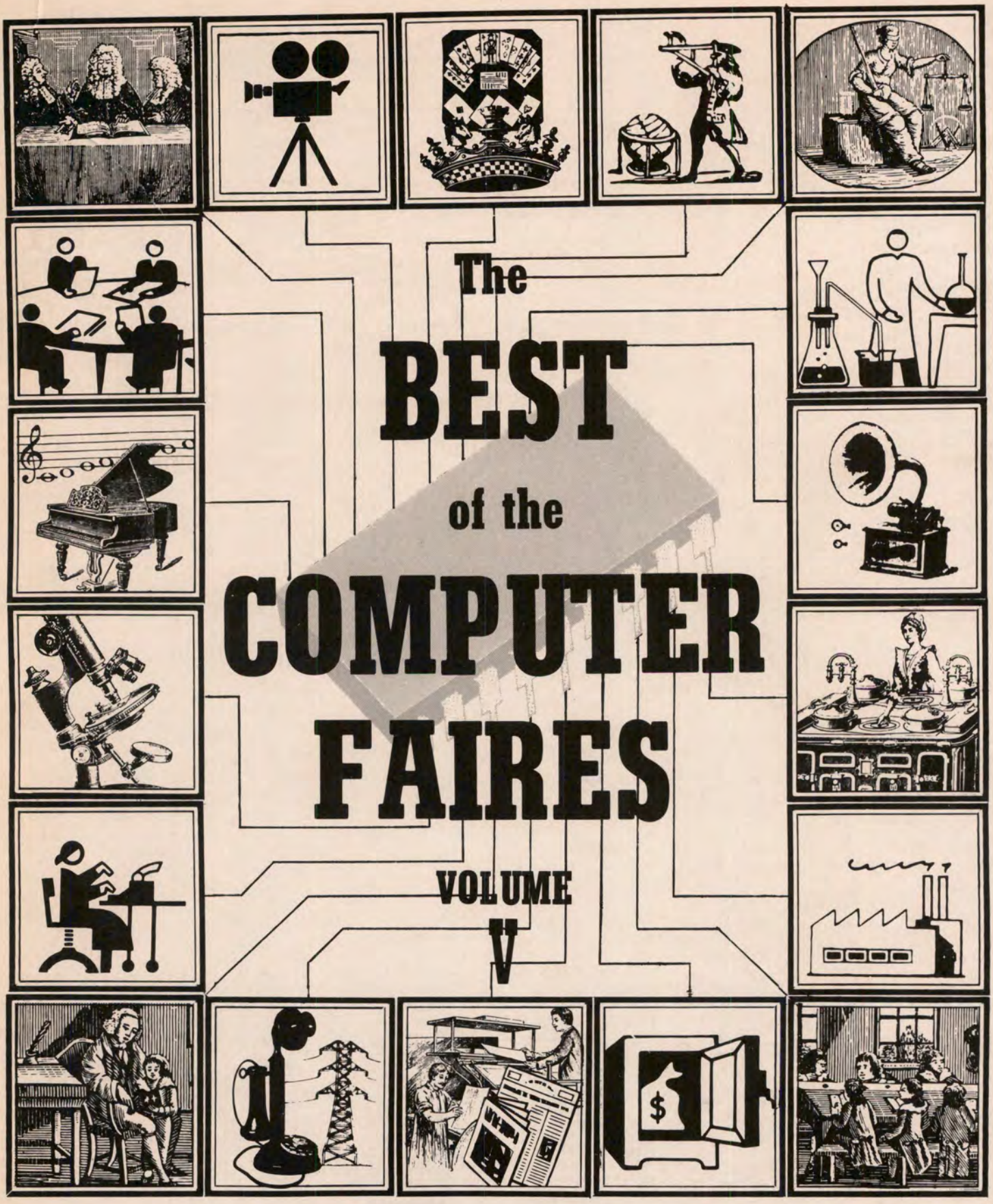
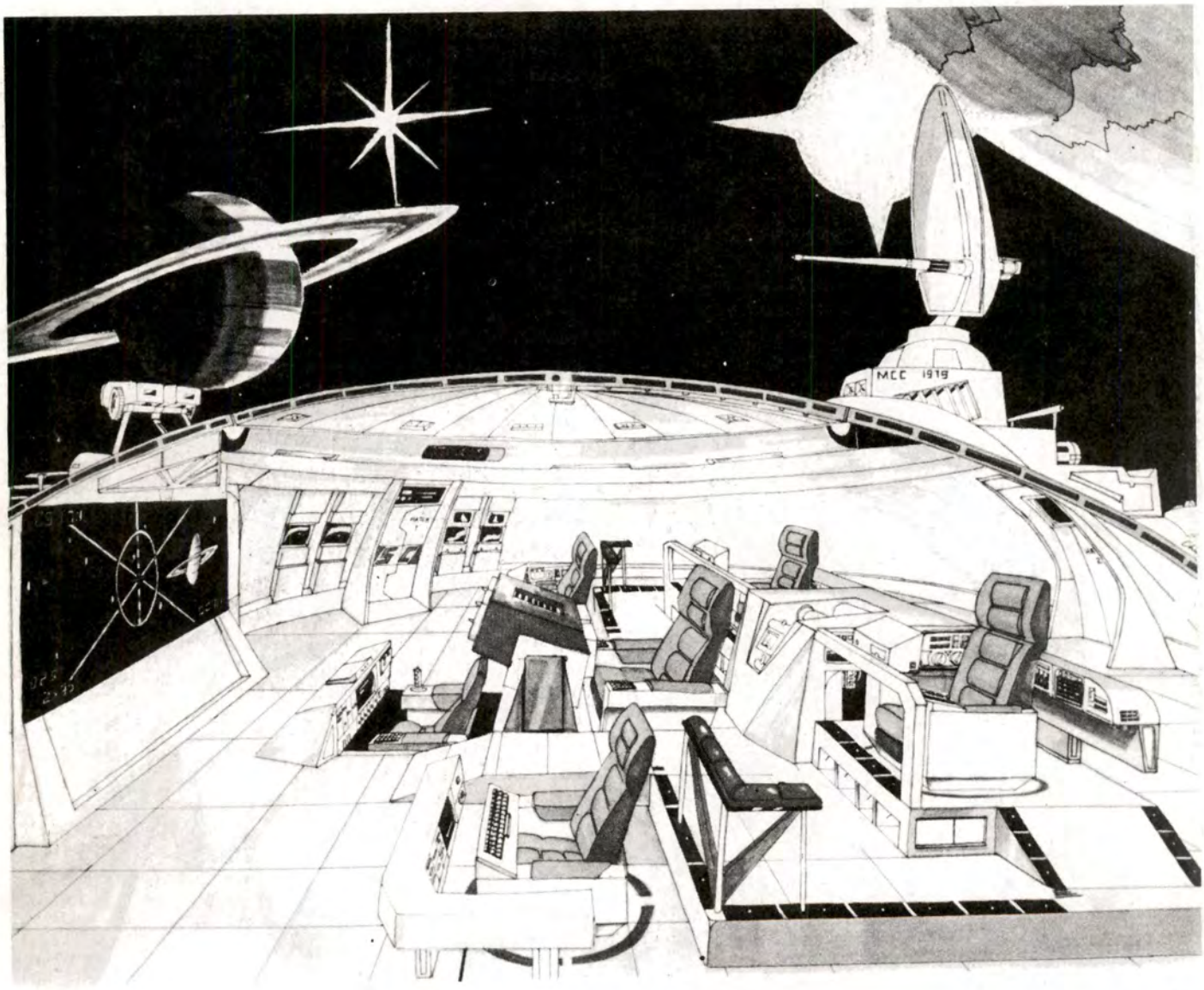
The Best of the Computer Faires
Volume V



# The BEST of the COMPUTER FAIRES

## VOLUME V

CONFERENCE PROCEEDINGS OF THE FIFTH WEST COAST COMPUTER FAIRE
San Francisco, California




Please see *The Starship Simulation Project* on page 34.

# THE BEST OF THE COMPUTER FAIRES, VOLUME V:
## Conference Proceedings of the 5th West Coast Computer Faire

Jim C. Warren, Jr., Editor

The Fifth West Coast Computer Faire
*held in*
The Civic Auditorium & Brooks Hall
*in*
San Francisco, California USA
March 14-16, 1980.

Computer Faire; (415) 851-7075

333 Swett Rd., Woodside, CA 94062

Manufactured in the USA.

International Standard Book Number: 0-930418-04-X
Library of Congress Catalog Card Number: 80-80515

Computer Faire; (415) 851-7075

333 Swett Rd., Woodside, CA 94062

PREFACE

This is the fifth time this writer has produced a preface to Conference Proceedings of Computer Faires. In that period of time, we have seen considerable change and maturing take place in the microcomputing industry and user community. And, as is reasonable to expect, that change is reflected in these latest conference papers. For instance:

The first several Proceedings contained a number of papers addressing how things might be or could be done. In this volume, many more papers address how things have been done. This is particularly true in such areas as educational computing, computer-assisted prosthesis for the physically impaired, and significant systems software.

In this volume, we see more emphasis on detailed, "hard" facts regarding specific applications environments, and less prognosis of possible futures, or reports on soon-to-be systems. We also see more emphasis on complete systems, and considerably less emphasis on hardware components or small subsystems.

This is as it should be, for the power and widespread use of low-cost computing facilities rests in their utility as complete systems, appropriate for application to real-world problem environs; not simply in their being brought into existence, bits and pieces at a time.

(In no way is this meant to depreciate the value of the significant and exciting hardware and subsystems work that has been done by the many innovative entrepreneurs and inventors in this field. It is simply meant to observe that the emphasis has now shifted to the development of mature and comprehensive systems, appropriate for application to significant real-world problems.)

Finally, it is worthy of note that each successive volume of these Proceedings has placed less and less emphasis on the hobbyist aspect of computing. The hobbyists have not gone away. Rather, they have simply become more and more diluted by the increasing numbers of applications-oriented manufacturers and users. And, more than a few of those who began as hobbyists have been transformed into manufacturers and software producers addressing the needs and desires of that growing community of users.

Though much of this volume consists of solid reports of work completed, it is interesting to note that at least several areas remain distinctly incomplete, with the papers being predictive and hypothesizing. (These are the kinds of papers that are often refused admission to the more reputable, "professional" conferences...but constitute a type of communication and basis for dialogue that this writer strongly feels is essential to continued and efficient progress.)

For example:

The papers on computers and personal or mass communications are almost completely predictive. Their predictions are becoming more and more believable, however. Up to this point, we have been busy developing reasonably economical, reasonably reliable, reasonably complete information processing systems. Now that they exist -- to a reasonable extent -- we have massive information processing power available to individuals ... but no reasonable access to massive quantities of information to process, at least, not in a useable (machine-readable) form. Telecommunications systems are certain to address this situation. And, now that the computing systems exist in reasonable forms, we may expect to see the rapid introduction of a wide variety of innovative communications systems and technologies addressing those systems.

The papers on legal, ethical, and business aspects of low-cost computing imply much more vague futures. These address human, rather than technological, issues for the most part. As such, they address problems for which the solutions are more difficult -- from the technologist's viewpoint -- and certainly less predictable. The papers addressing legal protection of proprietary rights make it amply evident that protection currently afforded by law is somewhat comparable to that available to prefeudal peasants. Much innovative work remains to be done in this area.

Jim C. Warren, Jr.
345 Swett Road
Woodside, CA 94062
80 February 17

# THE BEST OF THE COMPUTER FAIRES, VOLUME V:
## Conference Proceedings of the Fifth West Coast Computer Faire

### Table of Contents

# BEGINNERS, GATHER 'ROUND

or

# WELCOME TO THE SMALL COMPUTER REVOLUTION

Nicholas Rosa
Nicholas Rosa Associates
1901 S. Bascom Avenue, Suite 337
Campbell, California 95008

YOU ARE ATTENDING the Computer Faire because you have some reason to be interested in small computers. It may only be curiosity: you know that small computers now exist and you wonder what the fuss is all about. Or you have played with computer-game devices and understand that real computers can do even more wondrous things. You may already have been exposed to small computers and have some ideas about what you could do with them. But you feel you need to learn more. Perhaps you realize that small computers are destined to become, very soon, very important in our society. So you are here to learn more about them. You may be a small-business person, wondering whether a small computer might do something for your business. Most of you are wondering, "Should I get a small computer?"

Today you can have in your own hands the same amount of computer power that only a few years ago was reserved to the large corporations and to governments. You can buy it for the price of a household appliance, a color TV, or, if you get fancy, a car.

That's something to think about.

## Yes, Yes, Yes

Somewhere each of you has a bottom-line question about computers. The answer to everybody's question is a qualified Yes.

It's a qualified Yes because the computer cannot do anything for you all by itself. You will do whatever it is you hope the computer will do. You will work all the miracles. The computer is only your tool. In order to make it work its magic for you, you will have to understand the tool. You have to learn how to use it.

## But What Happened to That Genie?

That may sound discouraging. The very word "computer" already sounds so technical, so forbidding. Could you really learn to use this tool? The answer to that is Yes. You already use countless tools. You use pencils, you use sewing needles. You use typewriters and sewing machines. You run dishwashers and clothes washers and ovens-with-timers. You drive a car. Using each of these tools requires a certain amount of skill. You have learned that skill.

Even so, you may be holding onto a secret terror about computers. Everybody in modern American society is a little paranoid about computers. We all remember HAL from 2001. HAL was as smart as the astronauts aboard, but tricky, sinister. Evil. HAL fits our deep-down notions about computers. We all know that there's a computer somewhere watching us. Several computers. Lord knows how many computers. The IRS has got one (several, BIG ones). The Telecredit network has got one. Your bank has got one. All those computers are tigers, waiting out there in the dark.

Your personal computer will be a pussycat. Your pussycat. And some day---perhaps sooner than we can imagine---an army of personal pussycats may put those tigers to rout. For one thing, your personal computer is going to help you become computer-wise. That is going to make a difference. Out there.

Your personal computer will be no HAL. Your personal computer will just be a dumb box that sits there, waiting for you to command it to do something. You may find ways to have it working for you all the time, while you do something fun, something creative, something profitable. The computer won't---can't---do anything you haven't told it to do. You're the boss.

But---"Will I break the computer? You know, I'm so fumbly." No. There is nothing you can do to harm a computer by, say, making mistakes at its keyboard.

## But I Can't Even Type!

So? You never learned to. Besides, you

don't have to be able to type to use a computer. Yes, all those machines out there have keyboards. But a computer is patient. It will wait while you hunt and peck. Commands to the computer use only one or a few letters. The big thing the computer does for you is to process information. You will run much of that information into the computer from tape cassettes or magnetic disks. You will joyfully hunt-and-peck the rest.

