is obtained. The only sound control the editor has on the movieola is a volume control which permits him to hear a volume change that the mixer will have to make. One cannot appreciate or comprehend the amount of work involved until you observe the editor knee-deep in tape, with stacks of transfers around him, cutting, splicing, threading and unthreading the movieola, while attempting to achieve the "right" effect on a mechanical machine whose noise defies description and whose sound quality leaves a lot to be desired. The final task is to assemble the edited cuts into 10 minute reels maintaining synch with the picture. These subsequently will be mixed down onto a final track and then be mixed with music and dialogue. The preceding operations are all manual and quite time consuming especially if the original sounds chosen are not "right." Sound quality is degraded with each transfer from library master to work copy to final mix due to the accumulative signal/noise ratio loss. Continuous cueing and copying from the library master tapes also contributes to the problem.

Original Concepts

The initial concept of ACCESS came about as a result of the attempts by Neiman-Tillar Associates (NTA), an independent post-production sound editing company located in Los Angeles, to modernize the editing process just described. A considerable amount of effort was expended in search of a solution with responses ranging from "it can't be done" to "several years and a lot of money." The project was originally begun by ISYS Technology, now inactive. Their approach was quite ambitious and required some yet to be developed techniques that appeared feasible but in practice became very difficult to implement. Basically their solution was to have a 15 channel control panel (similar to a mixing board) which would allow up to 15 sound tracks to be played and modified independently and simultaneously in real time. Each channel would have its own microcomputer and would accept digitized sound data that had been compressed digitally and stored in moving head disk storage. Each channel would decompress or expand the sound data back to its original form. Compression ratios up to 10,000 to 1 were anticipated but this proved to be impractical if not impossible to achieve. The reasons for this were due to the frequency complexity of sound effects being badly underestimated which made classification and coding extremely difficult which in turn created a processing problem for the channel as it would have to regenerate the sound in real time in addition to the normal tasks it had to perform. The problem was compounded when it was discovered that the 8 bit digital sample size was not adequte for quality sound reproduction for sound effects. The preliminary work had been done using music to empirically determine digital sample size and sampling rate. The 8 bit companding DAC came out at that time and although it provides an effective 12 bit resolution and good dynamic range it also introduces unbearable distortion for some sound effect. It would also be very difficult to process an 8 bit non-linear word. During this period ISYS became involved in a legal dispute over another contract (in a non-related field) involving another company and ISYS was deactivated. By this time NTA had become committed (if not possessed) to seeing a system at almost any cost. A small group of consultants then assumed the task and formed the nucleus of a new company to be called MIni-MIcro Systems. The project was redefined using a more practical, realistic approach and subsequently and efficient, workable system was developed -- ACCESS.

HARDWARE

General System Description

The ACCESS hardware is comprised of one 50 megabyte (MB) and seven 200 MB moving head, removable disk pack, disk drives. There are two microcomputers, auxiliary memory banks, three sound data channels and a two channel independent high speed memory bus (DMA) which interconnects all of the preceding devices. There are various peripheral controllers for interfacing and controlling external equipment such as video tape record/playback units, magnetic tape recorder/players, sound amplifiers, level monitors, and SMPTE code conversion units. The CRT terminal (with keyboard), a 300 line per minute printer, video monitor, speakers, and the operations console which contains switches, indicators, and sound modification controls are all located in the operations room which is about 60 feet from the computer room where all the other equipment is located. The computer electronics hardware is mechanized on two sided printed circuit cards (PCB) approximately 5 x 8 inches, with wire-wrapped integrated circuit sockets, and interface connectors on both ends. The PCBs plug into standard 7 inch high, 19 inch wide card cages with combination wire-wrapped and printed circuit backplanes. Ribbon cables on the back edge of the PCBs interface between card cages, and to external peripheral equipment. The card cages, disk drive controller, and power supplies are mounted in a standard cabinet 84 in H x 19 in W x 24 in D.

Major Component Description

The master computer (MACPU) contains a microprocessor, CRT terminal interface, 36 kilobytes (KB) of memory, disk controller (DCU) interface, arithmetic/logic unit, interrupt logic, printer interface, a direct control interface to the monitor computer (MOCPU), and the master DMA control logic. The MACPU performs most of the

data processing load, handles interrupts, issues control tables and checks status of the DCU and DMA. As the name implies the MACPU is the master of the system and essentially controls the entire system. The CRT terminal and the printer interface directly to the MACPU. The DMA consists of two independent 8 bit data channels that allow simultaneous data tranfers between any two pairs of devices on the DMA. Data transfer rates vary with the fastest rate being memory to memory transfers, 2 megabytes (MB) per second, and the slowest when the DCU is one of the devices, 806 KB/sec. There are six major devices interconnected on the DMA -- MACPU, MOCPU, DCU , two auxiliary memory banks (AUX1, AUX2), and the sound channel DMA controller (SCHDMA) which coordinates data transfers to two sound data output channels (SCH1, SCH2) and one sound data input channel (SCH3). The MOCPU is almost identical to the MACPU except for the resident peripheral controllers. It handles all SMPTE code operations for synchronization purposes and has a floppy disk controller interface. It interfaces to all three sound channels via an independent 8 bit bidirectional data bus which is used to send/receive sound modification data, issue mode controls, monitor sound data memory balance counts, and is also used to control and monitor status of the external video tape equipment and audio tape equipment. The AUX1, AUX2 memory banks contain 28 KB each and are used primarily as buffer storage areas. The DCU is the most sophisticated device in the system. It can control up to 8 disk drives, each capable of storing 200 MB on a removeable disk pack. Each pack can hold 40 minutes of digitized sound data so 4 hours and 40 minutes of sound effects (7 packs) can be on line and instantly available to the system. One drive is used as the system software drive and is also used to hold sound modification data, master library index, system maintenance and test routines, etc. Normally only a few seconds of a sound effect are required to be stored in the library. That basic piece of sound can be modified in an infinite number of ways and then only the mod data need be saved which requires virtually no storage space by comparison. The DCU requires a minimum of intervention from the MACPU which initiates operations and monitors status. The DCU accesses its command tables and transfers data under DMA control to/from any device on the DMA except the SCHDMA. Command tables can be linked or chained which allows a series of operations to be performed automatically. The DCU will select a drive, move its heads to a specified cylinder, search for a selected record or key, and when located perform a read or write data block operation, and then signal the MACPU that it is finished. The DCU can execute 25 commands and can provide full track buffering to ease timing constraints. The sound channels each contain 36 KB of memory configured as first in first out (FIFO) memory. SCH1,2 accept digital sound data and the analog sections read the data out and convert it into an analog voltage, sent it to the output multiplexer which routes the sound voltage to external equipment such as audio amplifiers, level monitors, and tape recorders. The nominal output sample rate is 50 KHZ and the digital sound sample size is 12 bits. SCH3 is identical to SCH1,2 except for the analog section which accepts, scales, and conditions a sound voltage input from any external source or from the outputs of SCH1 and/or SCH2 through the input multiplexer. The input sound is digitized at a 50 KHZ rate into a 12 bit word and written into the FIFO where it is read out to the DMA. The operations console contains the modification controls and indicators for two sound channels (effectively an automated mixer with added capabilities). Each channel has a volume (level) control, frequency (pitch) control, six equalization controls that provide a range cut and boost from -15db to +15db over three frequency bands (each adjustable). There are also bandpass shape selection switches for each band. There are enable switches and indicators for each of the controls, system status and mode displays, and a SMPTE time code display. There are remote controls for both video tape and audio equipment.

SOFTWARE

A detailed description of the software is beyond the scope of this paper and it is difficult to describe in a general manner and still be meaningful. In order to realize maximum efficiency and flexibility from the system the hardware was designed as general purpose as possible to permit continuing software development to expand the capabilities of the system. A very good balance has been achieved whereby hardware handles real-time system requirements such as sound data transfers, time code processing, interrupt generation, high speed 16 bit arithmetic and logical operations, etc. The DCU performs many operations while requiring very little intervention from the MACPU which allows maximum processing time. Programs are generally modular and self contained which permit modification and up-grading without impacting the entire system. The power of the DCU enables the computer to use virtual memory techniques which allow the system disk to look like program storage. Program data blocks can be moved between DCU and MACPU very rapidly, 36 KB in less than one tenth of a second and MACPU processing is suspended for less than one third of that. The variable record format capability of the DCU permits optimum program and data block size. Indexed sequential access methods (ISAM) techniques are used allowing the DCU to locate and fetch a sound (begin transfer) in less than one tenth of a second. The editor seldom has to wait on the system because of programming delays. The menu concept is used where the editor is presented a list of functions, sounds, etc., which are displayed on the CRT. He then responds by choosing the desired item by entering a letter of number corresponding to that item. Direct questions are displayed

requiring only a single entry to respond. Keyboard entries are kept to a minimum as much as possible. Entries are error checked and error messages displayed if the editor makes a mistake such as an invalid time code, specifying overlapping sounds on the same track, attempting illegal operations, etc. If auxiliary information is allowed such as descriptions of sounds or modified sounds the entry format is free form, i.e., the editor can enter anything he wants. There are no "computerese" type entries required. All information entered or generated during operation is saved by the system to eliminate redundant entering of data which permits many tasks to be performed virtually automatically. Complete recall and recreation of previous work done at any level is kept available so changes can be made quickly no matter how long ago the original work was done. Reports are generated in various formats and printed for use by editors, sound technicians, and mixers. System software utilities consist of a modified INTEL assembler and editor for program generation, DCU format, test routines, disk pack evaluation, memory test programs for fault detection and isolation, and system diagnostics.

OPERATION

A brief description of the major programs and their functions available to the editor will be covered although it should be noted that approximately one hour is required for a live demonstration of the general capabilities of the system.

Editor

This program is used to fetch and edit a sound(s). The desired sound is selected by entering its key ID (up to 15 alpha/numeric characters) or the list program will display a screen full of sounds with their ID and description for selection. The display can be started at any point in the library and will be continually displayed until the end or until the editor selects one by entering a number associated with the sound. Previously modified sounds can be called by entering the mod sequence number and they also can be displayed on the CRT. The selected sounds can be played for listening only or for modification using the operations console. The sounds will be played upon command from the editor or if the VTR is being used for picture display when SMPTE time code agreement is found. (All picture materials is recorded on 3/4 inch video tape cassette with SMPTE time code super imposed on the picture and also recorded on one of the audio tracks. If production sound was available it is recorded on the other audio track.) The editor can choose one of the following functions -- play the sound as it exists in the library, play a previously modified sound, create background (continuously repeats the sound), or edit the sound. The edit function permits modifying the length of the sound by progressively delaying the beginning or shortening the end of the sound or will enable the editor to select any portion of the sound. After the sound is played he can again select one of the functions or if mods were made he can reset them or save them. If the mods are saved the system assigns a mod sequence number and after the editor enters a description it is entered into the library. Mods made using the operations console are retained and replayed so mods can be made to mods on successive passes without losing the previous mods (automated mix feature).

Prescript

This program is used to create up to 19 sound tracks (10 min. each) for up to 23 picture reels (10 min. each). For example a 1 hour show will typically consist of six 10 min. reels, each requiring seven sound tracks for a total of 42 tracks. The editor will assign the reel number, track number, start/stop time code, and the sound ID or mod number and the program will assign a chronological sequence number within the track and error check the entries. The accumulated track data is always available for display or printing. Once entered the entries can be modified, deleted, or moved to any tracks or reels very simply. This program also consolidates all data pertaining to the show being worked on -- sounds, mods, timing data and stores is on one (or more) disk packs called the "show" pack. This permits an entire show to be played or layed off without the need for all the library packs (where the sounds originated from) to be on line. It can also be used as a library pack and allows an entire show to be saved in the physical space required for about five 35mm 1000 foot reels instead of the typical 42 reels required for a one hour show. Changes can be made and layed off almost instantaneously using the show pack except the basic sounds. A show pack is not normally retained after a show has completed final production. When a show pack is to be recreated the programs will tell the editor which library packs need to be on line. Up to 80 show pack histories can be accomodated by one show history pack.

Play Tracks

This program permits the editor to play back one or two tracks (of the same reel). The editor enters the reel number, track(s) number, and if the track playback is to begin at a point other than the beginning of the track the sound sequence number is entered. Playback commences upon command from the editor (the SMPTE time code is simulated by the program) or when SMPTE time code agreement is found with the external equipment (usually the VTR). As with all programs the operation can be terminated at any time by the editor.

Record

This program is similar to the Play Tracks

program and it is used to lay off or record sound tracks to an external device (usually a single or multi-stripe magnetic tape recorder). An output multiplexer allows the tracks to be assigned to any of six output lines. The editor enters the reel number, track(s) number, the stripe assignment number, and if recording is to start at a point other than the beginning the sound sequence number is entered. Recording begins when the external equipment is interlocked (up to speed and in synch) and SMPTE time code agreement is reached.

Update Library

This program is used to load new sounds into the sound library. A sound can be input from any source (synchronized or not) and after being digitized it is saved temporarily on the system pack. It then can be played back and checked for quality, level, and then edited for length (usually several seconds is sufficient for most sounds). when satisfied with the sound the editor enters they key ID and description and specifies the library pack the sound will be stored on. The program checks the sound key ID for duplication and verifies that there is enough space on the specified library pack. The sound data is then transferred to that pack and the master library index is updated and the key entered into the ISAM table.

List

This program enables the editor to display and/or print out numerous lists of information in a variety of formats (alphabetic or numeric order, time code sequence, etc.,). Some of these are the contents of the sound library, modified sound library, showpack, show history pack, and track contents. Cue sheets with time ordered entries and 35mm feet/frames references for mixers and lay-off sheets for sound technicians are also provided.

SUMMARY

ACCESS was installed at Neiman-Tillar Associates, 8304 Beverly Blvd., Los Angeles, in January 1977 and has been used to create sound effects tracks for many feature films and TV shows. ACCESS has enabled the editor to expand and utilize his creative abilities as well as increase production output fivefold. The instantaneous availability of sounds and the electronic editing capability results in a tremendous time savings. Software development continues to enable dialog clean-up and music editing to be performed on ACCESS. The second system is almost completed (pre-production model). ACCESS has become the biggest technical advance in the Post-Production sound editing field over the past 50 years.

COMPUVOX™ :

VERY LOW COST VOICE INPUT FOR HOME COMPUTERS

Bill Georgiou
MICROSIGNAL, Box 161988, Sacramento CA

Abstract

Very low cost voice input to computers can be achieved using a novel approach based on sentence rather than word recognition. A sentence recognizer discriminates between sentences on the basis of the number and duration of words within a sentence. The actual meaning of the words is not important allowing the user to chose the most descriptive words for a given situation without limited vocabulary constraints. An application where COMPUVOX performs very well is motion control as in interactive video games. In addition it can be used as a low cost readily available sound interface to computers for a variety of applications not related to speech.

Introduction

The natural means of communication for people is speech. It is only natural to want to use speech to communicate wit computers as well. Unfortunately it is fantastically difficult to get a computer to decode the acoustic speech signal as a human does. Thus computers so far have mostly keyboards.

It is possible however, (and within the limits of present technology) to recognize words pronounced carefully when they are drawn from a small vocabulary and the machine is trained to the voice of the speaker. Commercially available industrial speech recognizers can do very well with a 16 word vocabulary and cost in the range of $4000 to $70000. Besides their relatively small vocabulary they are limited in that they require that:

1. The recognizer is trained to the voice of the user.
2. The user is cooperative and tries his best to help the machine understand him.
3. The words are spoken in isolation, i.e. they do not run together as is usual in normal speech.

Despite these limitations they have been found very useful in many industrial situations.

Two companies, Heuristics and Phonics, manufacture recognizers aimed at the hobby computer market in the price range of $300 to $500, offering lower performance at much lower cost.

With the introduction of the low cost "personal" or "home" computer systems such as the Commodore PET or the Radio Shack TRS-80, a new vast market for computer voice input opens which is very sensitive to cost and which is heavily applications oriented.

COMPUVOX is aimed at this market and provides most of the performance of a speech recognizer in a package that simply plugs into the "home" computer and is supported by ready to use applications software at a very low price.

Sentence Recognition: A Different Approach

Speech recognizers operate by discriminating between words on the basis of their acoustic attributes. Due to the variability of the speech signal, acceptable recognition accuracy can be obtained only by an elaborate software/hardware package which tends to be expensive.

Microsignal, in an effort to reduce costs and improve performance of voice input to the point it is acceptable to the growing personal computer market chose a totaly different approach to computer voice input.

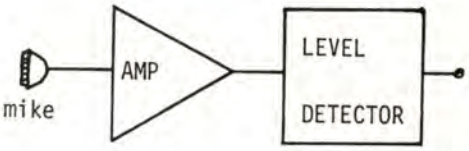
Rather than recognize individual words, COMPUVOX recognizes sentences. Not the meaning of the sentence which is a formidable task even for state of the art speech recognizers, but its structure. If we view a sentence as a sequence of words of various durations, then we can discriminate between sentences on the basis of the number of words within the sentence. Additionally, if we can reliably detect the length of the words, our discrimination capability is further increased.

Sentence recognition offers many advantages over speech recognition. First it requires simple hardware and software, thus it is inexpensive. Second it is speaker independent, i.e. the recognizer does not have to be trained to a given speaker. Training is undesirable because it is time consuming and boring and because if it is not done properly it will reduce recognition accuracy.

Third, sentence recognition allows freedom of expression. The contents of a sentence can be made to describe the function to be controlled without the restriction of a limited vocabulary. For example, the sentence "go west" is equivalent to "go left" or "go back" or "retreat now". In a given situation one of these may be more appropriate than others and the sentence recognizer allows for the best one to be used.

The Implementation of COMPUVOX

The COMPUVOX hardware consists of a microphone followed by an amplifier and a level detector as shown in the following diagram:



Its output is a TTL level signal which is "0" if no sound is present and "1" if a sound is present. The level detector is optimized for speech detection and the sensitivity of the system is adjusted so that it rejects common background noises such as people talking, the TV playing etc. Further noise reduction is achieved by requiring that COMPUVOX is held within 2" of the mouth while speaking to it.

All the hardware and the microphone are packaged within a small case, approximately 3"x2"x1" in size. A 3-wire cord is connecting COMPUVOX to the computer. One wire carries the signal, the other the +5V supply and the third is the ground. Power consumption is only 25mW and the power is supplied by the host computer since both the PET and the TRS-80 provide a +5 volt line for peripherals.

The software driver for the COMPUVOX performs the following operations:

1. Reject noise clicks.
2. Detect words.
3. Reject dropouts.
4. Detect end of sentence.

In effect the driver is acting as an intelligent filter tuned to the characteristics of speech. Noise and dropout rejection is very important because without them the system is too unpredictable to be useful. The filtering parameters have been adjusted after tedious experimentation to be a best compromise over the population of users. Fine tuning to individual voices is possible for increased performance. This is easily done in software.

The Counter and Duration Modes

COMPUVOX can operate in basically two modes: As a word counter and as a duration detector. Combinations of these modes are possible.

The information extracted in the counter mode is the number of words in the sentence. This can be utilized in a number of ways. To illustrate them, let us assume a motion control situation. This can be an actual vehicle under voice control or simulated movement as in the case of a video game. Direction control can be achieved with the following convention:

1 word : Go South
2 words: Go West
3 words: Go North
4 words: Go East

Thus if we want the vehicle to move south we say "South". To go north we may say "go north now" or perhaps "north you go!" if that is more appropriate. If more than four words are detected we have the option of ignoring the sentence or stopping. In the latter case we implement start/stop control as well.

Many variations are possible. To move faster we could say a sentence that has the same result result as the present movement. For example, we can arrange our program so that if we are going west and say "go faster" an increase in speed will result with no change in direction.

Duration mode operation entails detection of word length. For reliable operation discrimination on the basis of word length should be attempted only on word pairs with widely different lengths. For example, discriminating between "start" and "stop" on the basis of word length will give poor results. Very satisfactory operation is possible if "start" is replaced by "initiate" for example.

Applications

The limitations of both the word recognizers and COMPUVOX and the restrictions they impose on the user make selection and structuring of the applications environment critical for success.

The typical demonstration programs that come with the hobby speech recognizers allow the user to train the machine to his voice and then go through the vocabulary printing the words recognized. While this generates attention and interest when one is first exposed to it, its novelty dies off quickly. The next step to integrate the recognizer in an applications environment is left to the user and it is not usually taken. This results in an expensive recognizer collecting dust in the card cage of the computer. Companies that sell industrial recognizers often spend considerable amounts of money to identify and develop applications software for their customers. This accounts in part for the high price tags of their products.

The low cost personal computer market to which COMPUVOX is addressed is even more sensitive to applications than the hobby or industrial markets. Immediate and tangible uses of voice input are required in a plug-in, ready to go package.

COMPUVOX has been designed from the outset with an applications outlook and this is reflected in the support software. The first applications area it addresses is video games. It is well known that the majority of personal computers spend most of their computing time running games. It is also obvious that the most successful games are graphics and action oriented. COMPUVOX can be used to control motion by voice in these games, much like a super joystic. A number of such voice controlled games are offered by Microsignal. They require various skill levels which can be selected by the user. Due to the unique nature of voice input and particularly COMPUVOX, many variations of the actual playing of a game are possible with the same program. For example, a restriction can be placed that no two sentences contain the same words, making the player

search for synonyms. Or the games can be played in a foreign language one is learning.

This points to the potential of COMPUVOX in the classroom, where it can help introduce learning into playing. In addition, by forcing the user to speak clearly and at an even pace and volume, it may be useful in improving the articulation skills of preschool children. There is also the possibility it might be of help in training the deaf, since it provides a visual representation of sound.

Due to its low cost, COMPUVOX is valuable to anyone that is interested in using a speech recognizer, but is not very familiar with voice input. It can be used as a starting point to gain experience with voice input. Although not a recognizer, it has many advantages and limitations common to them, and can provide helpful insight in the pros and cons of voice input. And suprisingly, its performance equals or exceeds recognizer performance in some applications.

Questionnaire answering is another area where COMPUVOX can be useful. By using the duration mode, discrimination between "negative" and "yes" is possible. These two words can be used to answer questions posed by the computer. By carefully phrasing questions so that only a yes/no answer is possible, extensive "conversations" with the machine are possible. Going to a mixed duration and counter mode, "yes", "negative" and "maybe yes" could be used enhancing the choices of the user.

Non-Speech Applications

The sound detecting capabilities of COMPUVOX make it useful in a number of sound detection applications not involving speech. As one example, take the case of phone call logging while the user is away from home. By placing COMPUVOX next to the phone, the time the phone rang and the number of rings can be recorded. This does not require any connection to the Phone Company equipment. Noise rejection can be excellent by a short program that detects the phone rings on the basis of their duration, which is fixed.

Spacing between sounds can be accurately determined, such as the time between firing a firearm, and the bullet hitting a surface, or in the case of machinery that produces rhythmic sounds. Or the period between ticks of an old clock can be determined to adjust its timing. The times a baby cries during the night can be displayed in the morning for the parents to see if COMPUVOX is placed during the night. Or, in general, activity of people or animals can be logged on the basis of the sounds it produces.

Conclusion

By using sentence rather than speech recognition, low-cost voice input is possible for home computers. A number of applications and programs addressing the computer video game field offer a concrete basis for demonstrating and using this mode of voice input. This, however, only scratches the surface of possible applications that range from voice input to sound detection and measurement. Rather than leaving the development of these applications to the user as it has been customary, Microsignal is breaking new grounds in the field, by an actual commitment to provide the user with a wide variety of applications software.

UNIQUE PERSONAL COMPUTING APPLICATIONS FOR ATTORNEYS AND COURT REPORTERS

by

Douglas W. Du Brul, BSEE
5681 Mary Lane Drive
San Diego, California 92115
Phone: 714/583-3733

This paper presents a review and update on two unique personal computing applications for attorneys and court reporters. In addition to the applications which each can use independently, there are several which require interface compatibility across the attorney/court reporter interface.

It is rather obvious that all small offices may benefit from using small computers for accounting, scheduling of appointments and run-of-the-mill word processing. I will limit my discussion to two unique and lesser known applications: computer-aided transcription and key-word indexing of court and deposition transcripts.

Computer-aided transcription (CAT) involves computerized translation of machine shorthand (stenotype) notes to plain language. At first glance this application seems to benefit only the court reporter but actually it also benefits the attorney by reducing the delay between the time testimony is given and delivery of the completed transcript.

An overview of the basic CAT process is contained in a paper I presented at the Personal Computing Festival held in conjunction with the 1978 National Computer Conference. It is included in the Personal Computing Digest (1).

To the best of my knowledge CAT software for personal computers is not available to the general public yet. At the present time CAT services are available from several firms under a variety of arrangements. One firm will lease a complete stand-alone system which is capable of performing all of the processing required. Another firm does the required computer processing at a remote host computer.

Recently a new CAT system was introduced by Xscribe, Inc. of San Diego, California. Xscribe is selling a reasonably priced CRT text-editing station (about $8500 including a printer) which is designed specifically for rapid editing of the rough plain language output of their CAT computer. They provide the actual shorthand-to-plain language conversion on their equipment. A unique feature of their system is that it reads stenotype notes optically, thus eliminating the need for having a digital recorder attached to the stenotype machine. The rough output is delivered on a mini diskette.

The Xscribe editing station is actually a microcomputer. I would like to see its capability increased to include many of the functions currently possible with general purpose microcomputers.

KEY-WORD INDEXING

A key-word index, as its name implies, is an index to the location of certain key words and/or phrases. Key-word indexing can be a valuable asset to an attorney, especially on large cases involving thousands of pages of transcripts.

It has been rather common practice to have court reporters prepare a transcript and deliver it to a law firm which would then have a computer operator copy or "rekeyboard" the transcript at a computer terminal in order to place it in memory so it could be searched for key words and phrases. Alternatively, the court reporter might be asked to have the transcript typed with a special OCR type font so that the transcript could be read into a computer with an optical scanner.

Both of these approaches are

expensive since one involves the use of costly scanning equipment and the other involves having to type the same information twice. Small word-processing systems can be used to produce hard copy and floppy disk records with one typing operation. Once the floppy disk record has been made, it can be used to create key-word indexes.

The actual key-word indexing can be done by the court reporting firm or, alternatively, the reporter could deliver floppy disks to a law firm wishing to do their own searching. If the court reporter does the key-word indexing, the law firm could avoid the need to purchase computing equipment. However, the law firm will reduce the possibility of compromising strategy if they do their own indexing. Obviously, the two computers will have to interface correctly.

Court reporters frequently have typing done by independent contractors who work at home. It may develop that these "transcribers" will have to acquire microcomputer-based word-processing systems so that they will be able to capture their keystrokes on floppy disks when there is a need for key-word indexing. Their systems will not have to have key-word searching software since the actual indexing will be done on the attorney's or court reporter's computer.

Many of the available word-processing software programs lack several features which are important in key-word searching. For instance, most number pages during the printing operation. The numbers appear on the hard copy but not in the digital record. This makes it difficult to identify the location of a key word after it has been found.

There are at least two ways to identify page numbers (and preferably line numbers) when doing key-word searching. One is to "reformat" the record to add page and line numbers after the data has been entered as continuous scroll. Alternatively, a running count of lines can be maintained during the search operation. A line count can be converted easily to a page and line notation when a key word location is needed.

Care must be exercised when making searches for key words. The computer will recognize a given key word only if it occurs exactly as typed in by the operator. To the computer a given word may appear as four different words depending on whether it has an initial capital letter (as at the start of a sentence), all upper case, all lower case or if it is hyphenated. Unless a sophisticated program is used, it is necessary to search for each form separately.

References:

(1) "Computerized Shorthand", NCC '78 Personal Computing Digest, pp 147.

TWO YEARS BEFORE THE MASTHEAD or
WRITE THE TEXT EDITOR, THEN THE TEXT or
HOW THE COMPUTER MADE ME INTO A WRITER

Jef Raskin
Apple Computer Inc.
1Ø26Ø Bandley Drive
Cupertino CA 95Ø14

September, 1978

This is an account of the intimate interaction between a human being and his personal computers. These computers changed his life, helped him perfect a new skill, and gave him a bit of pleasure on the way. The person is me, and I sit before my text-editing system, sending these words into the computer, to appear ephemerally on a screen. I hesitate, quibble over choice of words, re-arrange the text, make errors and correct them, and give nary a thought to the complex process that goes on hidden beneath the keyboard. Later, my computer will print these words on hard, enduring paper. In the meantime, I have done far less work to produce these words than people would have imagined just a decade or two ago.

My word-processing computer has become as much a part of my life as my car, my telephone, my house, or my electronic calculator. It would be more devastating to my life style and goals to lose the computer than it would be to lose any of these other material objects.

The computer industry, which is going gung ho these days, was not doing quite so well in 1975. In face, it was having a small recession. I had a good background in computer science, but things didn't look too interesting in the industry. Programming jobs were available mostly in Business Data Processing, Operating Systems and Military Projects. I had already done a stint as a professor teaching programming at U. C. San Diego, and having left the golden shores of Southern California was ekeing out a living as a music teacher in San Francisco.

I had pretty much written off the computer industry as a case of infantile senility. It seemed to have both immaturity--no one seemed to know what anyone else was doing, and wheels were being re-invented all the time--and at the same time was already ossifying and becoming dogmatic. Microprocessors (and large scale integrated circuits in general) were injected into the middle of this depressing scene: they performed an instant cure. The industry was off and running again. My personal barometer to the well-being of the computer industry--the thickness of Datamation Magazine--began to rise. Thus, as soon as the Altair 88ØØ became available I took my meagre earnings as a music teacher, and with a beloved friend (and his money too) we ran off and got one.

The computer worked. The process wasn't easy, and it didn't happen right away, but eventually the computer worked.

We then got an IMSAI, a POLY 88, an APPLE I (etc.), and ran amuck with the variety of boards and accessories available. The main use of these computers was to learn about microcomputers. They weren't much good for anything else. Too little software, terrible documentation, and occasionally marginal operation. I remember getting appreciative laughter and applause when I announced at a computer club meeting that my computer had become so reliable I had been running it for nearly a week with the cover bolted down.

At this point, a mission began forming itself in my mind. The manufacturers were playing fast and loose with their customers. Parts were missing or wrongly labelled, you had to have inside information from the factory to make things work, and above all, the instructions were inadequate. My first attack on this problem was to find a wonderful little rag called Dr. Dobb's Journal, which didn't accept any advertising, and to write articles telling the exact, documented truth about the various products that I had run across.

Dr. Dobb's let me do this without editorial restriction, and soon I was their equipment reviewer. Since I was strictly truthful, there was no potential problem with lawsuits. In fact, there were none. The best part was that I got lots of equipment to review. Some I could even keep. Soon I was writing quite a bit, and not getting a cent for it. Remember, magazines that don't carry advertising aren't likely to have enough money to pay writers. On the other hand, I had broken in to the writing game.

But producing articles is not much fun. Testing equipment and having opinions on this and that is easy. Learning to write down your opinions in cogent form is a bit more difficult, but it comes to you with practice. It was the mechanical effort of getting my written opinions neatly typed that stymied me. I never did like working with a secretary. I just don't feel comfortable fobbing off on someone else something I'd rather not do myself. Especially dull, potentially unrewarding work like typing. On the other hand, I have no qualms about having a computer do dull, repetitive jobs. My computers never even whimper when asked to do work that would soon reduce me to a quivering wreck.

The obvious answer was to have the computer be my secretary. This is what I did. I took my last remnant of savings and borrowed a bit besides, and with reckless abandon I purchased the best quality printer I could find. It happened to be a Qume. I wanted to write articles and books on many subjects, and I didn't want the copy to look as if a computer had done it. I wanted it to look like perfectly typed copy.

Soon, as in a fairy tale, everything was going fine. Every article I wrote was promptly sold. Part of the reason, I am sure, is that they all looked so presentable. Each one was cleanly typed, with neat margins, proper pagination and no corrections. The computer made it possible for me, without excessive effort, to sit down and edit carefully. I could change spellings, re-arrange sentences, add ideas after the article had been written--all without the penalty of having to retype everything.

The single most important blessing that the text-editing system conferred was not speed, nor was it convenience. The most important benefit of the system was an improvement in quality. My writing was, in fact, a lot better for my using the text editor. I didn't have a master-slave relationship with it: it was a true symbiosis. Or, perhaps, I was its parasite, living off its strengths.

From using the text editor, I learned what features I really needed. Since I was the program's creator as well as its user, I could bend the program to my needs. Eventually, the program settled down and I formed a small company to write documentation, attempting to strengthen what I perceived as the weakest link in the personal computer industry. But that is another story.

ORGANIZING A LOCAL GROUP OF COMPUTER USERS

Douglas J. Mecham, Hughes Aircraft Company, Ground Systems Group,
P.O. Box 3310, Fullerton, California 92634

Abstract

You may wonder just how your friends get so much software or learn so much about computers without formal training. Perhaps you have noticed how technical articles in the trade journals seem to build upon previous articles. The primary active element involved in this phenomenon is a sharing of computer information among users. Often times the exchange of ideas is done at a meeting of involved users. This paper will present some of the aspects of forming a group of computer enthusiasts to share their experiences regarding computers. Such a group will provide you with a wealth of information about computers within a friendly social environment.

A Simple Approach

Undoubtedly you have one or more friends that you confer with over your computer system problems. If three or four of your computer friends have met to listen to one person explain his concept of a computer topic you have the beginnings of a local users group. Following the presentation your small group may have some refreshments before breaking up. Most likely you will return home with a new piece of software or computer technique to try out. Let us identify the basic aspects of such a users' meeting:

Meeting Arrangements--one of the group called the others and invited them to join him at his home.

Meeting Opening--The objective of getting together was, in some fashion, stated and any new persons were introduced.

Technical Presentation--One of the group explained how he had interfaced a new device and explained the associated software to control it. As the presentation progressed questions were asked and comments made by the others in the group. During the presentation reproduced copies of the device interface description and software listing were handed to the group.

Computer Products--Since each day a new product for the home computer market is introduced, one of the group discussed his research of the new product that may be useful to the others.

Social Hour--After all the involved technical material the group relaxed while the host served refreshments.

Next Time--Because this group was so involved with their computer hobby they decided to meet again to share more information.

As you can tell by the description a local users group meeting took place--simply and easily.

Needless to say with so many computer hobbyists around such a small meeting could rapidly grow. The basic meeting elements will not change but a little more planning and organizing is required. A meeting of computer users need not constitute an official group since user interface may be primarily through a newsletter subscription, or a combination of both. The following paragraphs present the considerations involved in organizing a group of local computer users.

Orientation

The objective of such a computer users meeting is to share one's own knowledge and experiences of computer systems with others who may not yet understand the system. This, of course, provides self-esteem, a key psychological requirement in a group. Also, such sharing certainly provokes others to share their knowledge and experiences. Additionally the presentation of two items of information often times results in a synthesis providing a third and different item of information. For example, development of a syntax analysis routine along with a dictionary routine might be put together to build a compiler.

The collective of computer users can often provide the resources necessary to solve a problem not easily solved by an individual. For instance, one user may use a particular type of terminal and another user a particular CPU which when interfaced could determine the solution to a data communications problem. Or, you may just need the ideas from several other enthusiasts to formulate a solution to your problem.

→ [A users group does not make sense without user involvement]

Likewise, individuals in the group may be able to contribute a mail listing program for distribution of mail to the group members. Economically, the formation of a group of users makes sense because products may be purchased by a group cheaper and a single software purchase may be shared among the group. An objective of a group may include one or more services such as publication and

distribution of a newsletter and/or maintenance of a software library. In particular, a formalized users group can often arrange for technical experts to make presentations to the group as well as sponsor training seminars.

Thus the formal orientation of a group of users is any objective which, by definition, must meet their needs. Before I describe the technical aspects of organizing a group of users let me reflect on the individual user who would make up such a group.

The User

The objective for most users of hobbyist computer systems is to solve their problems and enhance their software/hardware to perform more complicated tasks. The success of a users group is dependent upon understanding the member user's perspective; the perspective is to solve his problem in a simple expedient fashion. If a users group can meet this requirement both the user and the group will be a success.