## Just How Smart Are Those Computers?

You can dash out today and buy a computer, take it home, plug it in, turn it on---and the computer will sit there and do nothing. It needs instructions. So you type in, MAKE ME A HAM SANDWICH. It still does nothing. (You thought it was going to turn you into a ham sandwich?) It doesn't know what ham is, doesn't know what a sandwich is, hasn't the faintest idea of what you want or how to get it for you. Stupid? You betcha.

But all you will have bought is hardware. That's metal boxes. The electronic gadgets. The shiny stuff.

To do anything, the computer needs software. And software consists of programs and documentation. Programs are instructions for the computer, in computer code. Documentation is printed instructions for you, in English---on how to use the programs, how to put them into the hardware, how to make the resulting system do what you want. With software, the computer is suddenly very smart.

How do you get software? Buy it, borrow it, write it yourself---we'll discuss that later.

Most of the software you will use will be interactive. That is, it enables you and the machine to converse. You load an interactive program into the machine (from a handy tape or disk) and command it to RUN. Now, somewhere on that computer you have a TV-like display screen. On that screen, the program asks you questions. It offers a MENU. The menu lists the tasks the program can do for you, like balance your checkbook, write checks, figure automobile expenses, figure household budget, and so on. The menu tasks are numbered. The display politely asks you to choose a task and type in its number.

You comply, and the computer responds with a request for information, or another menu, or whatever it takes to get started on the task. At this point you may think the computer is telling you what to do. Actually, it is only PROMPTING you to feed it information. The program "knows" what it needs. So it asks you for it.

All those machines out there---all the hardware---are pretty much alike. Yes, they vary, the way cars vary: in workmanship and options and price. What's really important to you is the software. The best hardware can't do anything for you without software.

## "But I Don't Understand Electronics..."

Surprise, surprise. You don't have to understand electronics. You don't have to know much about the electronic technicalities of computers to use one. (How much do you know about the inside of your color TV?) But the technical buzzwords are in the air. Especially here. It would help to be able to understand some of the jargon.

What you really need to know is that there is a lot less to a computer than you may have thought. Suppose we take a look inside one.

## The Bare Bones

All computer systems, large and small, have three parts: an input device, an output device, and a central processing unit, or CPU. Suppose we just refer to that as "the CPU" from now on.

The CPU is the "brains" of the computer ---it really is the computer. The input device is usually a keyboard. It feeds data into the CPU. The CPU will process your data, and feed it to the output device---a printer, or a CRT display (a TV-like screen) or to both. A tape cassette or a disk drive unit is an input-output device. It can give the CPU information or accept information from it. (Cassettes and disks act as "electronic filing cabinets" for the computer.)

Think of the CPU as just a "black box." For now, you don't care what's inside it. It satisfies you that you can put information into this box somehow, and get information out. Something inside the box processes that information into the "answers" you are looking for.

The CPU does not work alone, however. It has helpers. It has memory. It needs another black box in which it can keep information before, during, and after processing. The memory "box" also holds instructions for processing the information: what to do with it, and how to do it. The memory, then, holds the programs that are in use.

The CPU, or "computer brain," has another helper: a clock. Everything inside the computer dances in step, to the quick beat of the clock. (Typically two million or four million "ticks" per second.)

The CPU needs a few more black boxes: interfaces. Interfaces are electronic circuits that enable the computer to work with its peripherals---its input and output devices. A computer is one kind of gadget. A printer is a different kind of gadget. The interface marries the two, sees that they get along. You are familiar with interfaces. Your car has one. Your car has an engine, and it has wheels. To connect the engine with the wheels, the car uses a boxful of gears: the transmission. An interface.

All these black boxes are wired together and fitted into one BIG black box, the computer cabinet. Typically, a personal computer's cabinet is about the size of a stereo receiver.

Some manufacturers of small computers mount an input device (a keyboard) and an output device (a CRT display screen) right on the computer cabinet. Maybe there will be a disk drive unit on the cabinet. But the computer proper is the CPU and its companion black boxes: the memory, the clock, and the interfaces.

The giant computers back at IRS and Bank of America and Telecredit are organized this same way. At the black-box level, all computers are alike. (It's when you pry inside the black boxes that any differences show up.)

That's all there is to any computer. (See? A pussycat.)

## Another Way to Look at It

Another way to consider a computer is as "an organized collection of simple switches." These are just on-off switches like the light switch on your wall, only the computer has thousands of them. (They're small.) They are lighting-fast switches, and they are automatic ---one switch can throw another. They are wired in a pattern that permits them to work as a computer.

In Computerese, if a switch is on, we say its output is "true." If the switch is off, we say it is "false." This is the beginning of computer logic, the electrical "thought-process" through which a computer does its work. We can represent "true" and "false" by 1 and 0 respectively. Doing it that way gives us a shorthand for describing what's going on inside a computer. It also gives us and the computer a pair of numerals to work with, 1 and 0.

You have heard that computers work in some mysterious way only with 1s and 0s. This is true, but not mysterious. This is not the time and place to explain how this zero-and-one number system, the binary system, works. Let it suffice here to say that the binary system can express any number by use of only the numerals 1 and 0. Also, any computer instruction can be written (coded) in combinations of ones and zeroes.

Is the binary number system too technical a matter for you to understand? Not at all. The fact is, you already understand it, provided you understand anything about the "everyday" decimal system that everyone was taught. I would like to prove that to you. I would dearly love to get into number systems right now with you. But you haven't time, really. It's something you can look into later if you care to.

## Forget It If You Want To

All that's important to you at this moment, for your present purpose, is that using the binary system permits building a computer out of thousands of simple on-off switches. Right now, no better way is known.

Don't worry! When you work or play with your computer, you will use familiar decimal ("ten-based") numbers, and the computer will give you back decimal numbers, unless you instruct it to do otherwise.

You needn't be concerned about that "worrisome" binary system, unless you want to learn sophisticated, deep-level computer programming ("machine language" programming) or plan to get down into the computer hardware in a big way, perhaps study computer design. Then you'll learn as you go.

That should be stressed. If you only want to use the computer, not get deep down inside it, you may ignore the binary system. The only reason I brought it up was to explain why a computer can be, is, must be, a collection of on-off switches.

And that was only so I could tell you:

## How Come?

How come small computers? How did they come into being?

Computers have been around for a little over thirty years. The very first all-electronic digital computer, ENIAC, completed in 1946, was the size of a house. It cost millions. With progress, computers came down to "big room" size. Then, in the 1960s, came something called the minicomputer. That was typically the size of a freezer chest. Oh, they got minis down to washing-machine size and even the size of a big desk drawer. But like the giant machines (still being made today), they cost. A typical price for a

mini system is around $100,000.

Suddenly---in the past few years---you can have all the computing power of the old main-frame giant computers and the big-ticket minis. And for a few hundred to a few thousand dollars. Anyone can buy this powerful but small computer for the price of a household appliance. The small business can buy a more elaborate system for the price of a new car. Does a small business need a computer, like B of A does? It sure does. And now it can have it.

It used to take "experts" to run a computer. Quite a team of them. Today, you could operate a small computer system all alone, even with what you know right now.

What happened?

## A Little History

That first computer, ENIAC, in 1946, used vacuum tubes---what your grandparents called "radio tubes." A computer is a collection of switches, right? These need to be very fast switches, and automatic. In 1946, vacuum tubes were the fastest switches possible.

ENIAC had 18,700 tubes. They ran hot. Like light bulbs, they tended to burn out. Vacuum tubes use considerable electric power. In total, ENIAC used 150,000 watts of power.

In 1948, Bell Labs begat the transistor. This was tiny---pea-sized. It did not run hot. It ran on a whisper of power. It did not burn out. It became available in quantity only in the mid-1950s. It was first widely used in hearing aids and portable radios, not in computers. Not many computers had been built until the mid-1950s---it had taken some years for industry, business, and government to catch on to their value. But computermakers recognized something good in the transistor. They used it. Computers became a little smaller and ran a little cooler. They became cheap enough for big corporations to buy. But they still cost a lot of money. Transistors themselves, however, kept getting cheaper as production runs increased.

## Introducing the Chip

Transistors are made on thin tiny flakes of pure silicon metal, called chips. By about 1960 somebody worked out a way to make two or more transistors on the same tiny chip. Not only that, ways were found to make tiny circuit components---resistors, capacitors and the like---on the same little chips, with the transistors. Thus was born the integrated circuit. Before this, an electronic circuit consisted of one or two vacuum tubes (or transistors), plus other parts, on a fairly large---at least postcard-sized---circuit board. With the integrated circuit, the entire device (all the parts) was on a single chip of silicon about the size of a baby's little fingernail.

By about 1969, it was possible to have over a thousand transistors and so forth on one chip. Thus came large-scale integration, (LSI).

A few years later, LSI achieved upwards of 30,000 transistors on a chip. Now they had put more than the equivalent of ENIAC's 18,000-plus vacuum tubes on one eensy teensy chip!

All those transistors (or "switches") on the chip can be wired up in any way the chip designer wishes. With 30 or 40 thousand "switches" on a chip, designers made the microprocessor----the brain-circuit for a potentially powerful computer.

Memory circuits also went onto chips. A memory chip is just another collection of thousands of switches. Each switch that is "on" stores a 1 and each switch that is "off" stores a 0.

All right, it's obvious how it became possible to make computers small. Small computers just use chips. For everything. But how did they get inexpensive?

It's simple. Whether you make a chip with one transistor on it, or with 100,000 transistors on it, the fabrication costs are about the same.

So there you are.

## And So, the Microcomputer...

The microprocessor chip made possible the microcomputer. All the computers out there on that display floor are microcomputers. Thirty years ago there were only one or two giant main-frame computers. Fifteen years ago came the minicomputers. Today there are microcomputers, and "everybody" can afford one.

The microcomputer was not the child of the computer industry, but of the semiconductor industry. (Semiconductors are a family of metals, including silicon, that conduct electricity in strange but useful ways.) The semiconductor people keep trying to put big-

ger and better circuits on the chips. About every four years they produce a new generation of more complex chips. And about every four years we ought to see a new generation of microcomputers and microprocessor-controlled devices.

As more and more people buy microcomputers, chip production runs will get bigger---and per-unit costs smaller. That's why prices keep coming down. That's why electronic devices (computers, stereo, whatever) hold their own against inflation. In a few years you will be able to buy more computing power for less "real" money.

However, the microcomputers you can buy today already have more computing power than you could put to full use through the next four years. If you need or want one, why wait?

If you are a businessman, and a microcomputer (or any computer) can make or save money for you now, you are losing money by waiting, and you will never earn back the money you have lost. That has been proven by a sophisticated dollars-and-cents analysis. It's straightforward, easy to understand, but is another one of those matters for which this is not the time and place.

A Look at Software

Only a couple of years ago, you could come to one of these microcomputer shows and see lots of exciting hardware, but little in the way of meaningful software. There were some computer-games programs and a few primitive business packages.

Today there are a few bigger and better games but many bigger and better business packages. For the past two years, in fact, business needs have provided the main thrust in software design. No business package is perfect because no two businesses are alike. But you can get good "starter" software that will do some things for you (perhaps payroll or accounts receivable) in a form satisfactory to you, and you can have other parts (order entry, inventory, whatever) modified to suit your exact needs and preferences. You must analyze your needs---start off by analyzing your business---figure out what you need to have a system do, and shop for the software to do it.

Everyone should remember that software ---the program---is what does the job for you. The hardware is merely something the software runs on.

Hobbyists often write their own software. It's not really hard to do, in itself. Of course a complex task requires a complex program and takes effort, knowledge and skill. But hobbyists have come up with some fine programs. A few years ago, writing your own was about the only way to go. Then microcomputer clubs and magazines came along, and began spreading software around. You'll find general computer clubs but mostly you will find "user" clubs: Apple user clubs, Horizon user clubs, PET user clubs, and so on. If you want to learn about computers, both hardware and software, and fast, join one of the clubs. The nearest computer dealer will tell you what clubs there are around and when and where they meet. (Often, they meet in a dealer's store.) The dealer also sells computer magazines and books, and you should be reading.

It's worth a businessman's time and effort to learn something about computer languages and programming, too, or to have someone on the staff who understands them. (Not, "someone on the staff who understands electronics.") It's a matter of understanding the tool. It's a matter of realizing the computer's capabilities and limitations. You will need to be able to communicate with ---and size up---any programmer you may hire, part-time or full-time depending on your needs.

The prevailing microcomputer language is BASIC---for Beginner's All-purpose Symbolic Instruction Code. Don't sell BASIC short because of that "beginner's" part. Powerful programs have been written in BASIC, including many good business programs. BASIC is a "high-level" language, "close to English," close to human thought. (A low-level language is close to computer thought, and deals in 1s and 0s. Hard to learn, hard to use.) Other high-level languages, such as PASCAL and FORTRAN, are coming into use in microcomputers; these are more powerful than BASIC and a little harder to learn.

Taking care of a computer, at home or in the office, mostly involves some common-sense precautions. Maintaining the system as a system requires an understanding of software. You will benefit from everything you can learn about programs and programming, but you will not benefit much, if at all, from what you might learn about electronics. A full system, especially a business system, is not designed to be tinkered with. It is intended to be used like an appliance, with its cabinet buttoned up. Keep the family (or office)screwdriver maniac away from your computer system.

This is NOT to say that learning

electronics is not useful in itself, especially digital electronics---the electronics of computers and related devices. This is NOT to say a person expert in, or even interested in, electronics should not get interested in computers. I hope I did not give the impression a minute ago that all computer hobbyists were (perforce) programmers, and programmers only. Microcomputers are a natural for the electronics hobbyist, for people like the radio ham. Hobbyists have built entire, fully respectable computers from the ground up. Many, many more have built them from kits.

My point was merely that you don't have to "understand electronics" to use the computer plus the warning to keep casual tinkerers our of your equipment. (Treat it like your color TV: you don't let the family screwdriver maniac inside that, either.)

## Approaches

For learning the electronics and digital side of computers, kit-building is of course a great way to go. Kits are available for everything from a self-teaching "microprocessor fundamentals" kit (the end-product of which is actually a small computer, though limited in application and ease of use) for fifty or a hundred dollars, to complete, versatile systems costing thousands. If you have never built an electronic kit before, buy something smaller and reasonably simple to practice on. Start with some sort of voltmeter kit, perhaps---you'll end up with a valuable test instrument, useful for when you build the computer. Don't buy the expensive full-power computer system kit if you have never even used a soldering iron; you will ruin too many expensive parts and circuits while practicing. But with a little practice anybody can build kits. The best kits to start with are "course kits" that teach the fundamentals of electricity, electronics, and electronic equipment building practice. Then you should let yourself "graduate" to a computer kit.