The contribution a user makes to a users group depends upon his participation. Certainly a user's expertese and experience will allow him to participate easily in technical functions. For the novice user participation may bbb in the areas of administration and support of the group functions, such as maintaining mailing lists. Together a users group can function. The novices will advance to the experienced levels, the experienced to bigger systems, and new users will join at the novice stage.

The most important user attribute to pay attention to in a group is the user's attitude Since most groups of users depend upon volunteers to keep the group active particular attention needs to be paid to keeping the atmosphere around the user's involvement positive and self-satisfying. The dedication of a user varies over a wide range as does his sensibilities. Recognizing these traits and managing them can be an asset. For the most part this involves common sense, common sense, common courtesy, being observant, and giving recognition.

A user's expectation is easier to meet in a volunteer organization than in a commercial organization. In the former the user is usually intimately involved and the destiny of his expectations is a function of his own effort. In the latter the user always expects more when he is paying money and is not involved in the "product".

Watch out for users who promise a great deal but contribute little. It is difficult to prevent users from taking advantage of a volunteer group for their own ends; on the other hand, there are most likely one or two individuals who contribute a great deal. While it is impossible to satisfy the latter's desires or repay them, public recognition is most fulfilling. Psychological satisfaction is often an important and satisfying reward.

→ [Apathy is the worse enemy of a volunteer group. Enthusiasn is a group's greatest asset.]

Now that some awareness of individual users has been brought out and objectives made clear where does one start to organize a group of computer users?

Where/How to Start

Why organize? It would be nice if the interface of all people just fit together to arrive at the desired goal. Since this is not the case in the real world someone needs to plan the sequence of events. Any organization is predicated on an agreed-upon group objective. This may come about through an informal conversation with a few friends with a common computer interest who decide to interface among themselves in the future. The user interface may manifest itself through a meeting, newsletter, software library, or a combination of each. The user meeting is one of the easiest to start with since people like to socialize, especially around a hobby. The particular subject matter may be around a special interest such as word processing or a particular vendor product, or both.

An effective method is to get a few "key" friends involved who you can count on to at least carry out an initial meeting of users. Make sure the communications between the key group is free and easy since a time lag and a hinderance of communications can postpone activity and dampen any enthusiasm. The next item of business is to establish communications with other users.

A first meeting in a home or local school classroom facilitates easy arrangements and control. By organizing the basic meeting aspects described in the opening of this presentation your meeting will be a success. Depending upon how ambitious the key group is the meeting advertisement may be by word of mouth or a formal, mailed, invitation.

If the meeting sparks some interest and a definable need to interface among users in the future exists then a more formal group may be organized, although several meetings may take place prior to serious organization of a group. A key to a successful first meeting is to arrange for all attendees to return home a "winner," that is, make a contribution to the user's future success. Such a "prize" may be a software program or inside information on a vendor's product. Before the first meeting has adjourned the next meeting or user interface mechanism needs to be planned and volunteers designated who will implement it, thus the next "key group" is formed. Note, if there are no volunteers either abort the group organization or be prepared to perform a GREAT deal of work.

For the first meeting keep expenditures and meeting operations to a minimum. To offset any dollar costs collect a donation at the door. An alternative is to find a sponsor such as a

Product manufacturer or retail store whose interests are served by such a meeting. Most likely both seed money and manpower for the first meeting will come from the "key" group. Thereafter a more formal mechanism for funding and incentive for manpower needs to be found. Some ideas will be discussed later.

Communications

During the formation and subsequent operation of a users group the most important element is communication. If those in the key group do not relate their problems and successes plans are difficult to make, let alone adhere to. Likewise constitutants of the group probably will not respond if they do not receive some form of communication from the key group. The greater the individual member communication the greater chance for group success.

The information that must flow in a technical organization is tips, techniques, hints, kinks, along with meeting times and places. Communication needs to take place as often as necessary. This means if you communicate while you are thinking about it the link is made and progress can take the next step. A number of simple and easy acts of communication can build an effective network.

→ [Lack of effective communication has been the downfall of many groups.]

Different requirements dictate different methods. For instance, the post card or note is easy and quick especially if mailing labels are already printed. Letters are more formal but do show more thought for a professional approach and they serve as a good audit trail so new ideas can build on past actions.

The newsletter of course is a more formal but easy eay to communicate with a number of users. However, a newsletter requires a dedicated volunteer and considerable user contribution. Special publications are very useful since they are specific in nature and usually treat technical material in-depth. Such a publication results from the dedication of a very few users but unlike the periodic newsletter it is usually a one-time task. Both of these methods of communication can be very effective and simple to produce. These publications may, however, be quite formal and require a complex publishing operation. The most effective newsletter/publication is not necessarily the "slickest," easy and simple techniques for production of these items is found in the literature [1].

The telephone is quick and easy for most users. While it is difficult to communicate many details over the telephone this instrument is very effective for directing a group of users by the key group and receiving status on activity. An automatic answering device can provide an easy method to communicate to the members and collect brief comments; such a device can be made available for communication 24 hours a day, unattended.

If the expectations of member users are to be met then mechanisms for user feed back need to be devised. This may take the form of questionnaire or telephone campaign. Keep in mind that talleying responses and performing an analysis on the results may overwhelm the uninitiated; so keep responses very simple. Also, keep in mind that the number and accuracy of responses is directly related to the complexity of the questionnaire. Again, keep the questionnaire short and simple. It is better to have several short questionnaires than one long one.

If your group warrants it, effective communication may be supported by clerical assistance; it is well worth hiring a professional secretary part time for typing tasks. Of course, micro computers, simple printers, and an elementary text editing program can serve as a secretary (except for spelling).

Handling the information flow and keeping it moving is important since newsletter editors thrive on new items and the key group needs the information to formulate plans to support the group. A central address and desk is useful for this purpose. Information must be appropriately redistributed in a timely fashion as well as just accepted. The formal information flow by mass mailings requires some serious thought. The key member responsible for mailing needs to be aware of the considerable effort required to stuff, seal, address, stamp, and mail a large set of letters not to mention printing. The cost of mailing can be optimized by planning the weight just below a postage price break and photo reduction techniques.

With the advent of word/text processing on micro computers the tasks of producing mailing labels, letters, etc. are easy. Even computer data communications can be used most effectively to get users to contribute articles; it is easy for a computer system editor to "type" an article.

The Users Meeting

Computer users need to get together to discuss their successes and problems of their systems. In most instances a local group can have a very successful but simple meeting by considering the items discussed in the introduction of this presentation. Consider the aspects of a more organized meeting; these concepts apply to the small simple meeting where appropriate.

There needs to be a central managing committee to insure consistency and follow-through before and after the users meeting. Be very aware of the effort required by each committee member; each member should commit themselves in a formal manner as a warantee on didication to complete the job. Make sure

all meeting tasks are well defined and assigned. The magnitude of each task needs to be evaluated in light of the committment made by a committee member to perform it.

→ [Large meetings supported by weak committees can turn a meeting to chaos.]

Smaller informal user meetings can take place often at convenient times such as over lunch or in the evening. However, the larger full day meetings require planning a day when there is a minimum of conflicts with member users. Multiple day meetings should be avoided due to accommodation problems, competition for a member's time, and user saturation levels. The location of a meeting should be central to the group; however, special locations are attractive. Depending upon the budget and nature of the gathering meetings are easily arranged in hotels, schools, homes, or vendor facilities. The city to hold a users meeting is most likely obvious for a local group of users.

Economics are always difficult for local groups since "seed" money is often lacking; asking for pre-registration donations, fees or vendor contributions can be effective. Club projects to raise money are also effective. The larger the meeting the more difficult to budget since all cost factors must be analyzed and scrutinized since profit margins are very narrow. A simple meeting fee is to have each member contribute a fixed amount towards a single budget item such as a meal/refreshments.

Since the computer field is expanding at a phenomenal rate there are a great number of resources for technical presentations. The simplest and easiest is the group membership; many computer hobbyists have a speciality topic he/she could discuss. The meeting committee can be very helpful in encouraging a user speaker and helping him prepare. Vendors of audio visual equipment often have booklets on making technical presentations. Vendors and universities are two good sources for speakers. Keep in mind any audio visual support required by speakers. It is often enlightening and fun to have a special speaker during the meeting talk about a non-computer related topic; it relaxes the minds of the listeners. Be sure to recognize the speakers in some manner; a free meal ticket is usually easy while a speaker's gift is nice. In any case send a letter to each speaker before and after the meeting to give him meeting particulars and thank him. A key to a successful meeting is to give sufficient lead time for users to plan to come and speakers to prepare; plenty of mailings help. Each potential attendee should be made to feel that if he/she comes to the meeting they will return home a winner and a greater success.

Proceedings and technical papers require a definite dedication on the part of a meeting committee. Remember each submittal needs to be edited and evaluated not to mention all the mechanics necessary to publish them. For local group meetings a simple Xerox reproduction or offset printing will suffice for hand outs at the meeting. Attendees seem to like even the briefest of descriptions on paper.

Vendors can often be a great source of support for a users meeting. They may even be willing to sponsor or underwrite costs, refreshments, facilities, ... provided they can display their wares. This approach works even on a small scale by contacting a vendor salesman.

Software Library

All computer users like to get a hold of another user's software contribution but often does not have time to contribute himself. An easy rule for this situation is, tit-for-tat, only those who contribute have access to the other contributions.

The form and format of a software contribution must be simple and easy but consistent so a user can easily compare and search for entries. The quality of contributions is difficult to maintain but if the user has documented his source code well, and provided a working example and a simple user's guide then the recipient should be delighted. Assuring quality is very time consuming but a minimum quality can be maintained if the examples can be proven to work easily.

Collection and distribution of software library entires needs serious thought. The media is the first consideration (sloppy, paper tape, etc.), organization of supplementary documentation second, and third (but most difficult) how to get the job of reproduction/distribution done. There are many approaches but the more the contributing user does formatting and testing his contribution the easier to collect and organize the software. Indexing a software library is not easy but a permuted title index is one easy method, provide each title is meaningful.

Managing and Administration

For small local groups this aspect is relatively easy since the atmosphere is informal and activities are minimum. As the group grows a serious effort on the part of the leaders needs to be put towards the business aspects, i.e., the transition from a "club" to a "business" is considerable.

Two key considerations for leaders of the group are:

1. How to best meet the needs and expectations of the users.
2. Volunteerism and getting the tasks done.

When hobbyists become involved and passionate towards a group they may tend to show emotion and be sensitive to personal reactions. That is, leaders need to pay attention to be sensitive to personal politics enough to make

most of the users successful most of the time.

→[Cater to active enthusiasts since they are the heart of a users group] Unfortunately the family related to the hobbyist may suffer and account should be taken of this fact and some recognition or activity planned to involve them. An individual's contribution to the group can be adversely effected by his family.

Administration of a larger users group requires establishing a well-defined objective, a guideline for operation, and specification of particular services to users for particular prices.

Summary

Becoming involved in a users group is a learning experience. You must realize that while you will gain a great deal of information your rewards will never match your contributions. There is, however, a great latitude for self satisfaction. The alternative to involvement is stagnation, an unchallenging position. If you do not belong to a group of users then join one or form your own, it is easy and simple.

Acknowledgements

Several key people have assisted my development with computer groups. The first is Alan Mitchell of Tandem Corporation who was co-founder of the HP3000 Users Group. Bill Bryden of Inland Systems Engineering has assisted me in developing several users group meetings, local and international. A great deal of reflection upon my ideas was done by Richard Nelson, Editor, PPC Journal. My thanks to them for supporting my efforts.

References

1. Nelson, Richard J.; Editing and Publishing A Club Newsletter, Second West Coast Computer Faire Conference Proceedings, April 1977, page 125.

2. After Proposition 13, Volunteers Needed, Time, August 7, 1978, p. 34.

3. Proceedings, Southern California Regional Users Group Meeting, March 1 and 2, 1978, Douglas J. Mecham, Chairman.

4. Mecham, Douglas J., Founder and First President, HP3000 Users Group, 1974-1976.

Your Computer May Speak, But Can You?

Jeffrey R. Wise, Student, CSU Chico
Bradley Hall Box 899
Chico, Ca 95926

Abstract

As the popularity of small computers increases more people will want to know about them, in various levels of technicality. You, as a relative expert on the subject, will be expected to share your knowledge of hardware, software, or applications. This paper will attempt to ease the fear that many people have about speaking by covering topics on the voice, delivery, gestures, stage fright, speech organization, wording and style, visual aids, how to listen, audience analysis, how to detect boredom, and how to correct the situation.

The Voice

The voice is the primary means communications in interpersonal situations. In small groups, you speak and are not self-conscience of your speaking, it is only when you stand up before large groups that you become nervous. Is my voice pitched too high or too low? Is it loud enough? Am I going too fast or too slow? Is my voice grating on their nerves? These and others are understandable and will be answered soon.

To study your own voice, it will be necessary to use a tape recorder. Pick a selection you like and read it while recording. On finishing, play it back. Believe what you hear. It will not sound like what you think your voice sounds like because you hear a mix of your true voice and vibrations from your vocal folds by way of bone conduction.

Now that you have listened to yourself, there are two items to look for concerning pitch. Number one, is it within normal limits? For a male it is C below middle C. For women, it is G sharp below middle C. The range is plus or minus two or three notes.

The next thing to listen for is the inflection of the voice. The three types of inflection are the rising tone, falling tone,and the circumflex. The rising inflection shows question or suprise; Really? You are? Falling inflection shows certainty, finality, and other assertive emotions. Circumflex inflections have patterns of rise-fall-rise or fall-rise-fall, showing doubt and hesitation.

I admit that inflection is not all that easily taken care of. A skilled speaker may take a given phrase and make it mean anything he wants.

It is likely that you have better insight into your voices volume than its pitch. Maybe you have been told that you have a vigorous or boomming voice or that your voice is so soft that people have to strain to hear it. If you are not sure that the people in the back row can hear you then ask them. If your voice is not heard then speak so loudly that it seems as thou you are shouting. The secret to this is to speak from diaphragm. This means using the stomach, not the throat. Ideally, you should visit the room or hall that you are going to speak at and try it out, take it out for a spin, so to speak. Have a friend in the back to listen and advise you as to the volume of your voice.

Now that you have the volume needed, attention must be paid to the variation of the volume. Some ideas must be expressed with more vigor than others. All of you are capable of great volume, it just takes practice.

The duration or rythm of your words have a great impact on the way people listen to you. If there is one piece of advice that could be given to a new speaker, it would be:

SLOW DOWN!!

You should pause to let your audience adsorb what you have said, to take a deep breath, or simply to help you slow down. This also adds a great deal of emphasis to your ideas. Use those pauses.

The fourth aspect of the voice is the most difficult to describe. It is the quality of your voice and arises from the harmonics of the voice and can be called hoarse, harsh, shrill, nasal, dull, rasping, rich, or resonant, among other terms. If your voice is not as pleasing as you think it should be then one of the books that I have listed in the back of this paper.

Nervousness

All of you have probably heard, at one time or another, that all people, regardless of their experience, are nervous when they speak. This is true but it does not do you any good. What you need to know is the tricks of coping with this stage fright.

The first idea is to be prepared. The better prepared, the less you have to worry about. The second trick is to think about the ideas you are going to communicate, not about yourself. Third, be organized. Fourth, realize that you can succeed. The last and best idea is to practice, practice, and practice. That will help you more than anything.

Delivery

Delivery is the art of nonverbal communication and it is very important to the new speechmaker.

There are very few hard and fast rules for speaking and one of them is: Look the audience right in the collective eye. This has two effects. One, it adds sincerity to whatever you are saying. How many of you would buy a used car from a person that could not look you in the eyes? Even if you can not see the audience because of the lighting you know that they are out there. You can hear them breathing. The second thing about eye contact is that it allows you to watch for audience reactions and I will explain that later.

The face expresses quite alot of what we are really thinking. It is also seen very well by the audience and should be covered. It can be in one word: practice. It also helps if you believe what you are saying. If that is the case then the facial expression will take care of itself.

The body movements can be a powerful aid or detractor of the speaker, depending on how it is used. Body moves should be very limited. Do not pace, bob, weave, shift from foot to foot, fold your arms, fiddle with your glasses, or anything that calls attention to itself. On the other hand, do not freeze on the podium. You can take a step back or forward as you emphasize points. Turn your body as well as your head to look at other people. In essence, make only positive, purposeful movements.

The same can be said for the hands. Make your gesture and return your hands to your side. As for the other parts of your body, the rule holds. You can shrug your sholders, lift your eyebrows, and, in particular, smile. If it helps express what you want to say, do it.

Speech Organization

Speech, like Gaul, comes in three parts. The first type is fully written, as thou it is to be published. With such a speech, you have two ways of delivery. The first is memorization. This is alright for actors but not too good for public speakers. It sounds too stilted and the speaker is unable to adjust well to the situation. The second method is to read it. This has the same objections as the memorization but can be used if accuracy is a must, as in political or scientific speechs.

The second method is the expemporaneous. This is the method of having an outline and almost winging it by filling it out on the spot. This is highly recomended because it allows you to adjust to the reactions of the audience. Don't worry about freezing up on stage, practice and it will come naturally.

The last method is the impromptu. The thought of totally winging it, that is with no preperation. The only real use of this is in forensics tournaments.

Three is a magic number in speech and any of the above speech types can be seperated into three parts; the introduction, the body, and the conclusion.

It has been determined by studies that a speaker will lose his audience early in his speech if it does not catch on. The meaning is clear: The introduction must grab the interest of the group as

it progresses or the speaker will be lost. The second purpose of the introduction is getting on the good side of the audience. This involves establishing your credibility, removing misconceptions on the subject of the speech, and getting on a common ground with the listener.

Rules for the body are very few and vague. Basically the main ideas should follow a carefully considered order and you should repeat the key ideas often. This also means that you should not try to cover too many items unless they are easy to understand.

The conclusion should be a restatement of what you have said. In essence, say what you are going to say, say it, then say what you said.

Style and Wording

The basics of the style can be condensed into nine maxims:

1. The information must be relevant to the audience.
2. The information must be presented in a new light.
3. The information will be more readily adsorbed if presented in a startling manner.
4. Present the information in a humorous manner.
5. Associate the information with something the audience.
6. Relate the information visually, if possible.
7. Repeat the information by repeating it or restatement.
8. Present the information in an organized manner.
9. Do not ramble, that is be concise.

As for wording, you must be accurate, specific, concrete, and uncluttered. In essence, choose the words that present your ideas in the most clear manner with as little possibility for misinterpitation as possible.

You must not be too dry, thou, in your choice of words. Vivid speech begins with a vivid thought. These are hard to come by and there are two ways, that I know of, to develop vividness of speech. They are the verbal image, describing a picture in words, and the other is to express familiar ideas in an new way. This uses alternate meanings of a word and, like anything else, needs practice.

The choice of words and the expression of ideas, however, has one large guiding light: the audience. Analyze the audience, as in the later section, and you will be set in the major constraint of your speech.

Visual Aids

You have to be very careful with these. They have to be large enough to be seen by all in the room and seen well or they will ignore you. Another difficulity is for the speaker not to talk to the visual aid. Talk to the audience.

When the visual aid is unvailed, explain it. Tell the audience what it is when it is shown or the audience will try to figure it out and not listen to you. The same goes for handouts. When they are handed out, give everyone time to read or look it over before you resume speaking.

How to Listen

Listening uses energy. Sit up and pay attention. Don't slump in your chair or fold your hands like you are challenging the speaker to teach you something against your will. Be an active listener and let the speaker know that you understand him by nodding your head or smile when he makes a joke, and so forth.

You should listen to the items that have been covered already. Listen to the voice, delivery, gestures, organization of the speech, style and wording, and visual aids. Think of how you would do it. Watch the audience, see if it is bored and how the speaker corrects it, if at all. Watch and learn.

Audience Analysis

The level of technicality and the topic itself, if you get to choose it, will depend on who you are speaking to. The nature and extent of the audience's knowledge of the subject must be taken into account. This governs the technicality of the subject.

Interest of the audience in the subject is important in how you must approach the subject. Why

are they here to listen to you? Are they there by choice or by force? These questions will help you decide what approach will be the best.

There are many other factors in the analysis of an audience but they are not important in an informative speech. If, on the other hand, you are trying to convince someone of some ideas validity, you had better read up on pursuasion.

Boredom

Boredom, the bane of speakers, must be avoided, but first, how do you know that the audience is bored?

While you are keeping eye contact with the audience, look at what they are doing. Are their eyes on you or are they looking around? Have they stopped reacting? Are they mumbling? When you are bored, look into yourself and see what you are doing to show the speaker that you are bored. Once you know this, you can look for it in the audience.

Correcting this is not as easy as it is to detect it. Try to speed up or change your volume or tone. Toss in a joke, if you can think of one that is appropriate. If the speech is to go on for much longer you could let the audience get up and take a break for a few minutes. In essence, change whatever you are doing and it will help.

Conclusion

It is hoped that this has decreased some of the fears that you have about speaking. All that I can say is that you should go out and speak. Practice makes perfect in all fields, including speaking.

If you have no speaking appointments then write any speach on any topic of interest to you and deliver it to your tape recorder and watch yourself in the mirror. Or get Vital Speechs from your library and read them for practice and to look at the style. Study the masters and learn.

As for delivery, watch Billy Grahm or listen to other public speakers and practice. And good luck!

Refrences

Rudolph F. Verderber,
The Challenge of Effective Speaking
Belmont, Ca: Wadsworth Publishing

Loren Reid
Speaking Well
New York: McGraw-Hill

Georgiana Peacher, Ph.D.
How to Improve Your Speaking Voice
New York: Frederick Fell, Inc.

George Fluharty and Harold Ross
Public Speaking
New York: Barnes and Noble

USING FUTURES RESEARCH METHODS TO ASSESS POLICY IMPLICATIONS OF THE PERSONAL COMPUTER

Paul Gray
Graduate School of Business Administration
University of Southern California
Los Angeles, California 90007

Introduction

For a mature industry, such as steel or soap, it is often sufficient to extrapolate current trends to forecast the future. A sophisticated analysis may take into account foreseeable changes that could impact an industry, such as demographics. Such simplistic approaches are not adequate for a dynamic, emerging industry such as personal computing. In this paper, we will describe the approach being used at the University of Southern California to evaluate the future of the personal computing industry.

The approach has five components:

1. Market survey
2. Market model
3. Scenarios
4. Delphi Inquiry
5. Cross-Impact and Policy Analysis

We will discuss each of these in turn.

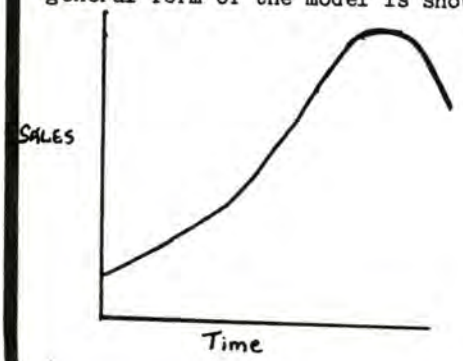
Market Survey

The market survey being undertaken is designed to measure the extent of the market for personal computers. The basic questions are designed to find out about the intentions of people: at what price level (if any) would they buy a machine? What will they use it for? How much effort (e.g., programming skill) are they willing to put into it? If they have a personal computer are they planning to upgrade it? replace it with a unit having greater capability? The approach of the market survey is fairly conventional.

The main use of the market survey is as input to the market model.

Market Model

The Bass model, developed in 1969 by Frank Bass at Purdue, is the basic model being used. This model is designed for predicting the sales of large-ticket consumer products that are purchased infrequently. The model has been validated for a whole range of consumer products such as washing machines, color television, etc. It has been extended in recent years and interactive versions are available. The model argues that there is an initial flood of buyers for a new product who purchase the product to be "first on their block." This is followed by a growth in sales over a period of time until some maximum is reached, after which sales decline. The general form of the model is shown below:



The Bass model requires the following information:

1. The fraction of people who will buy the product when it comes on the market, p.
2. The number of initial purchases over the life cycle, m.
3. The number of previous buyers, y(t).
4. A "coefficient of imitation" which, in effect, represents the pressure to buy because peers have bought, q.

In equation form:

$$S(t) = pm + (q - p)y(t) - q/m(y(t))^2$$

Scenarios

Scenarios are a convenient way of communicating overall results to decision makers. They represent verbal descriptions of alternative futures that can be used for policy analysis. The three basic criteria for scenarios are that they be:

1. Consistent
2. Possible
3. Plausible.

Typically, a set of scenarios is developed which cover a "scenario space." That is, one, two or three parameters are varied and consistent scenarios are developed for possible and plausible combinations of these parameters. As a first example, consider the three scenarios based on different annual sales levels in 1983 for personal computers as shown in Table 1.

Delphi Inquiry

The scenarios shown in Table 1 are first-order, speculative cuts at the problem. To make these scenarios more realistics, expert opinion is needed. One way of obtaining such opinion is through a Delphi inquiry. The Delphi process, developed by Helmer and Dalkey at the RAND Corp. is designed to obtain consensus while eliminating the "jury room" effect that is generally observed when a group of experts is put together in a single room. That is, one voice tends to dominate and persuade the others that their opinion is correct. In a Delphi inquiry, questionnaires are sent to the panel and they are asked to give their best estimate for the probability of occurrence of events and the values of trends over a time horizon. An event is a single occurrence (e.g., issuance of standards for personal computer hardware) whereas a trend is an ongoing sequence of values (e.g., price index, annual production of personal computers).

Since the object is to obtain consensus, the Delphi process typically involves several iterations. The initial responses by the Delphi panel are analyzed and the median values fed back to the participants. The participants are asked whether they want to change their responses based upon these medians and any other inputs that they may have. If they feel the correct estimate should be far different than the median, they are asked to state

TABLE 1

BASELINE SCENARIOS

1983 CROSS SECTION

	A	B	C
Annual Sales (1983) units	100,000	500,000	1,000,000
Hardware cost/system (1978 $)	2,000	500	250
Memory	64K	512K	1Mb
Programming Skill	BASIC	Some precoded	All precoded
Regulation	High	Little	None
Standardization	Little	Some	Full
Major Use	Hobby	Business	Home & Business
Software Protection	None	Some legal	Mass mkt & Regulated
Connectivity	Some	Full	Metered
User Interface	Keyboard	Keyboard limited voice	Voice, some keyboard light pens
Losses due to PC Crime ($M)	Minor	10	100
Primary Hardware Producers	Similar to '78	DEC. HP/TI dominate	IBM/Exxon/NEC dominate
Primary Mass Storage	tape cassettes	High density floppies	Bubbles
State of Economy			

their reasons and evidence. If no consensus is reached after two rounds, additional rounds are run with the responses from the outlyers also fed back. This feedback process has, over the years, been shown to be convergent and to give relatively reliable estimates. Tables 2 and 3 show some preliminary events and trends which will be used in te Delphi inquiry, which will be undertaken after the market survey is completed.

Cross-Impact and Policy Analysis

The next stage of sophistication beyond Delphi analysis is cross-impact and policy analysis. The Delphi estimates are based on "all other things being equal." However, all other things are not equal. An event either has occurred or has not occurred. When it occurs, it can affect the chance of other events occuring and it can affect the value of trends. For example, if IBM were to enter the personal computer market, many other events and trends (e.g., annual sales of personal computers) would be affected. In cross-impact analysis, the first order effects of events and trends on one another are analyzed.

The purpose of cross-impact analysis typically is to guide policy makers on what to do under various contingencies. Public policy can be used to make some events occur or not occur and to change the value of trends. For example, if the government were to require registration of all computers and impose an annual license fee, the sales of personal computers would be affected. Cross-impact analysis allows investigating the effects of policy actions.

At the initial phase of our work, we have identified a number of policy areas and issues for consideration. These areas include (not necessarily in order of importance) education, standards, effects on telecommunications, registration, crime, software protection and software warranty, privacy and the effects on small businesses.

The results of the Delphi inquiry and the cross-impact andpolicy analysis will be fed back into the scenarios to provide additional realism and to communicate to policymakers the implications of various policy options.

Conclusions

In this paper we have briefly described the approach being used to assess the long-range implications of personal computers. The series of steps involve a market survey, a market forecast through use of a market model, development of scenarios, Delphi inquiry and cross-impact analysis. The end objective is policy analysis that can aid public policy decision makers in dealing appropriately with the personal computer industry.

TABLE 2

INITIAL LIST OF EVENTS

1. IBM or CDC or Sperry Rand-size company enters the Personal Computer Market.
2. HP/TI/or other large calculator company enters the Personal Computer Market.
3. 32-bit address register is introduced for personal computers by one of the top four producers.
4. Japanese (or Europeans) enter the Personal Computer Market in United State (as major competitor?).
5. Passage of a law requiring registry of personal computer ownership.
6. Occurrence of economic slowdown (GNP growth less than 2.5% annually).
7. Establishment of a low-cost computer conferencing system specially designed for personal computers.
8. Establishment of a national information retrieval network for personal computers. (Does Dow Jones qualify for this?)
9. Issuance of comprehensive standards for Personal Computer hardware.
10. Issuance of comprehensive standards for Personal Computer software.
11. Establishment of a network of personal computers in a community (city, region or state) for polling voters on public issues.
12. Development of pirate-resistant software for minicomputers.
13. Legislation requiring open and public access to all data files kept on personal computers.
14. Establishment of a subsidy for personal computer ownership by low income people.
15. Flat screen for TV and personal computers.

TABLE 3

INITIAL LIST OF TRENDS

1. Cost of Apple II capability personal computer (1978 = $1200).
2. Annual sale (number of units) of personal computers -- total
3. Annual sales (number of units) of personal computers to businesses ($5,000 or less, $10,000 or less, $15,000 or less).
4. Number of personal computers used in elementary and secondary education.
5. Number of people who can program in BASIC or more advanced computer language
6. Number of crimes committed using personal computers.
7. Number of new jobs created by personal computer industry and by industries created through personal computers.
8. Number of people-days lost because of brownouts and blackouts.
9. Number of people who can be reached by electronic mail.
10. Number of people who substitute communication for transportation.
11. Number of homes using personal computers for complete monitoring (temperature, on-off, energy consumption, burglary, etc.).
12. Number of personal computers exported by the United States, destination countries.
13. Number of personal computers imported by United States, source countries.
14. Cost of application software
15. End user cost of volatile storage
16. End user cost of displays
17. End user cost of CPU
18. End user cost of modems

"The Current Situation of the Japanese Microcomputer Products, Market, and Hobbyists"

Toshiaki Yasuda, Professor
School of Telecommunication Engineering,
Tokyo Denki University,
2-2 Kanda-nishiki, Chiyoda, Tokyo, Japan 101

Summary

This paper refers to the current situation of the Japanese microcomputer industry, its market size and computer hobbyists. Between the West Coast of the U.S. and the Japan Islands, there lies the Pacific Ocean, which also leads to San Francisco Bay between Berkeley/Oakland and San Francisco/Palo Alto. As if to symbolize this linkage by the ocean, microcomputer market is very active in Japan as it is in San Francisco Bay area.

Just as there are Bay Bridge and San Mateo Bridge in Bay Area, so between the West Coast and Japan, communication means such as the satellite international communication line, coaxial cable, ham radio, mailing system as well as the jet airplane flying this route in less than 8 hours. The Japanese microcomputer industry is trying to contribute toward the further development of microcomputers in cooperation with the West Coast's counterpart. There is a language barrier between the West Coast and Japan, but fortunately we have the computer language as our common language. The author firmly believe that our mutual cooperation in this field by taking full advantage of this common language will make for even more development of microcomputers.

History of microcomputers in Japan

It was in the spring of 1972 that a microcomputer was employed for the first time in Japan. To be more specific, it was when two kinds of microprocessors, known as Intel i-4004 and i-8008, were imported to Japan for sale. OEM showed much interest in i-4004 as component parts for high-class desktop electronic calculators and i-8008 as sequence control equipment.

In the summer of 1972, NEC and Hitachi Ltd. developed their own microprocessors and put them on the domestic market. NEC developed 4-bit and 8-bit types, while Hitachi developed a 4-bit type, and all of them were of the PMOS type. So far as the domestic market is concerned, however, they failed to draw as much attention as Intel products.

Since around 1974, demands for microprocessors in Japan have shown a rapid increase as the import of microprocessors manufactured by U.S. makers including Intel is on the increase, and new products such as i-8080A, F-8, PACE, SC/MP, M6800, 6502 and PPS-8 have been announced in succession. Thus, in addition to their own developed microprocessors, Japanese semi-conductor makers also explored the possibility of manufacturing such U.S.-made products popular in Japan. As a result, they began to launch the second source production of U.S.-made products. To begin with, NEC started the production of i-8080A and i-4040 and their family LSI under agreement with Intel, and then Hitachi Ltd. and Fujitsu Ltd. started the production of M6800 and its family.

After the second source production of i-8080A and i-4040, NEC is, in principle, continuing the production of Intel's subsequent products such as the 8085 family. Following Hitachi Ltd. and Fujitsu Ltd., the Japanese makers engaging in second source production under agreement with the U.S. makers are on the increase as shown below:

1) Intel i-8080A

   NEC, Mitsubishi Electric Co., Oki Electric Co. and Toshiba Electric Co.

2) Motorola M6800

   Hitachi Ltd. and Fujitsu Ltd.

3) Rockwell PPS

   Sharp Co.

4) Zilog Z-80

   Sharp Co.

In general, the U.S. products had drawn much attention from 1972 through 1977, but since late in 1977 those microprocessors, mainly of 4-bit one-chip type, developed by the Japanese makers, have come to find a large market.

The Japanese microcomputer market

Since 1972, microcomputers have been applied mainly to electronic cash registers (ECR) and automatic vending machines, and then since 1974 to POS terminals and intelligence terminals. Since 1977, they have found a large application in overall control systems and medical information systems.

From early 1978 on, they have come to be used widely for color TV receiver sets, FM receiver sets, cassette tape decks, microwave ovens, electric washing machines, and household air conditioners; and computerized home electric appliances are well accepted by Japanese housewives.

Microprocessor marketed in Japan in 1976 were 1,180,000 units for both domestic and U.S.-made products; they topped 2 million units in 1977 and are estimated to exceed 4 million units in 1978. Of these microprocessors, more than 85% are system and equipment built-in computers and the remainder are for use in development, evaluation, education/training and hobbies.

The Japanese microcomputer makers and products

The following is a list of Japanese microprocessor makers and their products including those products as second source production.

In the alphabets in the parentheses following the product name, 'n' stands for NMOS, 'p' for PMOS and 'c' for CMOS.

[16 bit processor]

1) Sanyo Electric Co.
   * MCP-3000 (n)

2) NEC
   * μCOM-16 (n)
   * μCOM-1600 (n)
     (μPB768B)

3) Matsushita (Panasonic Group)
   * MN1610 (n)
   * MN1611 (n)

[12 bit processor]

1) Toshiba Electric Co.
   * T3190 (p)

[8 bit processor]

1) Oki Electric Co.
   * MSM-3901 (i8080A)

2) Toshiba Electric Co.
   * TMP9080AC (i8080A)

3) NEC
   * μPD8080A/AF (i8080A)
   * μPD780C/D/D1 (n)
   * μCOM-85 (i8085A)

4) Fujitsu Ltd.
   * MB8861H/E/N (M6800)
   * MB8871H/E/N (M6802)

5) Hitachi Ltd.
   * HD46800D (M6800)
   * HD46802D (M6802)

6) Mitsubishi Electric Co.
   * M58710S (i8080A)
   * M58712S (i8085A)

7) Sharp Co.
   * Z-80 (Zilog)
   * PPS-8 (Rockwell)

[4 bit processor]

1) Sharp Co.
   * PPS-4/2 (Rockwell)

2) Toshiba Electric Co.
   * TMP4110P (n)

3) NEC
   * μPD760 (p)
     (μCOM-41)
   * μPD750 (n)
     (μCOM-4)

4) Hitachi Ltd.
   * HD35404 (i4004)

5) Oki Electric Co.
   * MSM5840 (c)
   * MSM5841 (c)
   * MSM5842 (c)
   * MSM5843 (c)

[One-chip Computer]

1) Matsushita (Panasonic)
   * MN1400 (n) (4 bit)
   * MN1430 (p) (4 bit)
   * MN1402 (n) (4 bit)
   * MN1432 (p) (4 bit)
   * MN1403 (n) (4 bit)
   * MN1404 (n) (4 bit)
   * MN1405 (n) (4 bit)
   * MN1435 (p) (4 bit)
   * MN1450/3/5 (c) (4 bit)
   * MN1498/9/9A (n) (4 bit)

2) Fujitsu Ltd.