A growing group of computer users is not interested in electronics or in equipment per se but just wants to be computer users; these people don't care for the fine points of programming, either. At least, not right now. They just want applications: from self-teaching (about anything) to running a house (furnace, air conditioner, burglar alarm, baby minder, lawn sprinkler, budget and so on). They don't want to have to program and to an increasing extent they don't have to. They are like the businessman who just wants to run his payroll and inventory and doesn't give a damn about the nuts and bolts. All these people have a legitimate need or desire. They can, will, and must be served. All they need is more, and more versatile, programs to be written and sold to them (delivered on a disk or tape, as most programs already are). More and more of that is coming, as professional programmers get past the peak of the business-software demand, and as creative amateurs keep ginning good things up and publishing their programs in the computer magazines.

There is something in the small computer for everybody.

## Revolutions, Revelations

A computer is a teacher as well as a tool. It can be used for teaching or self-teaching about computers, computer languages, and programming, certainly. It can be used for teaching math or science or economics or history. It can be used to teach anything a program can be written for. It can just, in a sense, teach creativity. Just fooling around or doodling on the CRT screen with simple graphics elements can suddenly produce patterns of startling beauty.

You can even play and write music with a computer.

Or you can keep track of a warehouse inventory or run a payroll or just keep your personal date book. A computer, like a human being, is unspecialized and can be taught to do "anything." If you are a writer, a lawyer, or a secretarial service, a computer will do word processing for you: speed up your productivity immensely, ensure you perfect error-free copy. Or you may use the computer as a blackjack dealer.

Long ago there was an Industrial Revolution. That was followed by a succession of other revolutions that resulted in the world of today. Computers are helping shape the world of tomorrow. We are all still vibrating with the shock of the Big Computer Revolution that came along less than one scant human generation ago. And suddenly we are in the midst of the Small Computer Revolution that is going to change things even more markedly. And that revolution is for you.

## About the Author

With his partner and wife, both of whom are named Sharon Rosa, Nicholas Rosa is co-author of "Small Computers for the Small Businessman," from dilithium press, Portland, Oregon, 1980. When the Rosas are not working with computers and writing about them, they do outdoor things, like organizing charter boat cruises to watch California gray whales.

# AN EASY APPROACH TO OPERATING SYSTEMS...
# FOR EXAMPLE, CP/M
# (FOR BEGINNERS)

by Tony Bove
(Author of forthcoming book on CP/M)
SYBEX, INC.
2020 Milvia St.
Berkeley, CA 94704
415-848-8233

## Why?

Using a computer should always be easier than not using a computer.

As small computers invade the small business market, many ordinary people are bumping into operating systems that were patterned after the inscrutable systems of the past.

Many introductory books on microcomputers choose to teach binary arithmetic, flowcharting, and assembly language before explaining how an operating system works. Since most small business installations come already equipped with programs, device drivers, and an operating system, most ordinary non-technical people bump into the operating system first.

If you walked into a stereo showroom with your own cassette, you could easily manipulate any of the controls on the latest and most expensive cassette recorders. You should also be able to walk into a computer store and operate an operating system. You should know what to look for in a system, and know what kind of application programs would fit well with a particular operating system.

This paper will describe methods of teaching an operating system to "naive users," and outline typical operations in a system using CP/M (and the latest MP/M) as models. During the Faire, I would like to simply demonstrate some of these methods with ordinary people. If you are a total beginner to computers, you should be able to understand and even teach the fundamental operations of a system like CP/M.

## What Is CP/M (and MP/M)?

CP/M stands for Control Program for Microcomputers, and MP/M stands for Multi-Programming for Microcomputers. Both are compatible operating systems from Digital Research, Pacific Grove, CA.

## Why CP/M?

People, especially in business, want application programs tailored to solve specific needs, but they frequently end up with programs designed to satisfy general needs. These same people need operating systems that are similar and compatible, and they end up with unique "patched" systems with incomprehensible documentation that only technicians and wizards can operate with ease.

CP/M is a general all-purpose operating system that grew up fast and became almost a "standard" for small microcomputer systems. Almost overnight, companies seized the opportunity to sell CP/M-compatible software (mostly business applications), and CP/M grew in popularity.

I am using CP/M as a model not because I like CP/M. I don't find it to be easy enough to use. I am using CP/M because you probably already have it, and you can't use the existing documentation to teach yourself or your associates.

I am not endorsing CP/M, I'm just trying to prepare ordinary people (like myself) for operating systems of the present and future. I would only endorse an operating system that is so easy that a beginner could do something useful in less than five minutes.

## What Is An Operating System?

If you wrote a program to operate a device like a diskette drive, or a printer, and more programs to handle files, allocate storage space, supervise other programs and monitor the terminal and answer requests, and if you stacked those programs together and wrote a program to manage all of the programs, you would have created an operating system.

Luckily, you can buy one and save a lot of time and money.

The operating system is like a butler, waiting at the terminal for your commands, ready, willing and able to perform all of

its preprogrammed functions. Most operating systems, however, are not as polite, though they could be.

When you walk up to a terminal that is hooked up to a computer, and type a command, you are using some form of an operating system. Even if you type gibberish, your gibberish will be scrutinized by the operating system, because it will assume you are giving it a command.

Try it. Type gibberish into your terminal, and watch the computer respond by repeating what you've typed with a question mark (this will actually happen in CP/M). One of the best ways to ease people's fears of computers is to show them that no matter what they type, they can't hurt the system. They can only hurt their own files and diskettes.

Operating systems usually work this way: the system reads what you've typed, and if it doesn't already know the instruction (i.e., built-in command), it tries to find a file by the name you typed. It then tries to execute the instructions in the file. If the file is actually a program, then the computer will actually do something. If not, then you will get an error message. If you typed a word that is not a filename or a built-in command, it would return simply with an error message.

You should be able to type anything into any operating system and experiment with unknown operating systems. If you inadvertantly destroy the system, then the fault probably lies with the system. One good way to test an operating system is to try to abuse it before you buy it.

The best way to learn how to use one is to practice doing useful things. Soon there will be many kinds of operating systems that look similar but are designed for specific needs and tastes. They will become what people want them to become. They will not force people to comply with archaic formats and oppressive structures.

CP/M is an example of a simple operating system that handles one terminal. MP/M differs from CP/M in that it handles up to four terminals, each one doing different things. MP/M is a more complex program that treats each terminal as a "task." It schedules tasks so that they don't bump into each other, or take up too much time, or monopolize any resource. A program that does this is called a task scheduler. Every bureaucracy needs at least one.

The next higher level of complexity after a task scheduling system would be a "real-time" system on a minicomputer.

## What Do I Do With An Operating System?

The most popular activity is executing programs like a BASIC program, a word processing program, or a data entry program. In fact, your program could be a word processor written in BASIC for data entry. If the program is "ready to go" (or "assembled," or "compiled," or "installed"), you should only have to type the program's filename. In CP/M, you type the filename followed by a RETURN. In other systems, you might have to type a command like EXECUTE or XEQ or RUN.

If the program is not "ready to go," then it is in some intermediate shape--not quite the final product. A programmer might ask you to "assemble" or "compile" the program, and then run it (or copy it onto another storage medium, like another disk). You would use the operating system to do all of these things.

You use the operating system whenever you want to see a display of the files on your disk, and whenever you copy files and send files to devices like the printer or storage devices like a cassette recorder.

Most of these operations also occur on big computers. That is because a system usually acts like other systems. If you read about operating systems on other computers, you'll find that they're not too different from systems involving large computers with gigantic databases.

In fact, your microcomputer might already have the capabilities of larger and more expensive computers, and you might have a large database in mind.

A computer is really a general purpose machine that runs other machines--disk drives, terminals, displays, printers and more. The operating system runs these devices and sends them information to process. You, however, run the operating system. At first, you only need to know a few commands and how to use the terminal keyboard. After awhile, you should also know how to manage available memory, submit a batch of commands to the system for automatic execution, and understand the mysterious error messages. When you learn what the error messages "mean," you'll find it easier to talk to service technicians and get answers to specific problems. Finally, you should learn high-level programming languages as well as assembly language so that you can modify your own system to make it work better and to suit your tastes.

To learn these things, you need to have patience and you need to practice. You also should get familiar with the documentation supplied with your system, even if it is inscrutable. You can and should grasp the fundamentals of operating systems,

but you will eventually have to rely on the manufacturer's documentation for more accurate information.

How Do I Use Documentation?

A typical operating system manual will start by explaining in technical detail the bells, whistles and features not usually found on other systems. These manuals usually assume the reader is someone who knows how to program computers and who wants to know immediately about the far-out features.

Since most readers do not even know what the technical details mean, they tend to skip over these introductions and get right into the abstract titles of the table of contents. These chapters are usually designed to be replaced easily, in case they want to change the operating system frequently. If you recognize a heading's meaning, you should go directly to that chapter.

Somewhere near the beginning of the manual there should be descriptions of important keys on the keyboard, under the heading "Using the Keyboard" or "Keyboard Editing" or "Single-Key Commands." The manual should describe the use of keys like RUBOUT (DELETE) and RETURN and sequences of keys (like CTRL and C) that perform special functions. Practice using these important keys.

Operating systems have commands, and they usually take a special form (called a format, or command syntax). Find the general description of this special form. Usually commands take this form:

| Subject (understood) | Predicate (operation) | Object of operation |
|---|---|---|
| (Computer1) | ERA | THISFILE |

where "ERA" is the erase command, and it erases the file THISFILE.

Each command will be described in detail in the manual, with a format that symbolizes the actual words you would type.

Only manuals with good indexes will help you find out about a specific topic. Look up the entry "copying files" in your manual's index to see if it is any good.

To get familiar with your manual (and to get other people familiar with it), look up commands and practice them. When you perform a command perfectly, show your associates how to perform the command, and show them the information in the manual even if the technical jargon is hard to read. Eventually you will get used to the detailed descriptions.

How Do I Approach An Operating System?

Someone should take you by the hand and lead you through the simple operations. You should then be able to lead someone else through these operations.

First, turn on your equipment and "bring up" the system. Every system has a different way of being "brought up" or "bootstrapped." The term "bootstrap" came from the idea that if you were strong enough, you could "pull yourself up by your bootstraps." Actually, the system "pulls itself off the diskette and starts itself."

Once the system is up and running properly, it tells you that it is up and running and waiting for a command. It tells you by displaying a system prompt. A system prompt is a character or group of characters usually displayed on the left hand side of the screen. Every system has a prompt.

Immediately, you should find out what you have to do to bring the system down again, in an orderly fashion. Then find out how to stop the computer from doing something over and over again, infinitely. This is sometimes called "pulling the plug" or "interrupting execution." Knowing this will make you feel better about experimenting.

Now, you should tour the keyboard. Try some of those special key combinations. For example, in CP/M, you can turn on the printer (if the printer itself is on) and have it print everything you type on your terminal and everything the system displays on your screen--by hitting CTRL and P simultaneously. Hit them again to turn this "printer echo mode" off.

After playing with the keyboard, you should try an operating system command. In CP/M a good command to start with is the DIR command. DIR stands for Directory. You can find out what files you have in your disk by asking for a directory of the filenames. The characters "A)" form the prompt displayed by the system to tell you that you are looking at disk drive A:

A)DIR↵

(The ↵ symbol represents the RETURN key.)

This command would display the entire list of filenames.

You should create a file as easily as possible--using an "editor" program like ED supplied with CP/M, or using a word processing program like Microsoft's ED-80 or another ED-80 from SDT, Inc. It doesn't matter what the file contains or what editor program you create it with. Editor programs create and edit text files.

You should practice making copies of this text file and copying the file onto another storage medium. This is called "creating a backup." If you have a simple diskette drive system with CP/M, you would use the PIP command to copy files from diskette to diskette.

Most CP/M commands are actually programs that have been "assembled" and "LOADed" (using the CP/M LOAD command) and are "ready to go." They exist as files with "COM" as the filename extension (e.g., PIP.COM). To execute a "transient" command, as these commands are called, you have to have the file on your diskette. If it is not on the diskette you are currently looking at (shown by the system prompt), you have to specify the diskette drive's letter with the filename (e.g., B:PIP for PIP if it is in drive B). You can therefore execute any command from a different drive.

You can also use PIP to transfer files to devices like the printer. Other systems have different commands to send files to the printer, but CP/M uses a variation of the PIP command. CP/M's PIP looks like this when you copy a file:

PIP newname=oldname

PIP looks like this when you copy to another diskette:

PIP d:newname=d:oldname
PIP d:=d:*.*

where d stands for the letter of the drive. If you don't specify a newname in either example, the copy file keeps the same name. Be careful, however, because CP/M isn't smart enough to know if you are creating two files on the same diskette with the same name. If you do create two files with the same name, you can't access either of them.

Use the manual's examples to follow the formats for PIP. Other systems have easier formats, like this:

COPY oldname newname

If you have a word processing program, you can probably execute it, and it will make copies and send files to the printer for you.

If you have any other software, try it. For example, if you have Microsoft BASIC and some games, run them. BASIC is a programming language that is easy to learn. Games are very easy to play, you don't have to know anything about BASIC except how to execute a program (usually the command RUN).

To "bring up" BASIC, you execute the command supplied (usually MBASIC.COM, so you'd type "MBASIC↵." BASIC would come up with its own prompt (usually "OK"). You would then type whatever is appropriate in order to load and execute the game.

## How Do We Become Friends?

You will grow in your understanding of systems. You will realize, for example, that the system has "generic" names for certain devices, so that you can make the system think a cassette recorder is a card reader. In CP/M, the LST:, CON:, RDR: and PUN: names can apply to different devices. The STAT program can switch devices and display possible values for the device names. Most likely you will want LST: to be your printer (output only) and CON: to be your CRT screen and terminal (input and output).

You will also learn how to "batch" several commands together with arbitrary arguments you can change later, and execute the batch of commands automatically (in CP/M you use SUBMIT and XSUB). You will also know your limits of available RAM memory--the scratchpad memory where programs execute inside the computer, and the limits of your disks.

If you grow in terminals and want to use MP/M, or a similar multi-terminal operating system, you will be able to run up to four terminals at the same time, all running programs. You will learn how to assign priorities to tasks to prevent heavy computing programs from hogging the computer while other programs wait for information from a device.

One good practice is to always write down the circumstances surrounding mysterious error messages. Many are not documented properly. Learn the circumstances so that you can communicate easily with the service technician, who is more apt to help someone who can clearly state the problem.

You can always hound the company you bought it from, and maybe they'll put together more documentation.

If you finally get around to programming, learn many high-level languages in order to create scientific and business applications. Start with BASIC and look at languages like PASCAL, APL and FORTRAN. There are many books on languages.

After using the operating system for some time, go back and re-read the introduction of the documentation for using the system, and read the introductions for the interfacing guide and system alterations. You'll see that operating systems are divided into segments that control different aspects of the computer system. By learning assembly language for Z-80 or 8080 based systems, you could alter these components of your operating system to suit your needs.

Hopefully, if you choose to alter your system, you will make it easier and not harder to use.

# THOUGHTS WHILE WAITING FOR THE CALVERY TO RESCUE ME

Tony Severa, 131 Highland Ave. Vacaville, Ca 95688

Many people are looking at the micro-computer and playing with the idea of quitting their secular jobs and going into business for them selves with their micro. Tony Severa dropped out of the 9-5 job market to go into business for himself in 1978. He opened Tony's Data Service in Vacaville, California in March of that year. He has since started a new software company for the Apple computer for beginning owners called APPLE ORCHARD.

One of the reasons I decided to invest my time and money into the micro-computer field, 2.5 years ago, was the challange and excitement of jumping into a field that is new and rapidly growing. I find myself relating my experiences to that of a pioneer; a person who has decided to step away from the safety of the crowds and move into a new, challenging, fast-paced world complete with bonified dangers.

As a pioneer I have come to realize that my most important ally is information. I find that the dangers that lurk behind each bend I travel to be caused by the lack of correct, explicit, usable, at hand information. Because the field is so new and new developments seem to come in continuous waves, I have found that there is no one specific manual, or person that I can rely on to provide me with the answers I need if I am to survive the wilderness and stay out of danger.

In September of 1978 I decided to quit my 9-5 job as a Drug and Alcohol Counselor for a county operated mental health program. I wanted to go into the wilderness and set up a data processing/word processing business. I wanted to make a reasonable living working with this new tool called the Personal Computer. I also felt that since I literaly read all the magazines as they came out (no small task) and had made many contacts in the hardware and software fields that I had knowledge that I would be able to share with others who had less knowledge than myself.

I had bought a Sol20 computer and installed 40K of memory and bought a HeliosII disk system. My expenditure up to this point was in the range of 7000 dollars.(this included magazines, books, pamphlets, equipment and software) Since any good pioneer knows that having a backup in hand is worth more than two maintenance contracts 60 miles away at 12 percent of the original cost of the equipment, I decided to buy a second system.

I invested over 9000 dollars more for another Sol20, a HeliosII, 48K memory and a Diablo Hytype II printer. I armed myself with the phone numbers of the closest computer stores, especially the one I bought the equipment from. I must say at this point that they had always been there to assist me in any way possible when I bought my first Sol20 as a kit. But now I was about to ventur into a new and dangerous territory. They seemed willing though, to back me up with information and advice and repair even though I didn't have one of those expensive maintenance contracts.

I was blessed enough to find an experienced programmer who was willing to work with me and teach me some of what he knew. He was never more than a phone call away and most of the time was sitting beside me or at the console himself trying to figure out the system's limitations and capabilities.

Several phone calls and weeks later I was ready to venture out into the wilderness of computers and small businesses. Although the magazines basically agreed that the Sol20 was top of the line in the micro field; it still was a baby less than two to three years old and I was one of the first people to try and make a living off of it. One of my first encounters with danger was while posting an accounts recievable package for a local counseling center. Everything seemed fine, but, instead of writing the data we wanted to each of the records and updating the journals on the disk, the computer had written a fine assortment of jibberish into many of the records.

Needless to say, the system crashed and I had to shut it down. "No problem", I said to myself, as I clutched my list of ally's phone numbers, "I've got a backup just incase something like this would happen."

I replaced the memory boards as per instructions of my friends at the computer store. I turned on the system and I called my programmer over for moral support and technical assistance.(I wasn't sure if it could also have been a software problem.) Everything seemed fine so I ran a check of the R.A.M. and it seemed ok. I then ran a check of the Sol20 and Helios and they seemed ok. So, thinking the coast was clear I looked at the damage done to the data on the disk.

Well, it was definitely clobbered and only 1/3rd of the data was left in readable condition. So, being very cautious I placed the back-up copy into the slot to make another copy of it, just to be safe. Sure enough, three quarters throught the copying it crashed again. Not only the copy was clobbered but so was my back-up copy. (MORAL:Make two back-up copies of everything.) It took 18 hours to piece together all the cloberred data files before I was able to go on.

I must acknowledge the helpful presence of my programmer. He stayed through with me to the end. I cannot overemphasize the importance of having someone to support you during times of danger. The result of all this was three bad,or even worse, erratic Processor Tech 16k RAM boards. Repair was made by the dealer I bought them from, but they implied that these boards were ready for the dump due to what seemed like a fault in the design. Great! I only had 5 of these boards in my system!

The dealer did what they could to assist me. I made a deal with them that allowed me to exchange my bad Processor Tech boards for repaired Processor Tech boards on a one-for-one basis. The boards for some reason seem to run fine until the ambient temperature goes up to 80 then they started dying like flies during an aerosol attack.

Recently, I encountered one of the most harrowing experiences that a pioneer in this field could imagine; Processor Technology had shut down and was out of business. Then (you guessed it), three of my Processor Technology 16K RAM boards failed during a three week period and I was left with no usable memory to operate my system. My dealer said they were unable to repair these anymore (besides, they usually had to send them to Processor Tech) and,to their knowledge, no one was willing to even try. A couple of thousand dollars later and some Econoram memory boards and my system has been running with a clean bill of health.

My most recent encounter with the dangers of being a pioneer came when I and a friend bought D.C. Hayes Modems for our AppleII computers.

Now, the skuttlebut about the D.C. Hayes Modem designed for the AppleII was good to excellent, so, we each bought one. We wanted to be able to transfer our programs from one Apple to another. We put our money down and got our modems and off we went with our new products in our arms. Once home we read diligently the manual that came with them. I started to get worried when I saw the note on the cover which indicated that this was a "Preliminary" version of the manual. First of all there were no pictures where there should have been pictures. There were no figures where there should have been figures. There were no diagrams where there should have been diagrams. But, we tightened our belts and with text in hand we tried to call each other up. It worked the first time. Hurray! But, it wouldn't do what the book said it was capable of doing. The main problem was that it just wouldn't transfer programs from one system to another (the very reason we bought it).

The dealer didn't know anything about it. He did have some suggestions and two weeks passed with no success. The system would lock up and we'd have to disconnect. Then after a call to the manufacturer we found that we needed to type in a certain call code to assist us and all would be ok.

That same day, while watching another person using his D.C.Hayes Modem, I observed him doing the exact thing that I was not able to do with mine; transferring a program from one Apple computer to another Apple computer. We compaired Modems. His had a ROM chip that was dated 1979 and mine was dated 1978. His manual was white with pictures and figures and diagrams and programs that showed how to transfer programs and mine had none. I was operating with OUTDATED EQUIPMENT and SOFTWARE!

I ran to my dealer and he hadn't recieved any updates or revisions or notices from the manufacturer. The manufacturer said that they would send the updated chip if I would send them the old one. And so, another week went by while we waited for the Post Office to do their thing. Finally, a call from my dealer and I was headed 35 miles to have them install my new chip. Home again to try the system and, you guessed it, it still didn't work. This time it was a different kind of failure but, a failure, none-the-less. Another call to the manufacturer and low and behold they had made a change in the ROM that made it necessary for the user to type in the simple command PR#0. My old chip did it and the new one didn't. They finally decided to send me a new manual (it didn't come with the new chip) and supposedly the information I desperately needed was in the new manual.

With this new information in hand we again sat at our homes and tried to communicate. After some errors and some calls to the manufacturer to ask those questions that would be answered by the manual when it came, if it came, we finally got to the point where the modems worked. I would say that this only cost me five 60 mile roundtrips to the dealer and 4-5 phone calls to Atlanta, Georgia to the manufacturer and approximately 2 months of total frustration and depression, lots of depression. Yes, being a pioneer means that you are out there all by yourself and you can't expect the cavalry to be there everytime you need them. I sometimes wonder what would have happened to me if I hadn't found out about the new chip and manual. By the way, the modem still doesn't work as the book specifies and there are several of us owners getting together to try and find out what is going wrong within our boards.

Out of all this, though, comes some ideas on how to handle yourself when in the wilderness and especially when you're by yourself:

A. BEFORE BUYING A PIECE OF HARDWARE HAVE THE DEALER OPERATE IT ON YOUR SYSTEM. If at all possible bring your system in and have them install it and operate in there, so they can solve all the problems BEFORE you go out into the wilderness.

B. GET AS MUCH DOCUMENTATION, USER REPORTS, NAMES AND PHONE NUMBERS OF OTHER USERS YOU CAN CONTACT FOR ASSISTANCE, AND THE PHONE NUMBER OF THE MANUFACTURER. Don't be afraid to call someone up and ask for advice. They may save you a large amount of time and pain.

C. TRY AND GET MAINTENANCE DOCUMENTS AND, IF POSSIBLE, TEST PROGRAMS. By doing this, if the company goes out of business (and they do go out of business, dont they), you at least have some way of possibly getting your equipment repaired if something, heaven forbid, goes wrong. No one will work on it without specs.

D. IF YOU ENCOUNTER TROUBLE, PUT YOUR DEALER AND MANUFACTURER ON THE SPOT AND MAKE IT CLEAR TO THEM THAT THE PROBLEM HAD BETTER BE CLEARED UP OR YOU WILL BEGIN TO WRITE LETTERS TO ALL THE MAGAZINES AND COMPUTER CLUBS. Manufacturers and dealers need all the GOOD publicity they can get and any amount of BAD publicity can hurt them where it can do damage.

E. STAY IN CONTACT WITH USER GROUPS AND CLUBS AND LET THEM KNOW WHAT'S HAPPENNING TO YOU. One of the best and most satisfying information links is the one you get from other users. They can provide you the support, ideas and suggestions to help you deal with all sorts of dangers you may be encountering in this field. Use them and be willing to help others.

F. TRY AND FIND OTHERS WHO HAVE SIMILIAR PROBLEMS AND JOIN HANDS IN TRYING TO FIND THE ANSWERS. Misery loves company and if you can connect with other people who are having similiar mizeries then you can join hands and be able to exert much more energy and influence.

G. LAST, BUT NOT THE LEAST, CHECKOUT ALL THE PROGRAMMERS IN YOUR AREA AND CONNECT WITH ONE OF THEM FOR MUTUAL BENIFIT. The local colleges are producing all sorts of programmers and all you have to do is talk to some of them. Find one that seems to like what you are doing and is willing to work with you for a price that doesn't rip off him or you. He can use the experience and you can use the moral support.

Ah, life in the wildernesss; it isn't so bad. It's very exciting and you feel the freedom only working for yourself can bring. It provides for you a challange that few people ever take on. You can do OK in the wilderness IF you take care of yourself by following some of the suggestions above.

# MICROCOMPUTERS AND THE DESIGN OF CONTELLIGENT SYSTEMS

© 1980 Dean Gengle
P.O. Box 14431
San Francisco, CA 94114

ABSTRACT

Contelligence is the combination of consciousness and intelligence. The linking of human biocomputers with microcomputers defines a particular system. By virtue of the human side of the linkup, we know that the system possesses contelligence. We have all heard of programs combined with computers that teach various biofeedback methods: alpha/theta production and control; myography; temperature/blood pressure control; etc.

There are many esoteric traditions which have taught "altered states of consciousness" methods for spiritual and sometimes "magickal" reasons. Can any of these methods be augmented through computer/human combinations? The purpose of this paper is to explore some of the possibilities opening up in the age of the personal computer, and to suggest specific experiments that might be undertaken by the individual seeker/scientist.

Research in so-called "artificial intelligence" is proceeding at a rapid pace at dozens of centers worldwide. Likewise, research on "brain/mind" interactions (a field that covers neurochemistry, neuroanatomy, neurophysiology, cybernetics, general systems theory, and many other specialized disciplines) is advancing faster than ever thanks to disccveries and methods developed in the last five years. Even that stepchild of science, parapsychology, is making headway against previous academic and scientific prejudices. Parapsychology was finally admitted as a legitimate "discipline" to the ranks of the American Association for the Advancement of Science (AAAS) in 1969. Many of the methods developed for research into these loosely connected sciences utilize computer power and data processing techniques. Due to the rapid reduction in cost-per-byte capacity and computer size, these fields are now open to investigation by many who could not have mustered the necessary resources in the past. Science is once again becoming accessible to gifted amateurs thanks to the microprocessor and related technologies. This tradition of "amateur scholarship" has long existed in Europe, of course, but has not been encouraged much in the U.S. due to what one writer has called "the cult of expertise."

The purpose of this presentation is to quickly sketch out some of the things currently being done with microcomputers in what we'll call "contelligent systems" research (for reasons which will be explained), to provide some beginning resource information for people who are, or may become, interested in these lines of investigation, and to create the beginnings of a data bank for the collection and dissemination of information on these topics, and, finally, to suggest some areas for further investigation/exploration.

The word "contelligence" was first used by Timothy Leary to describe the inseparability of the aspects of the human mind we call "consciousness," on the one hand, and "intelligence" on the other. Prior to the psychedelic revolution of the '60s, consciousness was relegated to the status of an historic curiosity. Ratomorphism held sway in academic psychology, and B.F. Skinner was king of the Conditioners. One of the first academicians to break the spell of ratomorphism was Robert Ornstein [1] who placed the study of consciousness squarely at the center of academic and research psychology. Prior to Ornstein, Arthur Koestler had done a pretty thorough job of debunking the behaviorists' reduction of all experience to sets of reflex arcs [2].

However, the pendulum of popularity quickly swung in the direction of bliss-ninnyhood as the pursuit of "higher consciousness" turned into a general skepticism concerning the methods of science or even the applicability of science to the study of awareness. John Lilly [3] along with a few others brought the discipline of science (applied intelligence) to the study of consciousness. In addition, Lilly emphasized that the one scientific subject accessible to everyone is one's self. Higher consciousness, then, remains a rather pyrrhic personal victory if it divorces itself from intelligence, which operates in the here-and-now of the human condition.

A contelligent system, at its very minimum definition, is the individual human being, equipped with the finest biocomputer evolution can buy (at least for mammalian bipeds). Even without the use of microcomputers, individuals can concern themselves with the *design* of their systems. There are many esoteric psychologies that teach methods for "altered states of consciousness" and methods for self-improvement. However, what we are interested in here is the machine enhancement of contelligence as represented by human/machine linkups and manifested in such methods as biofeedback and psychosimulation. For us a "contelligent system" is any combination of hardward/software/biocomputer that leads to a synergy of performance thereof. (Synergy: the combined effect of individual parts which is not predictable by their arithmetical sum.)