* MB8841/2/3/4 (n) (4 bit)
* MB8881 (n) (8 bit)

3) Mitsubishi Electric Co.

* M58840 (p) (4 bit)

4) Hitachi Ltd.

* HD38600 (p) (4 bit)
* HD38630 (p) (4 bit)
* HD38750 (p) (4 bit)
* HD44750 (c) (4 bit)
* HD38700 (p) (4 bit)

5) NEC

* μPD548c (p) (4 bit)
* μPD546c (p) (4 bit)
* μPD547c (p) (4 bit)
* μPD550c (p) (4 bit)
* μPD551c (p) (4 bit)
* μPD766c (n) (4 bit)
* μPD8048c (i8048)

6) Toshiba Electric Co.

* T3444 (n) (8 bit)
* TMP4310 (n) (4 bit)
* TMP4315A (n) (4 bit)
* TMP4320 (n) (4 bit)

7) Sharp Co.

* SM-1 (p) (4 bit)
* SM-2 (p) (4 bit)
* SM-3 (p) (4 bit)
* SM-4 (c) (4 bit)
* SM-5 (c) (4 bit)

8) Oki Electric Co.

* MSM5840/1/2/3 (c) (4 bit)

In addition, there are 87 makers in Japan which specialize in the Single Bord, development evaluation system such as support system and built-in system based on the above-mentioned products and more than 500 kinds of such systems are being offered by those makers. As a CPU in use, 8080A is in comparatively high demand, accounting for about 40%.

Japanese Computer Hobbyists

Since SDK-80 of Intel Inc. was imported to Japan in 1975, there have emerged computer hobbyists who enjoy assembly and application of computer kits. Before Intel's products were imported, there had been some hobbyists in Japan, who installed a minicomputer in their home, but their number was only 10 or so, including Prof. Ryuichi Kobayashi of Rikkyo University and the author.

On the occasion of the marketing of the single-bord computer, such as Motrola's MEC 6800B, on top of SDK-80, the number of hobbyists had shown a ten-fold increase, totaling more than 100.

Some of these hobbyists went so far as to import personally SWTP's CT-1024TV typewriter in place of ASR-33. The above was the situation of the Japanese computer hobbyists three years ago.

Computer hobbyists in Japan number about 50,000 as of the summer of 1978, and they are now on the sharp increase. At first, they enjoyed the DIY kit TK-80 for educational purposes which was marketed by NEC for use with i-8080A (μPD8080A), and then went for imported kits such as Altair 8800, IMSAI and SWTP. In 1978, PET2001 and TRS-80 have gained great popularity and heavy demands are also concentrating on the BASIC oriented personal computers marketed by NEC, Hitachi, Matsushita and SORD.

A forecast of what great expansion the hobbyists or personal computers will show in the future can be made on the basis of the circulation of the following microcomputer-related publications written by the author and published by Kodansha Ltd., the largest publisher in Japan.

Written by T. Yasuda

1) Introduction to My Computer

(Published in March 1977, 350,000 copies sold)

2) How to Build My Computer

(Published in October 1977, 150,000 copies sold)

3) How to Use My Computer

(Published in April 1978, 100,000 copies sold)

In addition, a similar book written by Prof. Haruhisa Ishida of Tokyo University sold more than 100,000 copies, becoming a best-seller.

From the above, it may be forecasted that in Japan in the 1980's, demands for personal computers such as TRS-80 and PET2001 will be 300,000 to 500,000 units per year, while demands for microprocessors will top 10 million units per year. With such a huge market as a target, NEC, for instance, is making every effort to develop into the world's second

largest microcomputer device maker, and needless to say this effort is also being made by other makers.

The reason for such an increase in hobbyists is attributable to the marketing of NEC's training kit TK-80 in the summer of 1976. Equipped with the hexadecimal key input and 7 segment LED output, TK-80, like KIM-1 for 6502, is a variation kit for use in 8080A and is priced at about US$400.

Among those kits marketed at the similar price range are H68-TR (for M6800) by Hitachi Ltd., LKIT-16 (16 bit) by Panafacom (Panasonic Group), EX-80 (8080A) by Toshiba and LKIT-8 (for M6800) by Fujitsu. TK-80 has already sold 30,000 units.

All these kits have the Kansas City Standards-based tape, interface, printer interface, TV typewriter interface and color TV interface as attachments. Moreover, some kits are equipped with expanded functions which allow operation based on 2K bytes BASIC and 4K bytes BASIC and there are also many Japanese hobbyists who import the interface cards for S100-bus from the U.S. in order to connect them to these kits.

It is to be noted that since the end of 1977, it has become easy to obtain PE T-2001 and TRS-80 and these products so far purchased by the Japanese hobbyists amount to thousands of units. Further, Apple II is highly acclaimed in Japan as well.

Under these circumstances, the Japanese makers have put their comparable models on the market since September 1978. The typical products are "BASIC Master" marketed by Hitachi and "M100 Series" marketed by SORD, each being priced at about US$800. Other Japanese makers are also expected to follow suit. The present problem with these makers is that their plant production capacity cannot catch up with active demands.

Software and application of Hobby Computers

In addition to the home-made TK-80 and H68-TR, the Japanese computer hobbyists are buying many of the U.S.-made kits such as SDK-80, SDK-85, MEK 6800DII, SC/MP kit, IMSAI, MITS, SWTP, POLY and those hobbyists who are not capable of assembling these kits are scrambling for PET 2001, TRS-80, SORD M-100 (cpu: Z-80) and Hitachi BASIC Master (cpu: 6800). The purchasers of these DIY kits and personal computers are divided into two groups. One group is the younger generation consisting of high school and college students; these students, who can be said to be the hobbyists in the true sense of the word, are finding interest in controlling a model railway and music synthesizer and also enjoying the computer game. The computer game most popular among them is of course "STARTREK". The Japanese TV is also broadcasting a similar program titled "Space Battleship-YAMATO". In the case of this Japanese edition of the "STARTREK" same, "Enterprise" is replaced by "Yamato" (the most famous battleship in the pre-war Japanese navy) and "Klingon" by "Gamilas Star".

The second group is the adult hobbyists consisting of machine system designers, doctors, accountants, operators of small businesses; they are bent on finding how they can reflect their hobby effectively on their occupations. Both groups are usually making use of BASIC as their computer language. Most of the Japanese students majoring in science course master FORTRAN, while many students majoring in economics and business administration master COBOL. There are also many who use PL/I.

With the mounting diffusion of microcomputers, those who take an interest in BASIC began to increase, with an interest in CP/M also on the rise. We, as researchers for computer system development, have a great interest in PASCAL of late, but for the time being we are depending on the performance of the cross assembler and cross compiler.

Prof. Haruhisa Ishida of Tokyo University is credited with the diffusion of BASIC in Japan; he has introduced to the Japanese hobbyists the Palo Alto Tiny BASIC, which is in wide use as Toudai Ban BASIC (Tokyo University's version of BASIC). Further, Prof. Ishida has recently revised Texas Tiny BASIC and published it as Tokyo edition TB. Needless to say, all these are for use in 8080A.

In Japan as well, there are many 6800 users. Tom Pittman's TB has long drawn much attention, but it is not easy to obtain. In view of this situation, we have developed 2K bytes Tiny BASIC for use in 6800. Based on the computational method by binary processing, it boasts of the highest speed in the bench mark test. This Tiny BASIC, developed jointly by Mr. Fumiaki Hatanaka (an undergraduate of Tokyo Denki (Electric) University) and myself, is called Dendai Ban Tiny BASIC (Tokyo Denki University's edition of Tiny BASIC). The author is pleased to present the entire detail of this newly developed Tiny BASIC to the U.S. computer hobbyists as per Appendixed A and B.

Lastly, let me introduce Japanese magazines relating to the microcomputers, which are contributing toward the progress in microcomputers as in the case of the U.S. counterparts such as "Interface Age" and "BYTE".

Among typical Japanese microcomputer magazines popular among the microcomputer hobbyists are "ASCII" edited by Mr. Kazuhiko

Nishi, a student of Waseda University, "Transistor Technology", "Interface", "I/O", "Micom", "RAM", "Gakusyu Computer", etc. In addition, there are two famous electronic products shopping centers in Japan; one is located at Akihabara, Tokyo, and the other at Nipponbashi, Osaka; computer stores in these two shopping centers number as many as 25.

Appendix: A

Abstract of TDU Tiny BASIC (for 6800) Grammar

(under MIKBUG control)

(TDU: Tokyo Denki University)

1) Command

NEW RUN LIST [n1, n2]: (list and tape save)
LOAD: (tape) AUTO [n]
EXIT (monitor call) GOTO n
RETURN (or RET): (rerun)
PRINT$ (or PR$): (memory size)
Control-C: (break) Control-0: (del)
Control-X: (can)

2) Statement

[LET] PRINT (or PR)
INPUT (or IN) GOTO
GOSUB RETURN (or RET)
IF STOP REM END
FOR s TO e [STEP e]
NEXT [v] THEN

s: substitute expression
e: expression (-32768≦e≦32767)
v: variable (variable name: = A,B,~,Z,a,~,z)
array name: % e (or % (e))
: (key-input)
: @ (e) : (display)
: multiple statement separator (use for〈statement: statement: ..》

3) Operator

+ - * / = > < >=
=< >= <= <> ><

4) Function

RND (e) ABS (e) MOD (e)
CHR (e):(ASCII ← e)
TAB (e):(print pointer ← e)
(e):(poke) # (v)=e:(peek)
USR (e) : (PC ← e)
USR (el, e2):(PC ← el, IX ← e_2)
USR (el, e2, e3):
(PC←el, IX←e_2, B·A←e3)

5) Constant

-32768≦integer≦32767 (decimal)
$0000≦$FFFF (hexadecimal)
'string' or "string"

6) Error

Error number	
100	Input Error
110	Memory Size Over
120	Invalid Line Number
130	Print Statement Error
140	Zero Divide
150	Expression too muth Complex
160	Illegal Arithmetic
170	MOD Error
180	Illegal Statement
190	Subroutine Error
200	Undefind Line Number
210	For Loop Error
220	Undefind For Loop

Microcomputers Goes to TV-Top-Star.

In the fall of 1977, NEC's TK-80 microcomuter system was telecasted through the News Wide Show (11 PM Show) of Nippon Television Network System.

Photo

left: Professor Toshiaki Yasuda
right: Mr.Kyosen Ohashi (TV caster)
(Mr.Ohashi introduced TK-80.)

Appendix: B Assemble list of TDU BASIC

note:[¥] means [$]