Proof of what can be accomplished *sans* machine is presented by Jack Schwarz, a Dutchman who taught himself voluntary biophysical control as he grew up in pre-World War II Holland. As a preface to a talk or lecture, he used to lie on a bed of nails and then allow the audience to inspect the healed puncture wounds at the end. At the Menninger Foundation or Langley Porter neuropsychiatric research centers, he would stick a knitting needle through his arm for the scientists. He would ask whether he should allow the hole to bleed or not while an array of sensors recorded his heart rate, brain waves, and whatever other functions the investigators could think up to monitor. [4]

There are five major approaches we have isolated for purposes of initial categorization of contelligent systems: (1) software design alone; (2) software interactive designs; (3) biofeedback linkups; (4) esoteric research design; and (5) combined methods for future research. This categorization is a broad one and should not be considered as final, since developments in any field always prove to be surprising to the paradigm promulgators of its beginnings.

Category one, software design alone, is accesible to any individual who currently possesses a microcomputer. Examples include the computerization of the *I Ching*, or Chinese book of changes, which acts as an intuition-honing instrument; computerized astrology computations; computerized tarot readings and the like. All of these programs are currently available with greater or lesser sophistication, depending on the skill and knowledge of the programmers, of course. A valuable service might be the collection and rating of the available programs in these areas with a view to providing a kind of consumer feedback for potential users. There are many such programs that are little more than trendy gimmicks and not very worthwhile to serious investigators of these esoteric tools of tradition. To paraphrase the poet Rumi: just because there are counterfeiters does not mean there is no such thing as real gold.

Category two, software interactive designs, includes the design of teaching games, simulations and self-help programs of various kinds. The design of lifework planning programs, for example, which would take the individual through a course of personal exploration of skills, interests, aptitudes and aspirations is the kind of design/research we're thinking of. This must combine, again, the skills of the programmer with the skills of the counselor, psychologist and educator.

One programmer/wizard of the author's acquaintance has taken the esoteric tradition of Tibetan Buddhism as represented in "The Game of Life" and has designed an interactive program which teaches, entertains, and incrementally enlightens all at the same time. [5] There is much to be gained, both personally and economically, in this particular category.

Category three, biofeedback linkups, begins to get us into the area of biofeedback instrumentation as well as good program design. One programmer, who became interested in dreamwork, designed a system for his microcomputer that would monitor his sleep states and wake him after a set period of REM (rapid eye movement) sleep. He would record his dream impressions and then go back to sleep. He reports many beneficial personal results and will, eventually, publish his experiences in order to guide others wishing to embark on their own dream exploration work.

Tim Scully, a gifted student of altered states of consciousness, has published several articles on the subject of biofeedback and microcomputers. [6] A technique called "visual evoked response averaging" has been used to study the brain's response to lights, sounds, and touch. In a series of experiments leading from this technique, a computer has been taught to identify words that an experimental subject was thinking. These early experiments are quite limited, of course, but point the way for further investigation by interested parties equipped with micros.

There are other signals besides brainwaves that can be used as input to a computer. Temperature feedback from the hands can be used as an aid to relaxation and visualization, both important elements in any kind of esoteric training. Voltages generated in

the large skeletal muscles can also be used for control purposes. [7] These other kinds of biosignals have implications not only for new kinds of interactive programming, but for new ways to aid the handicapped as well. Here, too, the field is wide open to amateur research.

In category four, esoteric research, we begin to impinge on the field of parapsychology. Researcher Nick Herbert of the C-Life Institute describes a contelligent system he constructed to investigate the effects of consciousness on the quantum-mechanical level. He coupled a radioactive source (thallium) to a typewriter through a computer which correlated the statistical properties of the decaying particles and the English alphabet. Many physicists are convinced that consciousness does affect events on quantum levels. Here, then, is a device, or at least the beginnings of a device, to help test that hypothesis. We quote directly from a paper describing this device called a "metaphase typewriter":

> The essential feature of the metaphase typewriter is that a quantum-mechanically indeterminate system (the unstable thallium nucleus) is coupled to an output meaningful in human terms (a printed text). One can imagine other outputs such as visual patterns, vocoder sounds, or musical instruments which could be interpreted by human observers. The meaningfulness of the output is necessary to arouse the observer's unconscious gestalt-constructing systems, inducing a particular kind of active expectancy as the output is produced. The quantum nature of the source guarantees that *anything can happen* without violating physical laws, since quantum mechanics places only *statistical* restraints on the output. Meaningful messages are not excluded by present laws of physics.
>
> Thus the metaphase typewriter becomes a desirable ESP machine. Infinitely responsive and without prior constraint, can it react to psychics, produce cryptic oracles when ceremoniously invoked, detect the presence of latent extrasensory information? Its quantum anagrams form a new instant and never reproduced kabbalistic gematria where the encrypted will of god may be read out from a computer console by an operator in the proper state of grace. As was the tradition in magical circles, the successful ritualist must create his own instruments, hand-copy his own grimoire (Xeroxed magic texts don't work), gather his own psychoactive plants ...

The researcher goes on to describe the possibility that discarnate entities might take over such a "voluntary typewriter," making it "the forerunner of a new race of quantum automatons, novel beings come from 'somewhere else' to share our living space." [8]

The two-person system formed when fingers are lightly placed on the planchette of a Ouja board might be duplicated and enhanced through computer linkup. Using as its variables the combined EEG output of two or more individuals, a metaphase device might be constructed that flashed letters onto a CRT screen in response to the input of the human beings involved.

Networked linkups, with the people involved interacting through telephone lines could accomplish the same thing. It becomes particularly exciting to imagine what gifted psychics might produce using such a system.

Those familiar with current sex research will remember the wave-shape of the typical sexual response cycle for men and women. It rises gradually at first, becomes steeper and then "peaks" or "spikes" at the point of orgasm. The shape is somewhat different for men and women, of course, but the point is that each position on this response curve represents, in essence, a particular biophysical state of the organism.

Using computer-enhanced feedback, might it not be possible to teach the principles of sexual tantric yoga much more quickly? The idea is to maintain oneself at any given point on the curve for prolonged periods of time, up to and including the experience of orgasm. Might this be the pathway toward the release of "kundalini" or the "serpent power" spoken of in esoteric tradition? Appropriate feedback monitors already have been invented, but there is also room for new research in this category.

The last category, combined methods, includes all the others in various combinations and permutations. For example, the design of a system which monitors biophysical indicators of relaxation while teaching a new skill or mental capacity combines the research being done in the pedagogical field of suggestology with biofeedback. The computer might be used to effectively determine appropriate states of concentration and to terminate the lesson when the individual tires and needs a rest.

Psychosimulations have been hinted at by some researchers already. From the imaginative involvement one experiences while playing a good version of *Star Trek,* to the complete simulation of experience possible with direct sensory/motor input, there is a wide range of possibilities that can be examined by the determined microcomputerist.

We have not covered all the subjects possible in each category in the interests of brevity. However, the bibliography lists some of the sources already available and should get the interested experimenter started. In addition, we have included listings of items which will be helpful to the amateur experimenter in the field of contelligent systems design. [9,10,11,12]

In order to encourage this kind of exploration on an amateur basis, we have formed the Contelligent Systems Interest Group, under the auspices of the Institute for the Study of the Human Future. Persons who have access to more information along these lines are urged to write and let us know. Scientific papers, articles, books, etc. are welcomed. In addition, we hope to be able to provide offprints of such resources in months to come. The address: Contelligent Systems Interest Group, ISHF, Inc., P.O. Box 14431, San Francisco, CA 94114.

ACKNOWLEDGEMENTS

Thanks from the author to Steven S. Smith for support forthcoming and past, Arlen and Robert Anton Wilson for egging him on, and to Nick Herbert, who gave him access to much valuable contelligence.

REFERENCES

1. Robert E. Ornstein, *The Psychology of Consciousness,* 2nd. ed., paper (New York: Harcourt Brace Jovanovich, Inc., 1977).

2. Arthur Koestler, *The Ghost in the Machine: The Urge to Self-destruction: A Psychological and Evolutionary Study of Modern Man's Predicament* (New York: Macmillan, 1967).

3. John C. Lilly, M.D., *Programming and Metaprogramming in the Human Biocomputer* (New York: Julian Press, 1972).

4. Jack Schwarz, *Voluntary Controls: Exercises for Creative Meditation and for Activating the Potential of the Chakras* (New York: E.P. Dutton, 1978).

5. Mark Tatz and Jody Kent, *Rebirth: The Tibetan Game of Liberation* (New York: Anchor Books, 1977).

6. Tim Scully, "Biofeedback and Microcomputers *et seq*," *People's Computers,* three parts, Vol. 6, Nos. 2, 3 and 5. Available from 1263 El Camino Real, Menlo Park, CA 94025, $2.50 ea.

7. Steve Ciarcia, "Mind Over Matter: Add Biofeedback Input to Your Computer," *Byte,* Vol. 4, No. 6 (June 1979). From Byte Publications, Inc., 70 Main St., Peterborough, NH 03458, $2.00.

8. Nick Herbert, "Metaphase Typewriter -- An Exercise in Free Assembly." C-Life Publication No. 3710, from C-Life Institute, Box 261, Boulder Creek, CA 95006.

9. Benjamin B. Wolman, ed., *Handbook of Parapsychology* (New York: Van Nostrand Reinhold Company, 1977). An indispensable volume for all would-be consciousness researchers.

10. Marilyn Ferguson, *The Brain Revolution* New York: Bantam Books, 1975). Herself a gifted amateur researcher, Ms. Fergusons' book presents a survey of brain/mind research. Anyone interested in keeping up with the field would do well to subscribe to her "Brain/Mind Bulletin," a newsletter published every two weeks. $15 per year from Interface Press, P.O. Box 42211, Los Angeles, CA 90042.

11. Elmer and Alyce Green, *Beyond Biofeedback* (New York: Dell, 1977). An interesting overview of biofeedback research by two of the pioneers in the field. The Greens are with the Menninger Foundation and have run, among others, Jack Schwartz through his paces.

12. Dean Gengle, "Neoshamanism in the 1980s," *Sphinx Magazine* (Basel, Switzerland, March 1980). Available from the author.

# ARTIFICIAL INTELLIGENCE AS APPLIED TO INPUT AND OUTPUT IN THE OFFICE OR..... MAKING COMPUTERS READ AND SPEAK

by
Art Derfall
Kurzweil Computer Products, Inc.
33 Cambridge Parkway
Cambridge, MA 02142
(617) 864-4700

Limited font optical character recognition and limited vocabulary speech response have been in office use for some time. By incorporation artificial intelligence (sophisticated programming that emulates simple human thought processes) two new technologies have been developed--omni-font OCR and unlimited vocabulary synthetic speech. Combining the two into one system you have The Kurzweil Reading Machine for the blind. Seperately the two have unique applications to system input and output. How a machine reads and speaks is as interesting as the "humanization" of that device.

A great deal of the technological developments in the office have been in hardware; the further sophistication lies in software enhancement especially in the area of artificial intelligence. Simply out, it's the science of making machines do things that would require intelligence if done by man. For example, reading any printed materials and speaking it out loud. It may sound far-fetched but such a machine was Kurzweil Computer Product's first device-a reading machine for the blind.

All the articles I've read concerning computer conferencing, word processing, COM, photocomposition, electronic mail, electronic libraries, etc. play up the fact that data storage has become much cheaper and transmission much faster. Huge data bases can be accessed by word-search strings, and the documents can be sent around the world to hundreds or millions of people. The one thing that is rarely mentioned is how the data bases are built. With all our existing knowledge in typed or printed format, data entry for storage or transmission has usually meant manual keying. OCR (optical character recognition) is faster than redundant keying but, up to now, has been limited to scanning only a few type fonts with many restrictions as to page format, line spacing, and most importantly, could only read typed material. Character recognition is accomplished by matching shapes which is a very rigid approach.

To enter text into a system from many source documents including journals and books, calls for an OCR device that can read any type face either uniformly or proportionally spaced. An omni-font scanner has to recognize characters on a page in much the same manner as you and I learned in school. The machine has to abstract the invariant properties of a letter, which in humans, represents a fairly high level of learning. An "A", for example, no matter what type style, has a unique form. It's a closed loop on top with an open cavity below, and an extension of the loop across the center. For final recognition the system has to seperate touching characters, mend broken characters, differentiate "els" from "ones" if they are the same character, and even recognize ligitures such as ff, fi, fl, ffi, and ffl that do not need splitting. The accompanying pictures show how the system is trained to recognize characters-how a machine learns to read.

Because of the variety of the materials that might be presented to the scanner some menu questions are answered first. In diagram #1, if zeros and 0's, ones and small l's look alike, the system will look at the surrounding characters in order to make a judgement. If the character is surrounded by numbers, it is probably a number also. You unconsciously would do the same. How would you differentiate capital I's form small l's if they were represented by the same character? It is probably a capital "I" if it stands alone or at the beginning of a word. The system uses the same logic.

Diagram #2 shows another menu for paragraph recognition determined by vertical spacing and horizontal indentation-informantion that might be important in text searching and photocomposition.

Diagram #3 shows Calibrating-the

scanner reads a line or two to get a feel for character weight, spacing, and differences in upper and lower case letters. When it has seen enough characters in Calibration mode, the system automatically enters Training. The inverse video characters are the system's way of saying, "Here's my best guess from my general, internal logic. I'm not 95% certain, though, so please tell me via the keyboard if I'm right or wrong." The large character on the right of the screen is a blow-up of the character on the page as the scanner sees it.

Diagram #4-Training. The steps involved are not much different from when we learned to read. The system makes initial identification. The operator makes corrections and deletes poorly formed images as being bad examples. As the system reads more lines with the operator (teacher?) intervening it becomes more and more confident in its guesses (fewer inverse videos). If the project has a mix of fonts, the system should be trained on each. Training can be stored and called back at a later date. In Data Entry mode the system will recognize characters in a mix of fonts and even put out a code as to when a certain font begins and ends. The operator in this mode can continue to make corrections and changes as with any edit station. The system with one operator can equal the output of ten operators manually keying text. The question becomes, "Is the machine assisting the human, or the human assisting the machine?"

The other area of technology Kurzweil Computer Products is involved in is unlimited vocabulary synthetic speech systems. Combined with the omni-font OCR technology, you have the Kurzweil Reading Machine for the blind. We have programmed our microprocessor to group letters into words. A 1000 rules of pronunciation and 2000 exceptions convert the text into synthesizes speech through hardware we developed. A special analysis then assigns a stress contour within each word. A sentence parsing program then analyzes the syntax of the sentence and its phrases to assign a stress contour within each sentence and appropriate pauses between words. The result is a relatively natural inflection. (Recordings of the voice are available upon request). The Quality of the speech is limited only by the number or algorithms that are added. Given time and a large enough market, synthesizes speech quality should approach that of the human voice.

Kurzweil Computer Products is developing a Talking Terminal for a variety of applications in the Office (and Home) of the Future. It will be a small plug-in device that can convert any digitized information to umlimited vocabulary synthetic speech. Connected to a word processing system or reservations desk, it would allow the blind to be hired for normally sighted positions by converting keystrokes and screen images to speech. A similar application is in the new home computer networks such as The Source from Telecomputing Corporation of America, and Viewdata just being introduced. U.P.I. news, stock market quotations, games, encyclopedias, catalogs, etc. can be called up right in your home or office. Speech output, even for the sighted, is desirable in order to "humanize" interaction.

Another application is accessing data bases from touchtone phones. The main telephone number plus a string of digits would connect you with the correct file. The computer would speak in response. Any unrecognized words could be spelled out by pressing the # key. The * key could signal "human assistance needed" or "please send me additional information." A subscriber identification number is all that would be needed.

Electronic messaging systems are usually designed around CRTs or printers. The equipment is expensive and messages can only be retrieved while in the office. One $1,000 Talking Terminal (and computer) accessed by 100 individuals from any phone would accomplish the same thing. Alternatively, each person's CRT or printer would cost that.

Weidner Communications in La Jolla, California and World Translation Company in Ottawa, Ontario market a software/hardware package that can convert text in one language to another on a work processor. The conversion is about 85% with the translation completed by the terminal operator. Certainly, an omni-font scanner is the best input device for books, journals, etc. but a more interesting system would have an unlimited vocabulary synthetic speech device for output. Scan a book in French or Spanish and have it spoken in English!

Artificial intelligence is a fascinating field with many applications for information processing. Since our store of knowledge is estimated to be doubling every five years (up from every ten years in the 60s) a computer terminal in every home, an omni-font scanner in every Post Office, or synthetic two-way interactive data wrist radios no longer seem impossible.

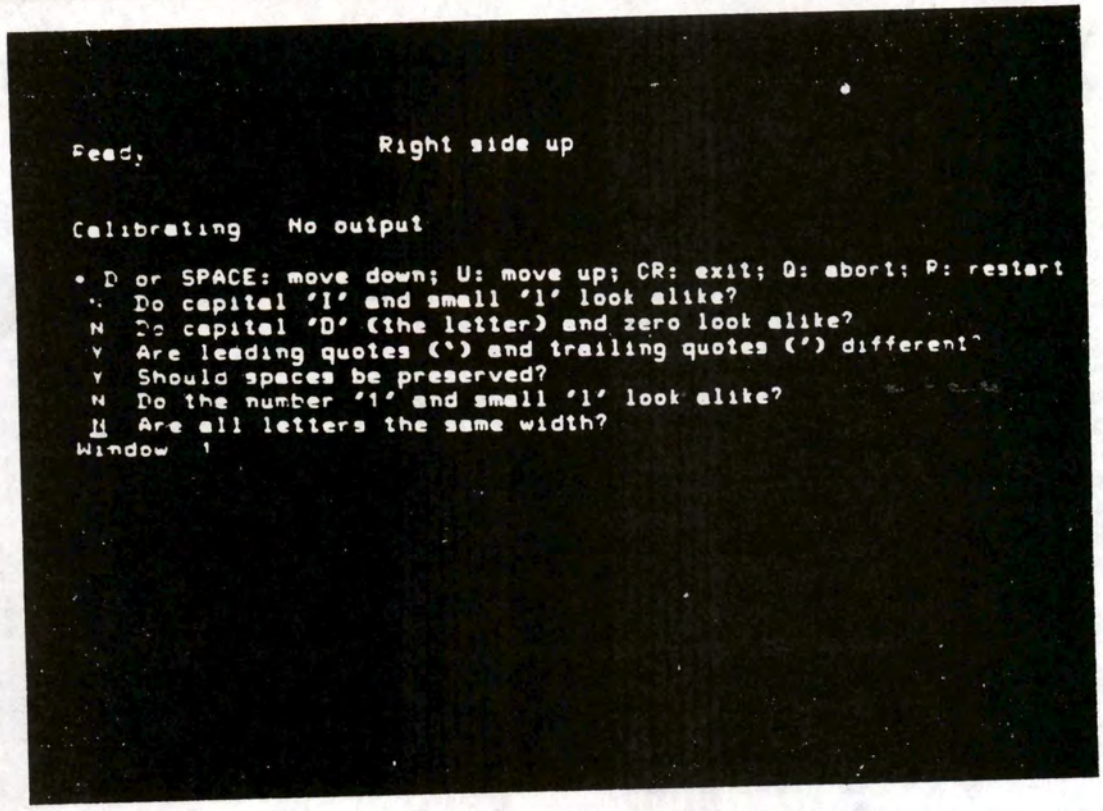



Diagram #1: Menu questions for aiding character recognition.

*Tuesday Morning.* It was this morning reported, that the *Bulldog* engaged the *Friseur,* yard-arm and yard-arm three glasses and a half,[3] but was obliged to sheer off for want of powder. It is hoped that enquiry will be made into this affair in a proper place.

*Tuesday Evening.* The account of the engagement between the *Bulldog* and *Friseur* was premature.

*Wednesday Morning.* Another express is arrived, which brings news, that the *Friseur* had lost all her masts, and three hundred of her men, in the late engagement; and that Capt. Grim is come into harbour much shattered.

*Wednesday Evening.* We hear that the brave Capt. Grim, having expended his powder, proposed to enter the *Friseur* sword in hand, but that his lieutenant, the nephew of a certain nobleman, remonstrated against it.

*Thursday Morning.* We wait impatiently for a full account of the late engagement between the *Bulldog* and *Friseur.*

*Thursday Evening.* It is said that the Order of the Bath will be sent to Capt. Grim.

*Friday Morning.* A certain Lord of the Admiralty has been heard to say of a certain captain, that if he had done his duty, a certain French ship might have been taken. It was not thus that merit was rewarded in the days of Cromwell.

*Friday Evening.* There is certain information at the Admiralty, that the *Friseur* is taken, after a resistance of about two hours.

*Saturday Morning.* A letter from one of the gunners of the *Bulldog,* mentions the taking of the *Friseur,* and attributes their success wholly to the bravery and resolution of Capt. Grim, who never owed any of his advancement to borough-jobbers, or any other corrupters of the people.

*Saturday Evening.* Capt. Grim arrived at the Admiralty, with an account that he engaged the *Friseur,* a ship of equal force with his own, off Cape Finisterre, and took her, after an

[3] The reference is to a nautical half-hour glass; hence "three glasses and a half" would be one and three-quarters hours.

Diagram 1A: The actual page to be scanned. Note italics and superscript in line 3.

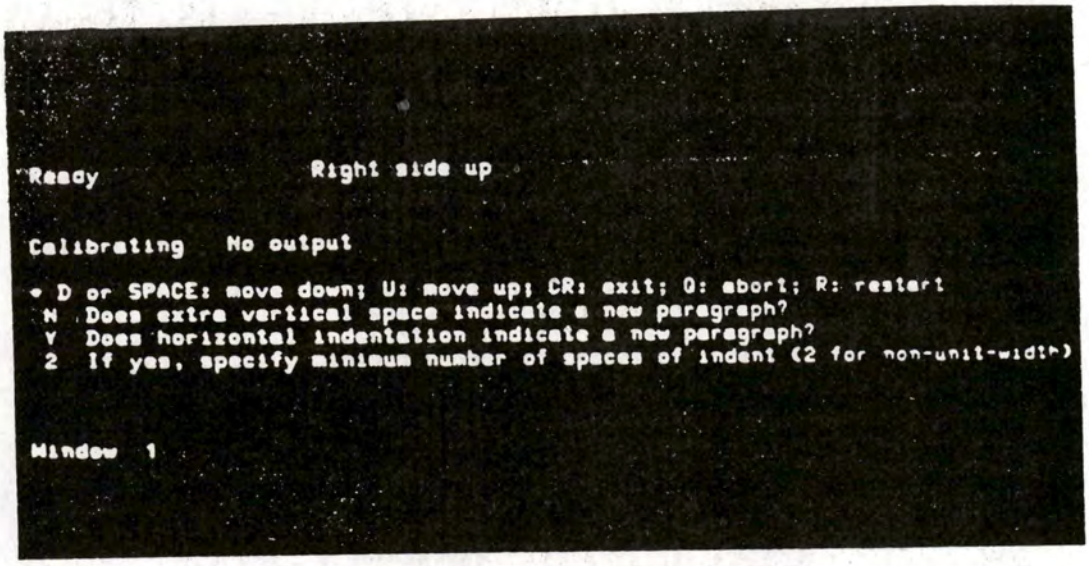



Diagram #2: Menu for aiding paragraph recognition.

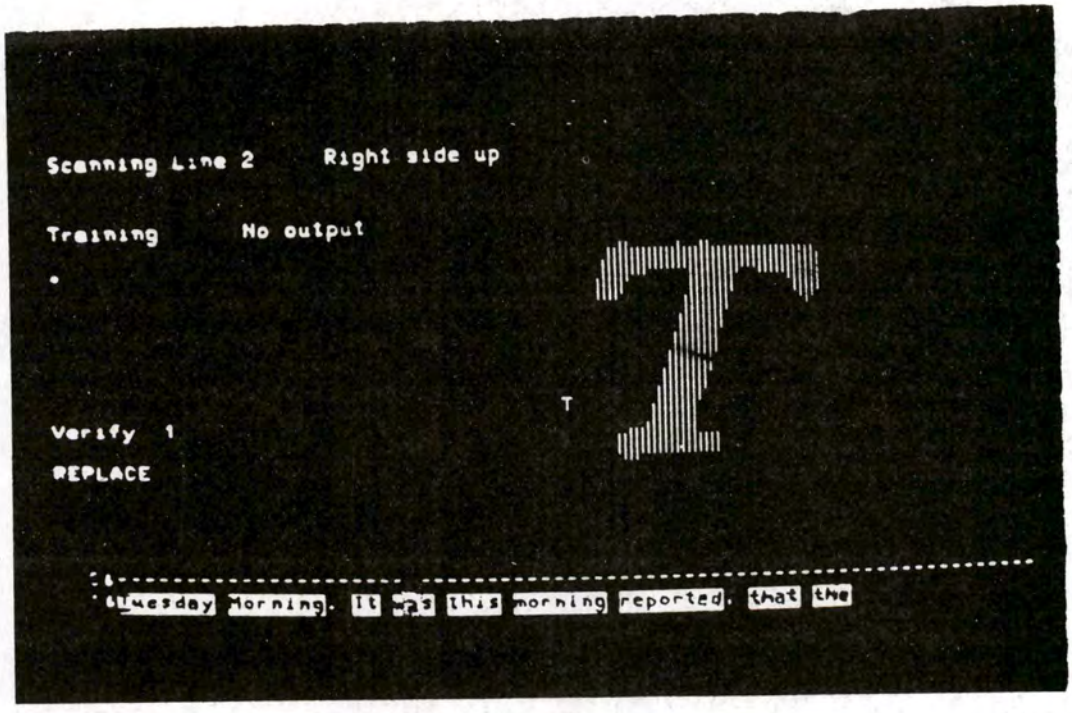



Diagram #3: Calibrating Mode. The scanner reads a line or two to get a feel for the size and spacing of characters. The system will automatically enter Training Mode. The large "T" is a blowup of the "T" in Tuesday as the scanner sees it.

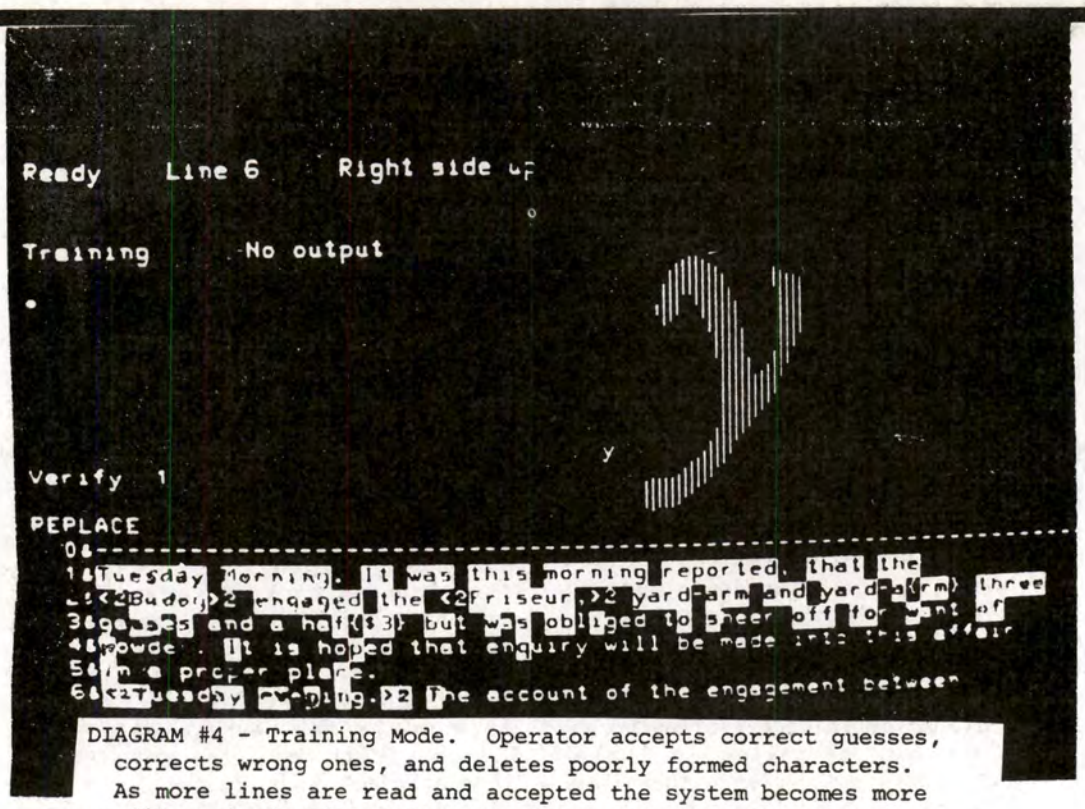



DIAGRAM #4 - Training Mode. Operator accepts correct guesses, corrects wrong ones, and deletes poorly formed characters. As more lines are read and accepted the system becomes more confident in its identification (fewer inverse videos). The <2 >2 notation indicates a change in font. In line 2, brackets aroung "rm" were placed by the operator to indicate the correct identification of two touching characters. In line 3, {$3} denotes a superscript.