Developed by Prf.T.Yasuda and Mr.F.Hatanaka (TDU)

```
001A                    ORG     #1A
001A 0002       V       RMB     2
001C 0002       W       RMB     2
001E 0002       Z       RMB     2
0020 0002       RND     RMB     2
0022 0002       LN      RMB     2
0024 0002       PX      RMB     2
0026 0002       SP      RMB     2
0028 0002       LNADRS  RMB     2
002A 0002       BUFEND  RMB     2
002C 0001       LDFLG   RMB     1
002D 0001       AFLG    RMB     1
002E 0001       CHCNT   RMB     1
002F 0001       F       RMB     1
0030 0050       BUFFER  RMB     80
     0080       EXSTK   EQU     *
0080 0002       EXSP    RMB     2
0082 0034               RMB     52
00B6 0002       PRSTRT  RMB     2
00B8 0002       PREND   RMB     2
00BA 0002       RAMEND  RMB     2
00BC 0002       SUBSP   RMB     2
00BE 0002       FORSP   RMB     2
00C0 0040               RMB     64
     0100       FORSTK  EQU     *
     A047       STACK   EQU     #A047
     A080       SUBSTK  EQU     #A080
     E1AC       INEEE   EQU     #E1AC
     E1D1       OUTEEE  EQU     #E1D1
     E0D0       RESET   EQU     #E0D0
0100                    ORG     #100
                *
0100 BD 01B3            JSR     START
0103 7E 01CB            JMP     CONTRL
                *
0106 8D 68      LOADM   BSR     IN
0108 81 02              CMP A   #2
010A 26 FA              BNE     LOADM
010C 20 1E              BRA     GL2
                *
010E 96 2C      GTLINE  LDA A   LDFLG
0110 26 F4              BNE     LOADM
0112 96 2D              LDA A   AFLG
0114 27 11              BEQ     GL1
0116 86 0A              LDA A   #10
0118 5F                 CLR B
0119 CE 0022            LDX     #LN
011C BD 049C            JSR     ADD3
011F BD 040A            JSR     PNUM
0122 BD 07EE            JSR     OUTS
0125 20 08              BRA     GL3
                *
0127 86 23      GL1     LDA A   #'#
0129 BD 0448            JSR     OUT
012C CE 0030    GL2     LDX     #BUFFER
012F 98 20      GL3     EOR A   RND
0131 97 20              STA A   RND
0133 8D 3B              BSR     IN
0135 81 0D              CMP A   #¥D
0137 26 0F              BNE     GL4

0139 6F 00              CLR     0,X
013B DF 2A              STX     BUFEND
013D 96 2C              LDA A   LDFLG
013F 26 03              BNE     *+5
0141 BD 0840    NEWL2   JSR     NWLINE
0144 CE 0030            LDX     #BUFFER
0147 39                 RTS
                *
0148 81 18      GL4     CMP A   #¥18
014A 26 04              BNE     GL5
014C 8D F3              BSR     NEWL2
014E 20 DF              BRA     GL3
                *
0150 81 0F      GL5     CMP A   #¥F
0152 26 0D              BNE     GL6
0154 86 5F              LDA A   #¥5F
0156 BD 0448            JSR     OUT
0159 8C 0030            CPX     #BUFFER
015C 27 D1              BEQ     GL3
015E 09                 DEX
015F 20 CE              BRA     GL3
                *
0161 8C 0076    GL6     CPX     #BUFFER+70
0164 27 05              BEQ     ERR100
0166 A7 00              STA A   0,X
0168 08                 INX
0169 20 C4              BRA     GL3
                *
016B C6 64      ERR100  LDA B   #100
016D 7E 085D            JMP     ERRMSG

0170 BD 093F    IN      JSR     INFFF
0173 84 7F              AND A   #¥7F
0175 27 F9              BEQ     IN
0177 81 7F              CMP A   #¥7F
0179 27 F5              BEQ     IN
017B 81 03              CMP A   #3
017D 26 0B              BNE     BR6
017F 20 4A              BRA     CONTRL
                *
0181 36         BREAK   PSH A
0182 B6 8004            LDA A   ¥8004
0185 2B 02              BMI     BR5
0187 8D E7              BSR     IN
0189 32         BR5     PUL A
018A 39         BR6     RTS
                *
018B DE B6      INIT1   LDX     PRSTRT
018D 07                 TPA
018E A7 00              STA A   0,X
0190 08                 INX
0191 A7 00              STA A   0,X
0193 DF B8              STX     PREND
0195 39                 RTS
                *
0196 8D F3      LOAD    BSR     INIT1
0198 97 2C              STA A   LDFLG
019A 20 41              BRA     C4
                *
019C 07         AUTO    TPA
019D 97 2D              STA A   AFLG

019F BD 06A7            JSR     EXPRPP
01A2 80 0A              SUB A   #10
01A4 C2 00              SBC B   #0
01A6 97 23      SETLN   STA A   LN+1
01A8 D7 22              STA B   LN
01AA 39                 RTS
                *
01AB BD 0335    STOP    JSR     PRINT
01AE BD 0684            JSR     PSX
01B1 20 1E              BRA     C3
                *
01B3 CE 0980    START   LDX     ¥¥0980
01B6 DF B6              STX     PRSTRT
01B8 86 AA              LDA A   ¥¥AA
01BA 08         MEMTST  INX
01BB E6 01              LDA B   1,X
01BD A7 01              STA A   1,X
01BF A6 01              LDA A   1,X
01C1 E7 01              STA B   1,X
01C3 81 AA              CMP A   ¥¥AA
01C5 27 F3              BEQ     MEMTST
01C7 DF BA              STX     RAMEND
01C9 8D C0      NEW     BSR     INIT1
01CB 8E A047    CONTRL  LDS     #STACK
01CE 0F                 SEI
01CF 8D 7F              BSR     INIT2
01D1 BD 0840 C3         JSR     NWLINE
01D4 CE 0932            LDX     #RDYMSG
01D7 BD 0857            JSR     PD
01DA 07                 TPA
01DB 97 28              STA A   LNADRS
01DD 8E A047    C4      LDS     #STACK
01E0 BD 010E            JSR     GTLINE
01E3 BD 0506            JSR     GETN
01E6 24 7D              BCC     DIRECT
01E8 8D BC              BSR     SETLN
01EA 8D 02              BSR     EDIT
01EC 20 EF              BRA     C4
                *
01EE DF 24      EDIT    STX     PX
01F0 9F 26              STS     SP
01F2 8D 37              BSR     LDTST
01F4 DF 1E              STX     Z
01F6 25 12              BCS     INSERT
01F8 8D 72              BSR     NXTL
01FA 08                 INX
01FB 35                 TXS
01FC DE 1E              LDX     Z
01FE 58         MOVE1   ASL B
01FF 33                 PUL B
0200 E7 00              STA B   0,X
0202 08                 INX
0203 2A F9              BPL     MOVE1
0205 24 F7              BCC     MOVE1
0207 09                 DEX
0208 DF B8              STX     PREND
020A DE 24      INSERT  LDX     PX
020C A6 00              LDA A   0,X
020E 27 3D              BEQ     INS3
0210 96 2B              LDA A   BUFEND+1
0212 90 25              SUB A   PX+1

0214 8B 03              ADD A   #3
0216 DE B8              LDX     PREND
0218 9B B9              ADD A   PREND+1
021A 97 B9              STA A   PREND+1
021C 24 03              BCC     *+5
021E 7C 00B8            INC     PREND
0221 8D 7D              BSR     SIZE
0223 24 0F              BCC     INS2
0225 DF B8              STX     PREND
0227 C6 6E      ERR110  LDA B   #110
0229 20 5F              BRA     ERROR
                *
022B 96 2C      LDTST   LDA A   LDFLG
022D 27 4C              BEQ     LNFD
022F DE B8              LDX     PREND
0231 09                 DEX
0232 0D                 SEC
0233 39                 RTS
                *
0234 9E B8      INS2    LDS     PREND
0236 08                 INX
0237 09         MOVE2   DEX
0238 A6 00              LDA A   0,X
023A 36                 PSH A
023B 9C 1E              CPX     Z
023D 26 F8              BNE     MOVE2
023F 9E 22              LDS     LN
0241 AF 00              STS     0,X
0243 08                 INX
0244 9E 24              LDS     PX

0246 34                 DES
0247 08         MOVE3   INX
0248 32                 PUL A
0249 A7 00              STA A   0,X
024B 26 FA              BNE     MOVE3
024D 9E 26      INS3    LDS     SP
024F 39                 RTS
                *
0250 CE 0000    INIT2   LDX     #0
0253 DF 2C              STX     LDFLG
0255 CE 0100            LDX     #FORSTK
0258 DF BE              STX     FORSP
025A CE A080            LDX     #SUBSTK
025D DF BC              STX     SUBSP
025F CE 0080            LDX     #EXSTK
0262 DF 80              STX     EXSP
0264 39                 RTS
                *
0265 DF 24      DIRECT  STX     PX
0267 CE 0880            LDX     #TABLE2-2
026A 20 57              BRA     STMT4
                *
026C 08         NXTL    INX
026D 08                 INX
026E 6D 00      NXTL2   TST     0,X
0270 26 FB              BNE     *-3
0272 39                 RTS
                *
0273 DE 28      FINDER  LDX     LNADRS
0275 2B 04              BMI     LNFD
0277 8D 04              BSR     LNFD+2

0279 24 19              BCC     LNFD3+4
027B DE B6      LNFD    LDX     PRSTRT
027D 96 23              LDA A   LN+1
027F D6 22              LDA B   LN
0281 2B 05              BMI     ERR120
0283 26 0B              BNE     LNFD3
0285 4D                 TST A
0286 26 08              BNE     LNFD3
0288 C6 78      ERR120  LDA B   #120
028A 7E 085D    ERROR   JMP     ERRMSG
                *
028D 8D DD              BSR     NXTL
028F 08                 INX
0290 8D 6E      LNFD3   BSR     CMPX
0292 22 F9              BHI     *-5
0294 39                 RTS
                *
0295 BD 06A7    IF      JSR     EXPRPP
0298 5D                 TST B
0299 26 20              BNE     STMT3
029B 4D                 TST A
029C 27 D0              BEQ     NXTL2
029E 20 1B              BRA     STMT3
                *
02A0 96 BB      SIZE    LDA A   RAMEND+1
02A2 D6 BA              LDA B   RAMEND
02A4 90 B9              SUB A   PREND+1
02A6 D2 B8              SBC B   PREND
02A8 39                 RTS
                *
```

```
02A9 7E 01D1 C7      JMP    C3
                     *
02AC 8D A2   RUN     BSR    INIT2
02AE DE B6           LDX    PRSTRT
02B0 DF 28   STMT    STX    LNADRS
02B2 A6 00           LDA A  0,X
02B4 48              ASL A
02B5 96 28           LDA A  LNADRS
02B7 23 F0           BLS    C7
02B9 08              INX
02BA 08              INX
02BB BD 0181 STMT3   JSR    BREAK
02BE DF 24           STX    PX
02C0 CE 08A0         LDX    #TABLE-2
02C3 8E A047 STMT4   LDS    #STACK
02C6 BD 0580         JSR    OPRATE
02C9 09              DEX
02CA A6 00   STMT5   LDA A  0,X
02CC 08              INX
02CD 81 3A           CMP A  #':
02CF 27 EA           BEQ    STMT3
02D1 4D              TST A
02D2 26 F6           BNE    STMT5
02D4 20 DA           BRA    STMT
                     *
02D6 5F      HEXA    CLR B
02D7 D7 1E           STA B  Z
02D9 8D 16           BSR    TSTHEX
02DB 24 C3           BCC    SIZE
02DD 49      HEX2    ROL A
02DE 48              ASL A

02DF 48              ASL A
02E0 48              ASL A
02E1 48              ASL A
02E2 79 001E SHIFT   ROL    Z
02E5 59              ROL B
02E6 48              ASL A
02E7 26 F9           BNE    SHIFT
02E9 08              INX
02EA 8D 05           BSR    TSTHEX
02EC 25 EF           BCS    HEX2
02EE 96 1E           LDA A  Z
02F0 39              RTS
                     *
02F1 BD 052E TSTHEX  JSR    TSTN
02F4 25 09           BCS    HEX3
02F6 80 07           SUB A  #7
02F8 81 3A           CMP A  #$3A
02FA 0C              CLC
02FB 2D 02           BLT    *+4
02FD 81 40           CMP A  #$40
02FF 39      HEX3    RTS
                     *
0300 E1 00   CMPX    CMP B  0,X
0302 26 02           BNE    *+4
0304 A1 01           CMP A  1,X
0306 39              RTS
                     *
0307 DF 24   SRCH2   STX    PX
0309 CE 0910         LDX    #TABLE4-2
030C 9F 26   SEARCH  STS    SP

030E 9E 24   S0      LDS    PX
0310 34              DES
0311 08              INX
0312 08      S1      INX
0313 32      S2      PUL A
0314 81 20           CMP A  #$20
0316 27 FB           BEQ    S2
0318 16              TAB
0319 A0 00           SUB A  0,X
031B 48              ASL A
031C 26 09           BNE    S5
031E 24 F2           BCC    S1
0320 08              INX
0321 31      S4      INS
0322 9F 24   S3      STS    PX
0324 9E 26           LDS    SP
0326 39              RTS
                     *
0327 A6 00   S5      LDA A  0,X
0329 08              INX
032A 2A FB           BPL    S5
032C C1 2E           CMP B  #'.
032E 27 F1           BEQ    S4
0330 43              COM A
0331 26 DB           BNE    S0
0333 20 ED           BRA    S3
                     *
0335 8D 22   PRINT   BSR    SKB9
0337 27 23           BEQ    PR9
0339 8D 40   PR3     BSR    PQUOTE
033B 27 08           BEQ    PR4

033D DF 24           STX    PX
033F CE 091A         LDX    #TABLE5-2
0342 BD 0580         JSR    OPRATE
0345 8D 12   PR4     BSR    SKB9
0347 27 13           BEQ    PR9
0349 81 3B           CMP A  #';
034B 27 06           BEQ    PR5
034D 81 2C           CMP A  #',
034F 26 49           BNE    ERR130
0351 8D 0F           BSR    PT
0353 08      PR5     INX
0354 8D 03   PR6     BSR    SKB9
0356 26 E1           BNE    PR3
0358 39              RTS
                     *
0359 7E 045F SKB9    JMP    SKB
                     *
035C 7E 0840 PR9     JMP    NWLINE
                     *
035F BD 07EE         JSR    OUTS
0362 96 2E   PT      LDA A  CHCNT
0364 85 07           BIT A  #7
0366 26 F7           BNE    PT-3
0368 39              RTS
                     *
0369 8D 29   CHR     BSR    FACTPP
036B 7E 0448 OUT7    JMP    OUT
                     *
036E 8D 24   TAB     BSR    FACTPP
0370 16              TAB
0371 20 03           BRA    TAB2
                     *
0373 BD 07EE         JSR    OUTS
0376 D1 2E   TAB2    CMP B  CHCNT
0378 22 F9           BHI    *-5
037A 39              RTS
                     *
037B 8D DC   PQUOTE  BSR    SKB9
037D 81 22           CMP A  #'"
037F 27 04           BEQ    *+6
0381 81 27           CMP A  #''
0383 26 0E           BNE    PQ5
0385 16              TAB
0386 08              INX
0387 20 02           BRA    *+4
                     *
0389 8D E0   PQ4     BSR    OUT7
038B A6 00           LDA A  0,X
038D 27 0B           BEQ    ERR130
038F 08              INX
0390 11              CBA
0391 26 F6           BNE    PQ4
0393 39      PQ5     RTS
                     *
0394 BD 05C1 FACTPP  JSR    FACT9
0397 7E 06A9         JMP    POP
                     *
039A C6 82   ERR130  LDA B  #130
039C 20 02           BRA    *+4
                     *
039E C6 8C   ERR140  LDA B  #140

03A0 7E 085D         JMP    ERRMSG
                     *
03A3 A6 00   DIVIDE  LDA A  0,X
03A5 AA 01           ORA A  1,X
03A7 27 F5           BEQ    ERR140
03A9 8D 40           BSR    SETCNT
03AB 8D 2C           BSR    ROL2X
03AD 49              ROL A
03AE 40              NEG A
03AF 16              TAB
03B0 8D 2C           BSR    CHECK
03B2 49              ROL A
03B3 86 01           EOR A  #1
03B5 46              ROR A
03B6 8D 1C           BSR    DIV3
03B8 BD 0300         JSR    CMPX
03BB 26 08           BNE    RDIV
03BD 4F              CLR A
03BE 16              TAB
03BF 6C 03           INC    3,X
03C1 26 02           BNE    *+4
03C3 6C 02           INC    2,X
03C5 08      RDIV    INX
03C6 08              INX
03C7 39              RTS
                     *
03C8 8D 21   BINDIV  BSR    SETCNT
03CA 8D 0D   DIV2    BSR    ROL2X
03CC 49              ROL A
03CD 59              ROL B

03CE A0 01           SUB A  1,X
03D0 E2 00           SBC B  0,X
03D2 8D 0A           BSR    CHECK
03D4 7A 002F DIV3    DEC    F
03D7 26 F1           BNE    DIV2
03D9 69 03   ROL2X   ROL    3,X
03DB 69 02           ROL    2,X
03DD 39              RTS
                     *
03DE 37      CHECK   PSH B
03DF E8 00           EOR B  0,X
03E1 33              PUL B
03E2 0D              SEC
03E3 2A 05           BPL    *+7
03E5 AB 01           ADD A  1,X
03E7 E9 00           ADC B  0,X
03E9 0C              CLC
03EA 39              RTS
                     *
03EB 86 10   SETCNT  LDA A  #16
03ED 97 2F           STA A  F
03EF 4F              CLR A
03F0 16              TAB
03F1 39              RTS
                     *
03F2 8D F7   MULT    BSR    SETCNT
03F4 48      MLT2    ASL A
03F5 59              ROL B
03F6 69 01           ROL    1,X
03F8 69 00           ROL    0,X
03FA 24 04           BCC    *+6

03FC AB 03           ADD A  3,X
03FE E9 02           ADC B  2,X
0400 7A 002F         DEC    F
0403 26 EF           BNE    MLT2
0405 08              INX
0406 08              INX
0407 7E 04A0         JMP    STAX
                     *
040A EE 00   PNUM    LDX    0,X
040C DF 1C           STX    W
040E CE 001C         LDX    #W
0411 8D 3E           BSR    ABSX
0413 24 04           BCC    PN2
0415 86 2D           LDA A  #'-
0417 8D 2F           BSR    OUT
0419 09      PN2     DEX
041A 09              DEX
041B C6 0A           LDA B  #10
041D E7 01           STA B  1,X
041F 6F 00           CLR    0,X
0421 57              ASR B
0422 37      PN3     PSH B
0423 8D A3           BSR    BINDIV
0425 33              PUL B
0426 36              PSH A
0427 5A              DEC B
0428 26 F8           BNE    PN3
042A CE 0030         LDX    #BUFFER
042D C6 04           LDA B  #4
042F 96 2F   PN4     LDA A  F
0431 32              PUL A
0432 26 04           BNE    PN5
0434 97 2F           STA A  F
0436 27 02           BEQ    *+4
0438 8D 04   PN5     BSR    PN6
043A 5A              DEC B
043B 26 F2           BNE    PN4
043D 32              PUL A
043E 8B 30   PN6     ADD A  #$30
0440 7D 002D         TST    AFLG
0443 27 03           BEQ    *+5
0445 A7 00           STA A  0,X
0447 08              INX
0448 7C 002E OUT     INC    CHCNT
044B BD 0181         JSR    BREAK
044E 7E E1D1         JMP    OUTEEE
                     *
0451 6D 00   ABSX    TST    0,X
0453 2A 08           BPL    RABS
0455 60 01   NEGX    NEG    1,X
0457 26 02           BNE    *+4
0459 6A 00           DEC    0,X
045B 63 00           COM    0,X
045D 39      RABS    RTS
                     *
045E 08      SKB2    INX
045F A6 00   SKB     LDA A  0,X
0461 81 20           CMP A  #$20
0463 27 F9           BEQ    SKB2
0465 4D              TST A
0466 27 03           BEQ    *+5
```

```
0468 81 3A          CMP A  #':
046A 0D             SEC
046B 39             RTS
                    *
046C 8D F1    EXPR3 BSR    SKB
046E 81 2D          CMP A  #'-
0470 26 04          BNE    E3
0472 8D 1F          BSR    E9
0474 20 07          BRA    E5
                    *
0476 81 2B    E3    CMP A  #'+
0478 27 01          BEQ    E4
047A 09             DEX
047B 8D 5C    E4    BSR    TERM2
047D DE 24    E5    LDX    PX
047F 8D DE          BSR    SKB
0481 81 2B          CMP A  #'+
0483 26 06          BNE    E6
0485 8D 52          BSR    TERM2
0487 8D 0E    E55   BSR    ADD
0489 20 F2          BRA    E5
                    *
048B 81 2D    E6    CMP A  #'-
048D 26 15          BNE    E7
048F 8D 02          BSR    E9
0491 20 F4          BRA    E55
                    *
0493 8D 44    E9    BSR    TERM2
0495 20 BE          BRA    NEGX
                    *
0497 BD 06A9  ADD   JSR    POP
049A 08             INX
049B 08             INX
049C AB 01    ADD3  ADD A  1,X
049E E9 00          ADC B  0,X
04A0 A7 01    STAX  STA A  1,X
04A2 E7 00          STA B  0,X
04A4 39       E7    RTS
                    *
04A5 BD 05C1  ABS   JSR    FACT9
04A8 8D 31          BSR    SAVE
04AA 8D A5          BSR    ABSX
04AC DE 24          LDX    PX
04AE 39             RTS
                    *
04AF 8D AE    TSTE@ BSR    SKB
04B1 81 3D          CMP A  #'=
04B3 26 15          BNE    R5
04B5 31             INS
04B6 31             INS
04B7 5C             INC B
04B8 20 0D          BRA    R3
                    *
04BA 5F       REL   CLR B
04BB 8D F2          BSR    TSTE@
04BD 8D 12          BSR    R7
04BF 26 0A          BNE    R6
04C1 8D EC          BSR    TSTE@
04C3 81 3E    R2    CMP A  #'>
04C5 26 03          BNE    R5
04C7 5C       R3    INC B

04C8 08       R4    INX
04C9 4F             CLR A
04CA 39       R5    RTS
                    *
04CB 8D F6    R6    BSR    R2
04CD 26 FB          BNE    R5
04CF 8D DE          BSR    TSTE@
04D1 81 3C    R7    CMP A  #'<
04D3 26 F5          BNE    R5
04D5 CB 04          ADD B  #4
04D7 20 EF          BRA    R4