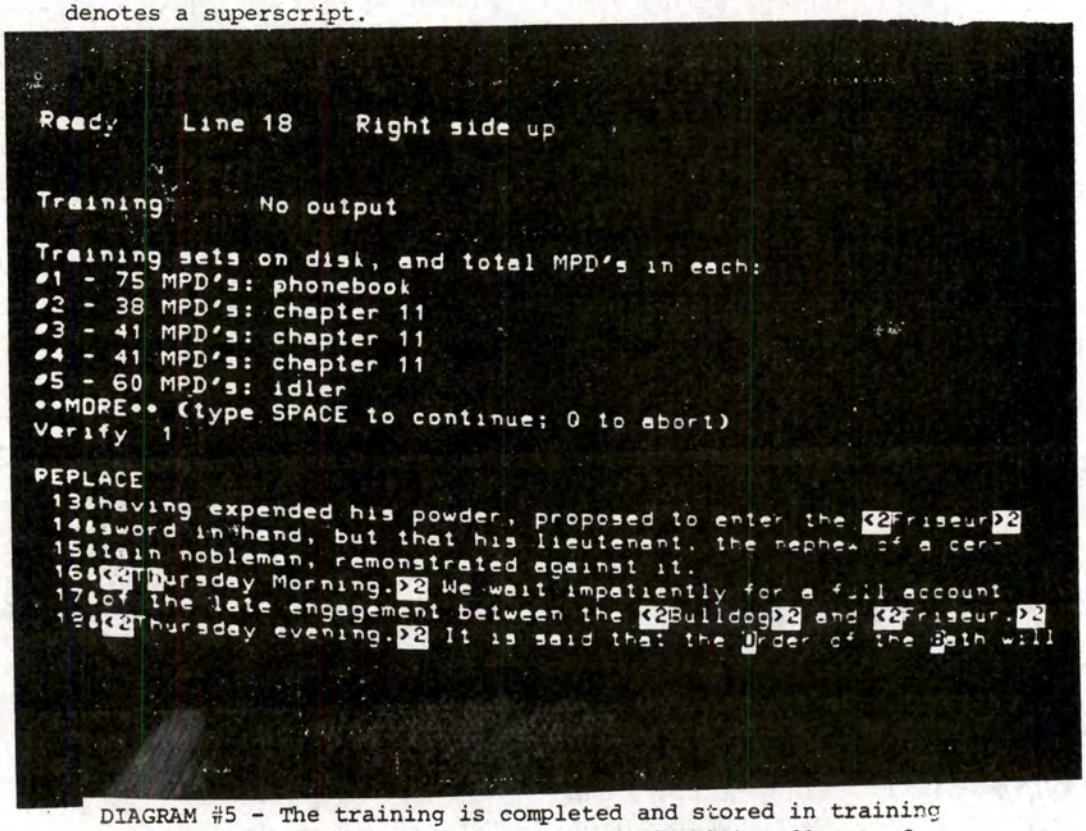



DIAGRAM #5 - The training is completed and stored in training location #5 (25 training locations available). 60 out of 205 MPD cells (multiproperty descriptor) have been used. The cells are software images of trained characters.

# THE STARSHIP SIMULATION PROJECT

David and Annie Fox
Co-Directors of MARIN COMPUTER CENTER
a non-profit educational organization
70 Skyview Terrace Room 301
San Rafael, CA 94903
(415) 472-2650

## Abstract

Anyone who has ever seen a clear sky at night has desired to leaved this planet (if only temporarily) and travel out into a sea of stars. People have been wondering about such an experience for thousands of years. Unfortunately, actual space technology has been hard pressed to keep pace with human dreams, and our imaginations have been further stimulated by science fiction books and films. And yet, partaking in these media creates a space experience that is a rather passive one.

What would it really be like to be a part of the crew of a galactic cruiser? With the use of computers, within the context of a complete sensory environment, we are creating an opportunity for people to find out.

Through the auspicies of a grant from the San Francisco Foundation, Marin Computer Center has, for the past 12 months, been putting together a full scale, operational bridge of an interstellar vehicle. It is called The Starship Simulation Project.

## Why?

In 1967 the television series Star Trek was first broadcast. Millions of people watched in awe as an incredibly talented group of writers, set designers, special effects people and actors pooled their energy to create an illusion of space travel for their audience. The dream was so real and compelling that for one hour each and every week we left our living room couches and sat on the bridge of the Starship Enterprise.

When the miscalculated decision to cancel Star Trek came down from the NBC network executives, there was an outpouring of sentiment from the public that was overwhelming and unprecedented. We were outraged and saddened that our reality/fantasy of being on board the Enterprise was snatched away from us only to have our beloved Starfleet Officers replaced by the inhabitants of some trite situation comedy. Our pleas to bring back the show went unheeded, and we were disconsolate to be grounded so heartlessly.

But true science fiction fans are never earthbound for long. We lived in the hope that Captain Kirk and crew would one day return to us. Then came the re-runs, and we watched them like junkies savoring a fix. And here it is 13 years later. We've experienced the **Star Wars** craze, the Star Trek movie has been released as well as Disney's **Black Hole**, and all of us have seen them at least once (so far).

How can we explain this phenomenal craving for an intergalactic experience? We can't really - except to say that from the beginning of humankind's origin on this planet, we have had a fascination with the stars and an unquenchable obsession with the possibility that there are other inhabited worlds to make contact with.

Towards this end, Marin Computer Center has begun building a Starship. Unfortunately, the ship we are building will never leave the ground, but its crew will never know the difference.

## What?

In March 1979 we began work on an endeavor which represents a revolutionary step in educational technology, the Starship Simulation Project, a computerized flight control deck of a simulated interstellar vessel.

We are constructing a multi-sensory environment that will enable people to have a very real experience of space travel. It's our goal to provide people with the experience that we live in a universe of infinite possibilities, that the Universe, as well as our role within it, is limitless! We are setting out to expand people's ability to dream, and in so doing, empower them with the knowledge that those dreams can become a reality. To dream is to move from the known into the unknown. After a "Starship Experience", the unknown will become more approachable. By giving people new dream material, new goals are set and

eventually attained. Man's ultimate future is not an earthbound one. We are a species of explorers. The Starship Simulation Project will inspire renewed enthusiasm for actual space exploration.

How will this be achieved? By making the special effects so complete that after a one-hour "flight" participants will be visibly shaken, that is, they will experience the challenge, thrill, joy and fear that a "real" Starship crew might experience. Realism is the key to the Starship's success. This realism will be generated by equipping the Starship with the following:

It will be contained within an oval shaped geodesic dome to isolate the players from the present day environment.

The dome will be contained within an 800 square foot "hanger deck" (a room at the Marin Computer Center).

The game will use slides, video tape, film, and computer graphics projected on a large screen to create images of space travel, alien encounter and communication (like a picture phone) and atmospheric evaluation of planets (scans). Also available will be visual access to a library of essential scientific information.

A quadraphonic sound system will be installed to give the most realistic sound effects possible.

Colored incandescent lighting will be used to create various moods on the bridge.

A galactic map will be projected on the dome ceiling.

The deck will be outfitted with six command consoles, each equipped with an Apple II, color monitor, light pen, joysticks, and various buttons, lights, and switches. All consoles will talk to each other through a Marinchip Microcomputer. Each console will be designed for a specific task. These stations are:

1) Captain/Trade Officer - overall responsibility for the success of the mission. Will make "ultimate decisions" on destinations, priorities, trade agreements.
2) Navigator/Helm - responsible for plotting course to desired destination, setting up hyperspace jumps as well as maneuvering the ship at sub-light speeds.
3) Communications Officer - responsible for all communications, audio, visual, navigational, and digital, within the ship and between the ship and other ships or planets.
4) Science and Technical Information Officer - responsible for analyzing and identifying objects around the ship. He has control of the ship's long and short range scanners and the on-board science computer library.
5) Engineer - responsible for maintaining the ship at optimal levels of operation through power distribution and control of life support systems.
6) Defense/Offense Officer - responsible for the ship's defensive weaponry/technology (tractor beam, pressor beam, shield, laser).

The underlying philosophy of this game is that **all life forms are intrinsically worthy of respect.** It is unethical to destroy either these life forms or their creations. The ship has a wide range of technological devices, but no "weapons". The technological devices could be used as weapons, though, if someone really wanted to (a potato peeler can be a lethal weapon in the wrong hands). When the Starship meets other life forms which appear to be hostile, it would be very easy to use the ship's technology to destroy them. However, the consequences of such an action would be dire (Karmic kickback). The players will rapidly begin to look for other ways to deal with hostile aliens. There will always be at least one or two workable alternatives to the use of violence. For example, if the players viewed the "enemy" as someone whose goals conflict with those of the Starship, then the conflict might be resolved by discovering a way to expand the Starship's goals to include those of the "enemy".

The crew will be presented with a wide variety of scenarios. These scenarios will include realistic obstacles to overcome while on their "mission". For example, a planetary medical emergency, alien contact, diplomatic negotiations with alien societies, information gathering/exploring, protecting a planet from a natural disaster, commerce and trade, time trip into the past, etc...

Phase I of this project includes the designing and actual building of the Starship. Groups of students, hobbiests and professionals (from the fields of video game development, video production, computer

programming, mathematics, electronics, art, architecture, and education) all have been volunteering their time and expertise to the project. These people have split up into four main groups - Design, Storyline, Hardware, and Software.

Phase II of the project is opening the "hatch" for the general public. The game will be designed so that people at all skill levels can participate. For example, the Simulation will give people playing at the beginning levels more time to respond to emergency situations than experienced players.

All players will need to combine their different talents and skills in a group problem-solving situation to achieve the goals set in the scenario. Players must apply their knowledge of mathematics, physics, logic, human relations, psychology, science, history, and more, in order to accomplish the mission. The accuracy and skill with which they apply this knowledge will determine their success in the game.

Storyline possibilities are endless, as are the possibilities for hardware embellishments. To keep the Simulation forever challenging, random events will be merged within a pre-defined galaxy. As new technologies become available, they will be incorporated into the Simulation.

The Simulation need not be limited to space travel. The bridge of a space ship looks very much like the bridge of a submarine. All that would be necessary to totally change the game to one of underwater exploration would be to re-write the software. Or how about a journey through the human body, a la Fantastic Voyage.

The Starship is a tremendous learning tool and will make it possible for anyone to directly experience the theories and abstractions of advanced science, which otherwise could only be comprehended intellectually, if at all. The participants are being given an opportunity to exercise and expand their creative potential and actually SEE their ideas become reality.

What In?

Mass Production - The storylines for the simulation call for our starship to be one of a mass-produced product, serial number 500.320 of the Terran StarTrain Works of Juno. If we look back at projections of the future of automobiles, computers, and many other technologies, we find that the forecasters neglected to consider how cheap such things would be after mass production got underway. Assuming that the drive isn't outrageous, there is nothing in a starship that is fundamentally very expensive once you're set up to make a couple of million of them (compare the price of a handbuilt car with a Ford, and remember that the handbuilt is using a commercial engine!). One of these starships will cost about as much as an oceangoing yacht costs today.

The fact that this is a mass produced starship is very important in maximizing the number of potential storylines. If this were the first starship, we preclude interaction with other human-controlled ships. If this were one of a small number of ships (as in Star Trek), we must therefore assume it is run by a huge bureaucracy back on Terra (e.g., Starfleet Command), and therefore has a more limited range of missions.

Imagine instead the area full of ships flitting from place to place. These ships are owned and operated by all kinds of people/groups, such as prospectors, scientists, corporations, governments, pirates, utopian colonists, adventurers, and traders. In other words, the kind of things you find on the high seas today. Our ship could, then, represent any one of these groups in interaction with any of the others. The possibilities are endless.

Modular Construction - our mass produced ship is a collection of modules. These modules are interchangable and configurable in a large number of combinations, more like railroad cars than current rockets. A list of possible modules follows:

* Command module (our simulator).
* Habitat module (living space: human and alien models are available).
* Star drive module.
* Planetary landing module.
* Cargo module (many kinds available).
* Fuel module (you use more for long missions).
* Farming module.

More modules can be invented as we need them. For example, a mission to observe the core of the Galaxy (per Larry Niven) might use a Laboratory module outfitted with a large telescope.

The use of a modular design supports the multiple mission scenario for the starship. We can put together the kind of ship needed for the mission we're flying. It seems to be the logical way you would go about mass producing starships. Mass production encourages us not to overspecify the mission in our design of the command

module (simulator). We want "soft controls", both because they're more realistic given the drift of technology, more suited to a flexible set of missions, and also easier to do given the technology we have to work with in building the simulator.

Who?

Below are the cursory descriptions of the functions of the entire crew. Many simplifications are possible for younger players, and much more variety is possible for very advanced players.

Captain/Trade Officer - His job is to keep track of "What's going on", and make the "Ultimate" decisions for trade agreements, priority, destinations, and leading the crew through the galaxy. The Captain can get business advice on deals from the computer which will educate him in basic business principles. In order to be successful in this role, a player must possess or develop communication skills, foresight, diplomacy, restraint, decisiveness and good judgement.

Helm/Navigator - Has two responsibilities:

1) Plot a course to a desired destination.

2) Implement the course and maintain it.

The ship has two types of propulsion - Hyperspace jumps and a "reaction" type drive in "normal" space.

Hyperspace travel is by means of "jumps" through Hyperspace. A minimum distance for a jump is stipulated and a maximum distance is dictated by several factors which worsen as the size of the jump increases - power requirements and the accuracy with which you can predict where you will re-enter "normal" space. Time required for the jump will probably be minimal (approximately 20 seconds, player time). The ship's position must be verified after every jump, either through navigational beacons or by identifying a star's spectral pattern. These functions are accomplished through interaction with communications and/or science stations.

"Normal" space movement is accomplished by means of a "reaction" type drive, which will be based on Einstein's equations. Time dialation may, or may not be modeled, but the limiting speed of light will be amply exemplified. The power available is not limited by fuel, but by other engineering concerns. See engineering for details.

All maneuvering is planned to be accomplished in 3-space.

Science Technical Information Officer - This officer has the responsibility of analysing and identifying objects around the ship. He has control over the ship's scanners. By changing the sensitivity of the scanners and their beam width, he gets a trade-off between how much he can see and how big an object is before he can see it. Also, he controls the area of the scan, which if abused, could blind the ship to some very real dangers. Until he can identify and tag an object, it remains a "boogie" for all other screens (ie, Navigators display).