                    *
04D9 8D 5E    TERM2 BSR    TERM
04DB DF 24    SAVE  STX    PX
04DD DE 80          LDX    EXSP
04DF 39             RTS
                    *
04E0 8D 8A    EXPR  BSR    EXPR3
04E2 8D D6          BSR    REL
04E4 5D             TST B
04E5 27 E3          BEQ    R5
04E7 37             PSH B
04E8 8D 82          BSR    EXPR3
04EA BD 06A9        JSR    POP
04ED DE 80          LDX    EXSP
04EF A0 01          SUB A  1,X
04F1 E2 00          SBC B  0,X
04F3 33             PUL B
04F4 2D 07          BLT    LT
04F6 2E 03          BGT    GT

04F8 4D             TST A
04F9 27 01          BEQ    EQ
04FB 57       GT    ASR B
04FC 57       EQ    ASR B
04FD C4 01    LT    AND B  #1
04FF E7 01          STA B  1,X
0501 6F 00          CLR    0,X
0503 DE 1E          LDX    Z
0505 39             RTS
                    *
0506 8D 26    GETN  BSR    TSTN
0508 24 23          BCC    NOTN
050A 5F             CLR B
050B D7 1F          STA B  Z+1
050D D7 1E    GN2   STA B  Z
050F 84 0F          AND A  #$F
0511 97 2F          STA A  F
0513 96 1F          LDA A  Z+1
0515 48             ASL A
0516 59             ROL B
0517 48             ASL A
0518 59             ROL B
0519 9B 1F          ADD A  Z+1
051B D9 1E          ADC B  Z
051D 48             ASL A
051E 59             ROL B
051F 9B 2F          ADD A  F
0521 97 1F          STA A  Z+1
0523 C9 00          ADC B  #0
0525 08             INX
0526 8D 06          BSR    TSTN

0528 25 E3          BCS    GN2
052A 96 1F          LDA A  Z+1
052C 0D             SEC
052D 39       NOTN  RTS
                    *
052E BD 05B3  TSTN  JSR    SKB8
0531 81 30          CMP A  #'0
0533 0C             CLC
0534 2D 02          BLT    *+4
0536 81 3A          CMP A  #$3A
0538 39       T3    RTS
                    *
0539 8D 33    TERM  BSR    FACT
053B 8D 76    T1    BSR    SKB8
053D 81 2A          CMP A  #'*
053F 26 0B          BNE    T2
0541 8D 14          BSR    FACT2
0543 BD 03F2        JSR    MULT
0546 DF 80    T11   STX    EXSP
0548 DE 24          LDX    PX
054A 20 EF          BRA    T1
                    *
054C 81 2F    T2    CMP A  #'/
054E 26 E8          BNE    T3
0550 8D 05          BSR    FACT2
0552 BD 03A3        JSR    DIVIDE
0555 20 EF          BRA    T11
                    *
0557 8D 15    FACT2 BSR    FACT
0559 7E 04DB        JMP    SAVE
                    *
055C 08       HEXA2 INX
055D BD 02D6        JSR    HEXA
0560 20 32          BRA    PUSH
                    *
0562 8D 0A    PEEK  BSR    FACT
0564 BD 06A9        JSR    POP
0567 8D 63          BSR    TAX
0569 A6 00          LDA A  0,X
056B 5F             CLR B
056C 20 24          BRA    PUSH-2
                    *
056E 08       FACT  INX
056F 8D 95          BSR    GETN
0571 25 21          BCS    PUSH
0573 81 23          CMP A  #'#
0575 27 EB          BEQ    PEEK
0577 81 24          CMP A  #'$
0579 27 E1          BEQ    HEXA2
057B DF 24          STX    PX
057D CE 08F5        LDX    #TABLE3-2
0580 BD 030C  OPRATE JSR   SEARCH
0583 A6 01          LDA A  1,X
0585 36             PSH A
0586 A6 00          LDA A  0,X
0588 36             PSH A
0589 DE 24          LDX    PX
058B 39             RTS
                    *
058C 8D 3E    FVAR  BSR    TAX
058E A6 01          LDA A  1,X

0590 E6 00          LDA B  0,X
0592 DE 24          LDX    PX
0594 DF 1E    PUSH  STX    Z
0596 DE 80          LDX    EXSP
0598 09             DEX
0599 09             DEX
059A 9C 2A          CPX    BUFEND
059C 2F 08          BLE    ERR150
059E BD 04A0  PUSH2 JSR    STAX
05A1 DF 80    PUSH3 STX    EXSP
05A3 DE 1E          LDX    Z
05A5 39             RTS
                    *
05A6 C6 96    ERR150 LDA B #150
05A8 7E 085D  ER    JMP    ERRMSG
                    *
05AB C6 A0    ERR160 LDA B #160
05AD 20 F9          BRA    ER
                    *
05AF C6 AA    ERR170 LDA B #170
05B1 20 F5          BRA    ER
                    *
05B3 7E 045F  SKB8  JMP    SKB
                    *
05B6 8D 68    NOTFNC BSR   TSTV
05B8 24 D2          BCC    FVAR
05BA A6 00          LDA A  0,X
05BC 81 28          CMP A  #'(
05BE 26 E8          BNE    ERR160
05C0 08             INX
05C1 BD 04E0  FACT9 JSR    EXPR
05C4 8D ED          BSR    SKB8
05C6 81 29          CMP A  #')
05C8 26 E1          BNE    ERR160
05CA 08             INX
05CB 39             RTS
                    *
05CC DF 24    TAX   STX    PX
05CE 97 1F          STA A  Z+1
05D0 D7 1E          STA B  Z
05D2 DE 1E          LDX    Z
05D4 39             RTS
                    *
05D5 BD 04E0  MOD   JSR    EXPR
05D8 A6 00          LDA A  0,X
05DA 81 2C          CMP A  #',
05DC 26 D1          BNE    ERR170
05DE 08             INX
05DF 8D E0    MOD2  BSR    FACT9
05E1 DF 1E          STX    Z
05E3 DE 80          LDX    EXSP
05E5 BD 03A3        JSR    DIVIDE
05E8 20 B4          BRA    PUSH2
                    *
05EA 96 21    RANDOM LDA A RND+1
05EC D6 20          LDA B  RND
05EE DB 21          ADD B  RND+1
05F0 89 95          ADC A  #$95
05F2 C9 AB          ADC B  #$AB
05F4 D7 21          STA B  RND+1
05F6 97 20          STA A  RND

05F8 8D 9A          BSR    PUSH
05FA 20 E3          BRA    MOD2
                    *
05FC 8D 02    USER  BSR    USER2
05FE 20 92          BRA    PUSH-2
                    *
0600 8D 18    USER2 BSR    ARG2
0602 36             PSH A
0603 37             PSH B
0604 8D 0D          BSR    ARG
0606 36             PSH A
0607 37             PSH B
0608 8D 09          BSR    ARG
060A 36             PSH A
060B 37             PSH B
060C 07             TPA
060D 36             PSH A
060E 8D B4          BSR    FACT9+3
0610 DF 24          STX    PX
0612 3B             RTI
                    *
0613 A6 00    ARG   LDA A  0,X
0615 81 2C          CMP A  #',
0617 26 04          BNE    NONE
0619 08             INX
061A 7E 06A7  ARG2  JMP    EXPRPP
                    *
061D 4F       NONE  CLR A
061E 5F             CLR B
061F 39             RTS
```

```
                *
0620 8D 91      TSTV    BSR     SKB8
0622 81 40              CMP A   #$40
0624 27 0C              BEQ     ARRAY
0626 81 41              CMP A   #'A
0628 25 26              BCS     NOTV
062A 81 5A              CMP A   #'Z
062C 22 22              BHI     NOTV
062E 48                 ASL A
062F 5F                 CLR B
0630 08                 INX
0631 39                 RTS
                *
0632 BD 056E    ARRAY   JSR     FACT
0635 BD 06A9            JSR     POP
0638 C5 C0              BIT B   #$C0
063A 26 11              BNE     ERR11
063C 0D                 SEC
063D 49                 ROL A
063E 59                 ROL B
063F 9B B9              ADD A   PREND+1
0641 D9 B8              ADC B   PREND
0643 D1 BA              CMP B   RAMEND
0645 26 02              BNE     *+4
0647 91 BB              CMP A   RAMEND+1
0649 24 02              BCC     ERR11
064B 0C                 CLC
064C 39                 RTS
                *
064D 7E 0227    ERR11   JMP     ERR110
                *
0650 0D         NOTV    SEC
0651 39                 RTS
                *
0652 BD 04AF    TSTE@2  JSR     TSTE@
0655 C6 B4      ERR180  LDA B   #180
0657 7E 085D    ERR19   JMP     ERRMSG
                *
065A BD 0620    LET3    JSR     TSTV
065D 25 F6              BCS     ERR180
065F 97 1D      LET4    STA A   W+1
0661 D7 1C              STA B   W
0663 8D ED              BSR     TSTE@2
0665 8D 40              BSR     EXPRPP
0667 DF 24              STX     PX
0669 DE 1C              LDX     W
066B 39                 RTS
                *
066C C6 BE      ERR190  LDA B   #190
066E 20 E7              BRA     ERR19
                *
0670 5F         THEN    CLR B
0671 37                 PSH B
0672 BD 0506            JSR     GETN
0675 25 4F              BCS     GOTO2
0677 7E 02BB            JMP     STMT3
                *
067A BD 056E    POKE    JSR     FACT
067D 8D 2A              BSR     POP
067F 8D DE              BSR     LET4
0681 16                 TAB
0682 20 63              BRA     LET2
                *
0684 9F 26      PSX     STS     SP
0686 35                 TXS
0687 DE BC              LDX     SUBSP
0689 8C A04A            CPX     #STACK+3
068C 27 DE              BEQ     ERR190
068E 09                 DEX
068F 09                 DEX
0690 AF 00              STS     0,X
0692 DF BC      RESTR   STX     SUBSP
0694 30                 TSX
0695 9E 26              LDS     SP
0697 39                 RTS
                *
0698 DE BC      PLX     LDX     SUBSP
069A 8C A080            CPX     #SUBSTK
069D 27 CD              BEQ     ERR190
069F 9F 26              STS     SP
06A1 AE 00              LDS     0,X
06A3 08                 INX
06A4 08                 INX
06A5 20 EB              BRA     RESTR
                *
06A7 8D 16      EXPRPP  BSR     EXPR7
06A9 DF 1E      POP     STX     Z
06AB DE 80              LDX     EXSP
06AD 8C 0080            CPX     #EXSTK
06B0 27 A3              BEQ     ERR180
06B2 A6 01              LDA A   1,X
06B4 E6 00              LDA B   0,X
06B6 08                 INX
06B7 08                 INX
06B8 DF 80              STX     EXSP
06BA DE 1E              LDX     Z
06BC 39                 RTS
06BD DE 24      EXPR9   LDX     PX
06BF 7E 04E0    EXPR7   JMP     EXPR
                *
06C2 4F         GOTO    CLR A
06C3 36         GOSUB   PSH A
06C4 8D E1              BSR     EXPRPP
06C6 BD 01A6    GOTO2   JSR     SETLN
06C9 32                 PUL A
06CA 4D                 TST A
06CB 27 02              BEQ     *+4
06CD 8D B5              BSR     PSX
06CF BD 0273            JSR     FINDER
06D2 24 04              BCC     *+6
06D4 C6 C8      ERR200  LDA B   #200
06D6 20 55              BRA     ERR22
                *
06D8 7E 02B0            JMP     STMT
                *
06DB BD 045F    LET     JSR     SKB
06DE 81 23              CMP A   #'#
06E0 27 98              BEQ     POKE
06E2 BD 065A    LET1    JSR     LET3
06E5 A7 01              STA A   1,X
06E7 E7 00      LET2    STA B   0,X
06E9 DE 24              LDX     PX
06EB 39                 RTS
                *
06EC 8D F4      FOR     BSR     LET1
06EE 96 1D              LDA A   W+1
06F0 D6 1C              LDA B   W
06F2 BD 0594            JSR     PUSH
06F5 BD 0307            JSR     SRCH2
06F8 8C 0914            CPX     #TOMSG
06FB 26 2E              BNE     ERR210
06FD 8D BE              BSR     EXPR9
06FF BD 0307            JSR     SRCH2
0702 8C 091A            CPX     #STPMSG
0705 27 08              BEQ     FOR1
0707 86 01              LDA A   #1
0709 5F                 CLR B
070A BD 0592            JSR     PUSH-2
070D 20 07              BRA     *+4
                *
070F 8D AC      FOR1    BSR     EXPR9
0711 DF 24              STX     PX
0713 96 25              LDA A   PX+1
0715 D6 24              LDA B   PX
0717 BD 0594            JSR     PUSH
071A 96 1D              LDA A   W+1
071C D6 1C              LDA B   W
071E DE BE              LDX     FORSP
0720 8D 18      FOR3    BSR     TST2
0722 24 0C              BCC     FOR4
0724 DE BE              LDX     FORSP
0726 8C 00C0            CPX     #FORSTK-64
0729 26 07              BNE     FOR4+2
072B C6 D2      ERR210  LDA B   #210
072D 7E 085D    ERR22   JMP     ERRMSG
                *
0730 8D 58      FOR4    BSR     INX8
0732 8D 10              BSR     FORPSH
0734 20 4F              BRA     NEXT5
                *
0736 BD 0300    TSTSTK  JSR     CMPX
0739 27 08              BEQ     TST3
073B 8D 4D              BSR     INX8
073D 8C 0100    TST2    CPX     #FORSTK
0740 26 F4              BNE     TSTSTK
0742 0D                 SEC
0743 39         TST3    RTS
                *
0744 8D 00      FORPSH  BSR     *+2
0746 8D 00              BSR     *+2
0748 BD 06A9            JSR     POP
074B 09                 DEX
074C 09                 DEX
074D 7E 04A0    STAX2   JMP     STAX
0750 C6 DC      ERR220  LDA B   #220
0752 20 D9              BRA     ERR22
                *
0754 BD 0620    NEXT    JSR     TSTV
0757 DF 24              STX     PX
0759 DE BE              LDX     FORSP
075B 24 04              BCC     *+6
075D A6 01              LDA A   1,X
075F E6 00              LDA B   0,X
0761 8D DA              BSR     TST2
0763 25 EB              BCS     ERR220
0765 DF BE              STX     FORSP
0767 A6 05              LDA A   5,X
0769 E6 04              LDA B   4,X
076B EE 00              LDX     0,X
076D BD 049C            JSR     ADD3
0770 DE BE              LDX     FORSP
0772 E1 02              CMP B   2,X
0774 0D                 SEC
0775 2D 07              BLT     NEXT4
0777 0C                 CLC
0778 2E 04              BGT     NEXT4
077A A1 03              CMP A   3,X
077C 27 13              BEQ     AGAIN
077E 46         NEXT4   ROR A
077F A8 04              EOR A   4,X
0781 2B 0E              BMI     AGAIN
0783 8D 05              BSR     INX8
0785 DF BE      NEXT5   STX     FORSP
0787 DE 24              LDX     PX
0789 39                 RTS
                *
078A 8D 00      INX8    BSR     *+2
078C 08                 INX
078D 08                 INX
078E 08                 INX
078F 08                 INX
0790 39                 RTS
                *
0791 EE 06      AGAIN   LDX     6,X
0793 39                 RTS
                *
0794 DF 1C      INPUT   STX     W
0796 96 1C              LDA A   W
0798 27 41              BEQ     ERR10
079A DE 2A              LDX     BUFEND
079C DF 1A              STX     V
079E DE 1C              LDX     W
07A0 BD 037B            JSR     P@UOTE
07A3 26 02              BNE     IN2
07A5 8D 37              BSR     TSTCM
07A7 BD 0620    IN2     JSR     TSTV
07AA 25 2F              BCS     ERR10
07AC 36                 PSH A
07AD 37                 PSH B
07AE DF 1C              STX     W
07B0 DE 1A              LDX     V
07B2 8D 37              BSR     SKB1
07B4 26 09              BNE     IN3
07B6 86 3F              LDA A   #'?
07B8 8D 36              BSR     OUT3
07BA 8D 32              BSR     OUTS
07BC BD 012C            JSR     GL2
07BF BD 06A7    IN3     JSR     EXPRPP
07C2 8D 1A              BSR     TSTCM
07C4 DF 1A              STX     V
07C6 30                 TSX
07C7 EE 00              LDX     0,X
07C9 31                 INS
07CA 31                 INS
07CB BD 04A0            JSR     STAX
07CE DE 1C              LDX     W
07D0 8D 19              BSR     SKB1
07D2 27 0F              BEQ     IN4
07D4 81 2C              CMP A   #',
07D6 26 03              BNE     ERR10
07D8 08                 INX
07D9 20 CC              BRA     IN2
                *
07DB 7E 016B    ERR10   JMP     ERR100
                *
07DE 8D 04      TSTCM   BSR     TSTCM2
07E0 26 01              BNE     *+3
07E2 08                 INX
07E3 39         IN4     RTS
                *
07E4 36         TSTCM2  PSH A
07E5 8D 04              BSR     SKB1
07E7 81 2C              CMP A   #',
07E9 32                 PUL A
07EA 39                 RTS
                *
07EB 7E 045F    SKB1    JMP     SKB
                *
07EE 86 20      OUTS    LDA A   #$20
07F0 7E 0448    OUT3    JMP     OUT
                *
07F3 5F         LIST    CLR B
07F4 86 01              LDA A   #1
07F6 BD 01A6            JSR     SETLN
07F9 C6 7F              LDA B   #$7F
```

```
07FB 86 FF          LDA A  #$FF
07FD 36             PSH A
07FE 8D EB          BSR    SKB1
0800 32             PUL A
0801 27 0E          BEQ    LIST2
0803 BD 06A7        JSR    EXPRPP
0806 BD 01A6        JSR    SETLN
0809 8D D9          BSR    TSTCM2
080B 26 04          BNE    LIST2
080D 08             INX
080E BD 06A7        JSR    EXPRPP
0811 97 27   LIST2  STA A  SP+1
0813 D7 26          STA B  SP
0815 DE 26          LDX    SP
0817 26 05          BNE    *+7
0819 CE 7FFF        LDX    #32767
081C DF 26          STX    SP
081E BD 0278        JSR    LNFD
0821 20 02          BRA    *+4
                    *
0823 8D 10   LIST3  BSR    LISTX
0825 96 27          LDA A  SP+1
0827 D6 26          LDA B  SP
0829 BD 0300        JSR    CMPX
082C 24 F5          BCC    LIST3
082E 86 03          LDA A  #3
0830 8D BE          BSR    OUT3
0832 7E 01CB C9     JMP    CONTRL
                    *
0835 86 02   LISTX  LDA A  #2
0837 8D B7          BSR    OUT3
0839 8D 12          BSR    PNUM9
083B 08             INX
083C 86 20          LDA A  #$20
083E 8D 14          BSR    PD2
0840 DF 1E   NWLINE STX    Z
0842 CE 0937        LDX    #CRLF
0845 8D 10          BSR    PD
0847 7F 002E        CLR    CHCNT
084A DE 1E   LX2    LDX    Z
084C 39             RTS
                    *
084D DF 1E   PNUM9  STX    Z
084F BD 040A        JSR    PNUM
0852 20 F6          BRA    LX2
                    *
0854 8D 9A   PD2    BSR    OUT3
0856 08             INX
0857 A6 00   PD     LDA A  0,X
0859 26 F9          BNE    PD2
085B 08             INX
085C 39             RTS
                    *
085D 8E A047 ERRMSG LDS    #STACK
0860 8D DE          BSR    NWLINE
0862 CE 092B        LDX    #ERR
0865 8D F0          BSR    PD
0867 17             TBA
0868 5F             CLR B
0869 8D 0D          BSR    PRACC
086B DE 28          LDX    LNADRS

086D 2B C3          BMI    C9
086F 8D CF          BSR    NWLINE
0871 8D C2          BSR    LISTX
0873 20 BD          BRA    C9
                    *
0875 BD 06A7 PREXPR JSR    EXPRPP
0878 36      PRACC  PSH A
0879 37             PSH B
087A DF 1E          STX    Z
087C 30             TSX
087D 8D D0          BSR    PNUM9+2
087F 33             PUL B
0880 32             PUL A
0881 39             RTS
                    *
     0080   N       EQU    $80
     0882   TABLE2  EQU    *
0882 4C             FCB    'L+'I+'S+'T+N
0883 49
0884 53
0885 D4
0886 07F3           FDB    LIST
0888 4C             FCB    'L+'O+'A+'D+N
0889 4F
088A 41
088B C4
088C 0196           FDB    LOAD
088E 52             FCB    'R+'U+'N+N
088F 55
0890 CE
```

```
0891 02AC           FDB    RUN
0893 45             FCB    'E+'X+N
0894 D8
0895 E0D0           FDB    RESET
0897 41             FCB    'A+'U+'T+'O+N
0898 55
0899 54
089A CF
089B 019C           FDB    AUTO
089D 4E             FCB    'N+'E+'W+N
089E 45
089F D7
08A0 01C9           FDB    NEW
     08A2   TABLE   EQU    *
08A2 4C             FCB    'L+'E+'T+N
08A3 45
08A4 D4
08A5 06DB           FDB    LET
08A7 49             FCB    'I+'F+N
08A8 C6
08A9 0295           FDB    IF
08AB 47             FCB    'G+'O+'T+'O+N
08AC 4F
08AD 54
08AE CF
08AF 06C2           FDB    GOTO
08B1 47             FCB    'G+'O+'S+'U+'B+N
08B2 4F
08B3 53
08B4 55
08B5 C2

08B6 06C3           FDB    GOSUB
08B8 52             FCB    'R+'E+'T+N
08B9 45
08BA D4
08BB 0698           FDB    PLX
08BD 4E             FCB    'N+'E+'X+'T+N
08BE 45
08BF 58
08C0 D4
08C1 0754           FDB    NEXT
08C3 46             FCB    'F+'O+'R+N
08C4 4F
08C5 D2
08C6 06EC           FDB    FOR
08C8 54             FCB    'T+'H+'E+'N+N
08C9 48
08CA 45
08CB CE
08CC 0670           FDB    THEN
08CE 52             FCB    'R+'E+'M+N
08CF 45
08D0 CD
08D1 026E           FDB    NXTL2
08D3 50             FCB    'P+'R+'I+'N+'T+N
08D4 52
08D5 49
08D6 4E
08D7 D4
08D8 0335           FDB    PRINT
08DA 49             FCB    'I+'N+'P+'U+'T+N
08DB 4E
08DC 50
08DD 55
08DE D4
08DF 0794           FDB    INPUT
08E1 50             FCB    'P+'R+N
08E2 D2
08E3 0335           FDB    PRINT
08E5 49             FCB    'I+'N+N
08E6 CE
08E7 0794           FDB    INPUT
08E9 53             FCB    'S+'T+'O+'P+N
08EA 54
08EB 4F
08EC D0
08ED 01AB           FDB    STOP
08EF 45             FCB    'E+'N+'D+N
08F0 4E
08F1 C4
08F2 01CB           FDB    CONTRL
08F4 FF             FCB    $FF
08F5 06DB           FDB    LET
                    *
     08F7   TABLE3  EQU    *
08F7 52             FCB    'R+'N+'D+'(+N
08F8 4E
08F9 44
08FA A8
08FB 05EA           FDB    RANDOM
08FD 41             FCB    'A+'B+'S+'(+N
08FE 42
```

"Computer Faire" Introduced to Japan through the Nationwide TV Network with T-Shirt Marked with "Computer Faire"

In the fall of 1977, "The First West Computer Faire" was introduced through the Japanese popular TV program (11 PM Show) of Nippon Television Network System (NTV).

This TV program was broadcasted through 24 TV stations across the country with more than 2 million people enjoying the microcomputer show.

Photo (clockwise)

Professor Toshiaki Yasuda (Tokyo Denki Univ.)

Mr. Kyosen Ohashi (TV caster)

Miss Reiko Itsuki (Miss Itsuki, one of the Japanese top fashion models, wears Computer Faire T-Shirt)

Miss Kitsuko Matsuoka (TV actress)

Mr. Kazuhiko Nishi (Editor of ASCII magazine)

```
08FF 53
0900 A8
0901 04A5           FDB    ABS
0903 4D             FCB    'M+'O+'D+'(+N
0904 4F
0905 44
0906 A8
0907 05D5           FDB    MOD
0909 55             FCB    'U+'S+'R+'(+N
090A 53
090B 52
090C A8
090D 05FC           FDB    USER
090F FF             FCB    $FF
0910 05B6           FDB    NOTFNC
                    *
     0912   TABLE4  EQU    *
0912 54             FCB    'T+'O+N
0913 CF
0914 1234   TOMSG   FDB    $1234
0916 53             FCB    'S+'T+'E+'P+N
0917 54
0918 45
0919 D0
091A ABCD   STPMSG  FDB    $ABCD
                    *
     091C   TABLE5  EQU    *
091C 43             FCB    'C+'H+'R+'(+N
091D 48
091E 52
091F A8
0920 0369           FDB    CHR
0922 54             FCB    'T+'A+'B+'(+N
0923 41
0924 42
0925 A8
0926 036E           FDB    TAB
0928 FF             FCB    $FF
0929 0875           FDB    PREXPR
                    *
092B 45     ERR     FCC    /ERROR /
092C 52
092D 52
092E 4F
092F 52
0930 20
0931 00             FCB    0
0932 52     RDYMSG  FCC    /READY/
0933 45
0934 41
0935 44
0936 59
0937 0D     CRLF    FCB    $0D,$0A,$FF,$F
0938 0A                F,$FF,$FF,$FF,0
0939 FF
093A FF
093B FF
093C FF
093D FF
093E 00
                    *
093F BD E1AC INFFF  JSR    INEEE
0942 81 60          CMP A  #$60
0944 25 02          BCS    *+4
0946 80 20          SUB A  #$20
0948 81 25          CMP A  #$25
094A 26 02          BNE    *+4
094C 86 40          LDA A  #$40
094E 39             RTS
                    *
                    END
```

SYMBOL TABLE

V	001A	W	001C	Z	001E	RND	0020	LN	0022
PX	0024	SP	0026	LNADRS	0028	BUFEND	002A	LDFLG	002C
AFLG	002D	CHCNT	002E	F	002F	BUFFER	0030	EXSTK	0080
EXSP	0080	PRSTRT	00B6	PREND	00B8	RAMEND	00BA	SUBSP	00BC
FORSP	00BE	FORSTK	0100	STACK	A047	SUBSTK	A080	INEEE	E1AC
OUTEEE	E1D1	RESET	E0D0	LOADM	0106	GTLINE	010E	GL1	0127
GL2	012C	GL3	012F	NEWL2	0141	GL4	0148	GL5	0150
GL6	0161	ERR100	016B	IN	0170	BREAK	0181	BR5	0189
BR6	018A	INIT1	018B	LOAD	0196	AUTO	019C	SETLN	01A6
STOP	01AB	START	01B3	MEMTST	01BA	NEW	01C9	CONTRL	01CB
C3	01D1	C4	01DD	EDIT	01EE	MOVE1	01FE	INSERT	020A
ERR110	0227	LDTST	022B	INS2	0234	MOVE2	0237	MOVE3	0247
INS3	024D	INIT2	0250	DIRECT	0265	NXTL	026C	NXTL2	026E
FINDER	0273	LNFD	027B	ERR120	0288	ERROR	028A	LNFD3	0290
IF	0295	SIZE	02A0	C7	02A9	RUN	02AC	STMT	02B0
STMT3	02BB	STMT4	02C3	STMT5	02CA	HEXA	02D6	HEX2	02DD
SHIFT	02E2	TSTHEX	02F1	HEX3	02FF	CMPX	0300	SRCH2	0307
SEARCH	030C	S0	030E	S1	0312	S2	0313	S4	0321
S3	0322	S5	0327	PRINT	0335	PR3	0339	PR4	0345
PR5	0353	PR6	0354	SKB9	0359	PR9	035C	PT	0362
CHR	0369	OUT7	036B	TAB	036E	TAB2	0376	PQUOTE	037B
PQ4	0389	PQ5	0393	FACTPP	0394	ERR130	039A	ERR140	039E
DIVIDE	03A3	RDIV	03C5	BINDIV	03C8	DIV2	03CA	DIV3	03D4
ROL2X	03D9	CHECK	03DE	SETCNT	03EB	MULT	03F2	MLT2	03F4
PNUM	040A	PN2	0419	PN3	0422	PN4	042F	PN5	0438
PN6	043E	OUT	0448	ABSX	0451	NEGX	0455	RABS	045D
SKB2	045E	SKB	045F	EXPR3	046C	E3	0476	E4	047B
E5	047D	E55	0487	E6	048B	E9	0493	ADD	0497
ADD3	049C	STAX	04A0	E7	04A4	ABS	04A5	TSTE0	04AF
REL	04BA	R2	04C3	R3	04C7	R4	04C8	R5	04CA
R6	04CB	R7	04D1	TERM2	04D9	SAVE	04DB	EXPR	04E0
GT	04FB	E0	04FC	LT	04FD	GETN	0506	GN2	050D
NOTN	052D	TSTN	052E	T3	0538	TERM	0539	T1	053B
T11	0546	T2	054C	FACT2	0557	HEXA2	055C	PEEK	0562
FACT	056E	OPRATE	0580	FVAR	058C	PUSH	0594	PUSH2	059E
PUSH3	05A1	ERR150	05A6	ER	05A8	ERR160	05AB	ERR170	05AF
SKB8	05B3	NOTFNC	05B6	FACT9	05C1	TAX	05CC	MOD	05D5
MOD2	05DF	RANDOM	05EA	USER	05FC	USER2	0600	ARG	0613
ARG2	061A	NONE	061D	TSTV	0620	ARRAY	0632	ERR11	064D
NOTV	0650	TSTE@2	0652	ERR180	0655	ERR19	0657	LET3	065A
LET4	065F	ERR190	066C	THEN	0670	POKE	067A	PSX	0684
RESTR	0692	PLX	0698	EXPRPP	06A7	POP	06A9	EXPR9	06BD
EXPRT	06BF	GOTO	06C2	GOSUB	06C3	GOTO2	06C6	ERR200	06D4
LET	06DB	LET1	06E2	LET2	06E7	FOR	06EC	FOR1	070F
FOR3	0720	ERR210	072B	ERR22	072D	FOR4	0730	TSTSTK	0736
TST2	073D	TST3	0743	FORPSH	0744	STAX2	074D	ERR220	0750
NEXT	0754	NEXT4	077E	NEXT5	0785	INX8	078A	AGAIN	0791
INPUT	0794	IN2	07A7	IN3	07BF	ERR10	07DB	TSTCM	07DE
IN4	07E3	TSTCM2	07E4	SKB1	07EB	OUTS	07EE	OUT3	07F0
LIST	07F3	LIST2	0811	LIST3	0823	C9	0832	LISTX	0835
NWLINE	0840	LX2	084A	PNUM9	084D	PD2	0854	PD	0857
ERRMSG	085D	PREXPR	0875	PRACC	0878	N	0080	TABLE2	0882
TABLE	08A2	TABLE3	08F7	TABLE4	0912	TOMSG	0914	STPMSG	091A
TABLE5	091C	ERR	092B	RDYMSG	0932	CRLF	0937	INFFF	093F

FIRST West Coast Computer Faire

CONFERENCE PROCEEDINGS

TABLE OF CONTENTS

SECOND Computer Faire
CONFERENCE PROCEEDINGS

TABLE OF CONTENTS

THE BEST OF THE COMPUTER FAIRES, VOLUMES I & II STILL AVAILABLE

The Computer Faire still has copies of previous issues of its *Conference Proceedings* in stock. When this stock is depleted, these unique reference and tutorial publications will *not* be reprinted.

The *Conference Proceedings of the FIRST West Coast Computer Faire* contains over 300 pages of technical papers and tutorials that were presented at the Faire, held in San Francisco in April, 1977. Though some of these papers have since been reprinted in various magazines and books, most of them remain available only in the *Proceedings.*

The *Conference Proceedings of the SECOND West Coast Computer Faire,* produced in March, 1978, contains over 500 pages of state-of-the-art technical papers, tutorials, and discussions relating to low-cost computing.

CONFERENCE PROCEEDINGS

Please send_____ copies of the 334-page *Conference Proceedings of the FIRST West Coast Computer Faire,* for which I have enclosed payment of:

( ) $13.00 each, for shipment to my non-California address

( ) $13.72 each, for shipment to my tax-laden California address.

Please send_____ copies of the 505-page *Conference Proceedings of the SECOND West Coast Computer Faire,* for which I have enclosed payment of:

( )$14.00 each, for shipment to my non-California address

( )$14.78 each, for shipment to my tax-laden California address.

I'd love to bend over backwards for you.

Please give me the $2 savings allowed for my ordering *both* the *FIRST* and *SECOND* Computer Faire *Conference Proceedings.* Please send me_____pairs of copies of both *Proceedings,* for which I have enclosed payment of:

( )$25 each pair, for shipment to my non-California address

( )$26.50 each pair, for shipment to my tax-laden California address.

I'd love to bend over backwards for you.

Please make checks payable to: "COMPUTER FAIRE"
Above prices include: cost of the *Proceedings*, handling, UPS or 4th class shipping, and applicable taxes. Both publications are currently in stock. Computer Faire will not reprint *Proceedings,* once the present stock is exhausted. Therefore, a *full* refund will be promptly made if its stock is depleted.

name_____phone number (just in case)_(___)_____

mailing_____

address_____

city_____state_____ZIP/postal code_____

BUSINESS REPLY MAIL
No Postage Necessary If Mailed In U.S.A.

FIRST CLASS
Permit No. 91
Palo Alto, CA

POSTAGE WILL BE PAID BY

COMPUTER FAIRE
Conferences & Expositions
on
Intelligent Machines
for
Home, Business, & Industry

BOX 1579
PALO ALTO CA 94302

PEOPLE'S COMPUTER CO.
PUBLICATIONS

PEOPLE'S COMPUTERS

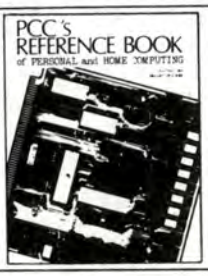
A magazine for beginning and intermediate level computer users, educators and those who wonder what computing is all about. It covers everything from elementary programming to assembly language, from ready-to-use program listings to the development of a new language. Not to mention games, listings, surveys, interviews and consumer advice.
Published bimonthly. $2.00 per copy.

DR. DOBB'S JOURNAL

DDJ publishes significant systems and applications software in the public domain and provides a forum for the personal computing community. It also offers independent product evaluations and consumer advocacy. "THE software source for microcomputers. Highly recommended." –*The Data Bus*, Philadelphia Area Computer Society.
Published 10 times/yr. $2.00 per copy.

COMPUTER MUSIC JOURNAL

A journal devoted to the high-quality musical applications of computers. This is a unique source of technical and consumer information on the art and science of computer generated music. No other publication like it in the field.
Published quarterly, $3.00 per copy.



THE PCC REFERENCE BOOK OF PERSONAL AND HOME COMPUTING

Contains a complete bibliography for computing, a massive index of articles from computing magazines, and a computer music bibliography. Comprehensive articles on software, hardware, applications, robots, and consumer info. . . plus extensive lists of clubs, newsletters, magazines, computer stores and more!
248 pages, paperback, $5.95.

WHAT TO DO AFTER YOU HIT RETURN

Computer games for all ages. Educational and fun! Most can be played with or without a computer. Readers can captain a starship, hunt the wumpus, draw a mandala, run the US economy and lots more. Fully illustrated.
184 pages, 4th printing. $8 retail.

DR. DOBB'S JOURNAL – VOL. ONE

One of the best bargains for inexpensive software anywhere. A single, bound volume of the first 10 issues of *DDJ*, complete and unabridged. The emphasis is on systems and applications software, including extensive source code program listings. Languages like CASUAL, SCELBAL, MINOL and a miriad of tiny BASICs, plus: a systems monitor, floating point routine, text editor and speech synthesis.
360 pages, 2nd printing, $13 retail.

☐ **People's Computers**
$10/yr., published bimonthly
Outside U.S., add $4.

☐ **Dr. Dobb's Journal**
$15/yr., published 10 times/yr.
Outside U.S., add $4.

☐ **Computer Music Journal**
$12/yr., published quarterly
Outside U.S., add $3.

☐ Bill me ☐ Payment Enclosed

Book Orders: Please add $1.00 shipping for orders under $10.00; add $2.00 for orders over $10.00.
Payment must accompany book orders. California Residents add 6% sales tax on price of book.

NAME_____
ADDRESS _____
CITY/STATE _____ ZIP_____
☐ Visa/BankAmericard Card No._____
☐ Master Charge Expiration date_____

Please Note: Payments must be in U.S. Dollars drawn on a U.S. bank. Please mail this form or a facsimile to:
People's Computer Co., Dept. 5H, Box E, 1263 El Camino Real, Menlo Park, CA 94025.

Serious personal computer enthusiasts read

BYTE®

the leading magazine in the personal computer field

Readers look forward to each monthly issue of **BYTE** for information on:

- software
- hardware
- simulations
- computer games
- robotics and scores of other novel applications
- computers and calculators
- languages and compilers techniques
- custom systems design

Tutorial information for both the beginner and experienced computer user.

To be aware of fast-paced changes in the fast-growing field of microprocessors you need to read **BYTE**—acknowledged as the leading magazine in the personal computer marketplace. You will find **BYTE** tutorials on hardware and software invaluable reading, the reports on home applications instructive, and the evaluative reviews based on first hand experiences with home computer products stimulating.

BYTE editorials explore the fun of using and applying computers for personal satisfaction in activities as wide ranging as electronic music, video games, investment analysis, household management, systems control, and other areas of vital interest to computer enthusiasts.

Each monthly issue of **BYTE** is packed with timely articles by professionals, computer scientists and serious amateurs. Isn't it time you enjoyed their company? Subscribe **now** to **BYTE**... the Small Systems Journal.

VISIT BYTE AT BOOTHS 415 & 417 OF THE 3RD COMPUTER FAIRE

Read your first copy of **BYTE**, if it's everything you expected, honor our invoice. If it isn't, just write "CANCEL" across the invoice and mail it back. You won't be billed and the first issue is yours.

BYTE Subscription Department: WATTS number: Toll Free 800-258-5485

Allow 6 to 8 weeks for processing.

BYTE Subscription Dept. P.O. Box 590 Martinsville, NJ 08836

PLEASE ENTER MY SUBSCRIPTION FOR

☐ One year $15 (12 issues) ☐ Two years $27 ☐ Three years $39

☐ Check enclosed (entitles you to 13 issues for price of 12) North America only

☐ Bill Visa ☐ Bill Master Charge ☐ Bill me (North America only)

Card Number _____ Expiration Date _____

Signature _____ Name (please print) _____

Address _____

City _____ State/Country _____ Code _____

FOREIGN RATES (To expedite service, please remit in U.S. Funds)

☐ Canada or Mexico $17.50—One year ☐ Two years $32 ☐ Three years $46.50

☐ Europe $25—One year (Air delivered)

☐ All other countries except above $25—One year (Surface delivery)
Air delivery available on request

7WB8