His information arrives as a Topographical map showing energy levels for a given type of energy (ie, radio, visible light, etc). Upon request, a summary of all bands of energy can be displayed for a given point. From this information, and with the help of a "library" computer, the object may be identified and tagged. Once tagged, it is identified on all other scanners on the ship, and is tracked automatically.

By using the same technique, the spectra of a sun may be used to identify the sun and use it as a reference point for navigaton. Note that three points are required for a fix, or a combination of suns and navigational becons.

Engineer - This crew member has control over an on-board power source which is nearly limitless. Alas, if you generate power you must use it - somewhere. If not in doing work, then by changing it to heat and radiating it. Also, since no device is 100% effective, heat (waste) is generated even when converting energy to work. The surface of the ship is a collection of radiators. By controlling which radiators are in use and how much energy is generated, the ship's temperature may be kept within acceptable limits. Exceed these limits, and critical devices may fail and eventually the ship itself! Here is a very practical lesson on Ecological Balance.

Communications Officer - This person is a kind of super switchboard operator. There are a limited number of receiver/transmitter pairs (channels) and they must suffice for all the needs of the ship. Four modes of communication will occur:

1) Digital communications - A version of teletype signal. This is the most preferred mode for most types of data, since it can be easily stored

by computer, encoded/decoded, and transmitted as a burst. It requires a minimum of ship's resources. The operator must decide where the messages are to be routed (captain, personal, ignored) and must bring priority traffic to the captain's attention.

2) Voice communication - requires two channels, for simultaneous listen/talk capability. Reserved for high importance traffic such as negotiations and diplomatic overtures, and sometimes "docking" maneuvers.

3) Navigational beacon receivers - Although inertial navigation techniques work well for normal space travel, a jump through hyperspace will disorient such a system. By triangulating off of three beacons, the inertial navigation system may be reset.

4) Video - videotaped messages may be created on board to be sent to alien cultures or to the home base.

Defense/Offense Officer - Does more than merely point the gun and shoot. He has three devices at his disposal:

1) Tractor Beam
2) Pressor Beam
3) Shields

The tractor and pressor beams are used for:

1) Destroying large meteors. In this mode the tractor and pressor beams are alternated, causing great stresses on the molecular level. Abuse of this could destroy the ship.

2) Loading the ship in space. In this mode the beams are used simultaneously balancing one against the other.

The shields can be used to block the tractor/pressor beams in a hostile situation, by setting up an interference pattern. They can also serve, with varying degrees of effectiveness, as protecton against other energy beams or projectiles. The use of the shields reduces scan and communications to very short range.

## Where?

The galaxy is to be as realistic in terms of its physical make-up as possible. 100 X 100 X 8 light years, spiral arms, central core with radiation storms around it. Radiation storms have only a few entrances for ships, but are uninhabitable except for small, long-lived space stations. A Black hole is in the center of the galaxy. The galaxy has inhabited systems, desolate waste areas, high-radiation quadrants, gas and dust clouds, etc...

The galaxy will be structured somewhat like our own in terms of star distribution. All types of systems and stars will be included.

All possible combinations of governments, civilizations, and cultures can take place in this galaxy, from stone-age planets up to multiple system cultures. Technical levels range from neanderthal to energy beings that can do it all by pure thought. All conceivable cultural interactions have taken place, are taking place, or will take place.

Systems change - worlds can evolve from one type to another. A colony may gradually become an industrial world, for example, or the economy may collapse, civil wars can break out, etc., etc.

The dangers to be encountered on an interplanetary voyage can be of natural origin, such as asteroids, radiation, novas, gravity hazards, time/space warps, plagues, and hostile life forms. However, disaster need not be limited to those already occuring in nature. Killer droids, wars, computer malfunctions, fuel shortages, and galaxy-wide recessions are all possible challenges to the ship's crew. Imagination is the limit in this area.

## Conclusion

Although 30 of us have been at work on the Starship for the past year, the scope and magnitude of the project allows room for many more participants. If you are interested in helping us out in any way, contact us in care of the Marin Computer Center. In any case, we invite you to become a member of the crew when the Starship is ready to depart!

## Acknowledgements

We want to thank Doug Fajardo, Michael Klassen, John Walker and the rest of the Starship Simulation Project participants for their contributions to this paper.

We also want to thank Chris Beatrice for his ability to capture the essence of the project in his art work.

**Please see frontispiece illustration.**

## COMPUTER GAMES IN EDUCATION

David H. Ahl
Creative Computing
P.O. Box 789-M
Morristown, New Jersey 07960

The term "computer games" refers to a wide variety of recreational and educational uses of computers which have already proved their value in a variety of settings but which are only in their infancy. For recreation, they offer far more potential variety and sophistication than TV or handheld games. And for education, they provide an incomparable motivational tool.

As an example in the educational context, suppose it's 1847. Your team of oxen has been pulling your wagon across Kansas for the last two weeks and you reckon you've been covering a good fifteen miles a day. Faced with the Republician River you decide to ford it and -- oh, no! -- it's deeper than you thought. The wagon is swamped and you lose most of your food and clothes. Fortunately you had secured your medical supplies and ammunition high in the wagon and you didn't lose those too.

But now you have to decide whether to make for the next fort which is three days away or to stop and hunt. Since you're in buffalo country, you decide to hunt and wait until later to stop at a fort.

And so it continues in the computer game, "Oregon Trail." During your six-month journey you face attacks from wild animals, Indians and bandits. Your wagon can get swamped, break a wheel or even have a fire. Your oxen can get injured or wander off. In the mountains, heavy rains, snow, and impassable trails are constant hazards. Illness and injuries are always a threat.

Fewer than 30 percent of the pioneers that set off from Independence, Missouri from 1840 to 1870 ever made it to the west coast. Now it's your turn to see if you can be one of the survivors. But at least if you don't make it, you get another chance.

How would you like to learn about history by living through it vicariously via a computer game? Given the choice, millions of kids are opting for the computer game approach and their teachers are finding the learning is frequently better than that provided by the traditional textbook.

These types of games, sometimes called simulations, are available in many subject areas for many grade levels. In social studies, young children can learn how a simple economic system works using the program "Hammurabi." Players decide how much land to plant with grain each year, how much to feed their people and how much to trade with neighboring city-states. Harvests are good some years, bad in others. Rats sometimes get in the grain bins. "King," a more complex simulation in the same vein, introduces the problems of industrial development, pollution and tourism. Economics on a local scale can be experienced with the program "Lemonade Stand." In the simulation, players must decide how much lemonade to make, the price to charge and how many signs to make. One outside variable is the weather; rainy days are obviously not good for lemonade sales. On the other hand, a circus parade is a big stimulant.

Science and ecology simulations allow players to experience situations and experiment with variables that are far beyond the reach of any normal classroom or textbook. For example, in "Malaria," students must try to control a malaria outbreak in a Central American country. Pesticides, treatment for the ill, preventive innoculations, and field hospitals are the variables to be controlled. Students discover that the disease is easy to control if cost is no


Sample runs from four educational games available from Creative Computing Software (l to r): U.S. Map, Lunar Lander (LEM), Malaria, and Hangman.

object, but unfortunately it's a major barrier in most third world nations.

"Tag" lets students experiment with the tagging-and-recovery method of measuring wildlife populations. Users of "Sterl" try to eradicate the destructive screw worm fly with various types of pesticides as well as male sterilization.

In English, games such as "Hangman" and "Don't Fall" help teach children word skills in a highly motivational way. In "Madlibs" and "Red Riding" the program creates funny, often hilarious, stories. However, for them to be readable, the player must use various parts of speech correctly. "Spelling," "Haiku," "Bard," and other games provide practice in other aspects of language arts.

"Adventure" is a wildly popular computer game in which players explore a Tolkien-like environment complete with trolls, animals, treasures in a landscape of hills, ravines, forests and caves. You give the computer commands such as "Go West," "Pick Up Lantern," or "Drink Water." Your object is to 1) discover how to play the game and 2) overcome the obstacles, find the treasure, and return to "civilization." Now imagine playing this fascinating and addictive game in French! Or Spanish. Or German. Language teachers have never before had such a powerful motivational tool to assist them in their teaching. It's like parachuting the student into the midst of a foreign country with the instructions, "learn to speak the language and you will be able to survive, find some treasure, and escape."

Similarly fascinating games have been written in many other subject areas: mathematics, geometry, and logic (where it all started), physics, chemistry, biology, economics, business, medicine, geology and many, many others.

Computer games know no age limits. For the new Sesame Place participatory play parks, Creative Computing Software has developed a series of games for players who cannot yet read. At the Lighthouse School in Wisconsin, first to third graders are writing their own games. Elementary and secondary schools, colleges, and graduate schools throughout the U.S. are using computer games. In Salt Lake City, a public education center is teaching adults English language skills via computer, while several museums have computer game exhibits oriented to visitors of all ages.

Of course, many of these "educational" games are being played for just sheer fun. The dividing line between education and recreation is a bit muddy. In playing chess, for example, is it solely recreation, or is there learning taking place? There are, however, scores of computer games written soley for fun. Computer hackers at many installations wrote games for their own enjoyment in the 50's. The first of these to emerge on a widespread basis was spacewar which surfaced at MIT's EE Department around 1961. Also, some of the chess and checkers programs written as part of artificial intelligence reseach projects in the 50's and 60's started to become more widely available too.

But the widespread availability of computer power in the late 60's and mass availability with the advent of the microcomputer in 1975, saw hundreds of small companies and individuals rush to produce the "ultimate" computer game. "Star Trek" held the lead in popularity for about two years, but Paramount's unwillingness to license the name has prevented it from real mass use. As this is being written (late 1979), contenders for the leading computer game include "Breakout" (Atari) and its many derivatives, "Super Invader" (Creative Computing), "Adventure," and several chess and backgammon programs.

Where to from here? It's anybody's guess. Obviously, new games will use the newer technology now available -- high resolution color graphics, multi-channel music and sound synthesizers, voice systhesis and recognition, light pens and much more. The real-time "Air Traffic Controller" game/simulation today almost exactly duplicates the situation faced by an actual air traffic controller. Tomorrow, one will be able to eliminate the word "almost."

Computers weren't invented to play games. But computerists, right from the very beginning, found that these machines could be programmed to do so -- awfully well. The known motivation of games in general and the fact that computer usage tends to increase personal interaction and peer tutoring make computer games one of the most powerful educational tools and one of the most enjoyable recreational pastimes available today.

# # # # #

# SOLVING THE SHOOTING STARS PUZZLE

Joel Shprentz
The BDM Corporation
7915 Jones Branch Drive
McLean, Virginia 22102

## Abstract

Shooting stars is a puzzle that can be played on small computers. The shooting stars puzzle is solved by transforming an initial pattern of stars into a final pattern of stars in the minimum number of moves. The puzzle is similar in structure to a network with the nodes representing patterns and arcs representing moves. The solution to the shortest path problem in the network is also the solution to the puzzle. A FORTH program that solves the puzzle on a TRS-80 computer is presented.

## The Shooting Stars Puzzle

The shooting stars puzzle can be played on small computers [2] and programmable calculators [3]. The puzzle is played in a universe which is arranged in a 3 by 3 grid (figure 1). Each of the nine universe grid positions holds either a star or a black hole. The goal of the puzzle is to transform the universe from an initial configuration (figure 2) to a final configuration (figure 3) in the minimum number of moves.

Moves are made by shooting stars. Black holes cannot be shot. When a star is shot it becomes a black hole. Neighboring stars and black holes are also affected by shooting stars: stars become black holes and black holes become stars.

The neighborhood around a star is known as a galaxy. The galaxies of a corner star, a side star, and the center star are shown in figures 4 to 6. The corner and side galaxies can be rotated around the center of the universe to form the other corner and side galaxies.

Shooting stars is played on a small computer. The computer is used to display the universe, accept the player's move, and update the universe. This process is repeated until the player either wins when the final configuration (figure 3) is displayed or loses when no stars remain to be shot.

## The Shooting Stars Network Model

The shooting stars puzzle can be modeled by a network of nodes connected by arcs. Each node represents a pattern of stars and black holes. Each arc represents a move that transforms one pattern into another. The network contains 512 nodes and 2304 arcs.

The portion of the network representing the first two moves of the puzzle is shown in figure 7. Each node is labeled with the pattern it represents. Each arc is labeled with the number of the star that was shot in the move that the arc represents.

The player can win by finding a sequence of arcs that creates a path from the initial node (representing the initial configuration) to the final node (representing the final configuration). The goal of winning in the fewest moves is achieved by finding the shortest path from the initial node to the final node.

## Exhaustive Search

The exhaustive search approach is to let the computer find all possible paths from the initial node. The computer could identify paths which reach the final node and record the shortest such path.

This brute force approach cannot be used because of the loops in the network. The loops would trick the computer into finding an infinite number of paths through the network. The computer would literally be going around in circles. One of the smallest loops is shown in figure 8.

## Finding the Shortest Path

The algorithm used to find the shortest path in the shooting stars network is a modification of Dijkstra's algorithm for the shortest path problem [1]. The algorithm has been simplified and tailored to fit the special structure of the shooting stars network.

Information is kept in two places during the solution process. Nodes awaiting processing are maintained in a queue. The best arc to follow from each node is kept in an array. The best arc is the one that lies on the shortest path to the final node. Initially, the final node is on the queue and all best arcs are set to 255 (an invalid arc number that flags an unprocessed node).

The shortest path is found by starting at the final node and tracing backward through the network, recording the best arc to take from each node. After initialization, a node is removed from the queue and all nine possible arcs (one for each grid position) ending at the node are traced backward as described below. The process of removing a node from the queue and tracing the arcs backward is repeated until the queue is empty.

The key to the algorithm is tracing the arcs backward from each node. Here are the four steps of tracing an arc:

1. Check whether the backward arc exists. Backward arcs exist only for grid positions containing black holes.
2. If the arc exists, determine the node from which the arc originates. This is the process of moving backward through the network.
3. Check whether the originating node has already been processed. The best arc from the originating node will be 255 (an invalid arc number) if the node has not been processed. This step prevents searching in loops.
4. If the originating node has not yet been processed, process it now. Record the arc under consideration as the best arc to take from the originating node. Put the originating node on the queue for processing.

## The Computer Program

The program (listing 1) is based on the algorithm described above. The program is written in FORTH [4] and is used on a 16k level II TRS-80 computer.

The word SOLVE (line 9, screen 3) is the main loop of the program. It processes nodes until the queue is empty. Each node processed is removed from the queue and has its nine moves traced.

The word TRY-MOVE (line 3, screen 3) traces a single move (arc) backward from a node. It executes MOVE-OK? to check whether the move is valid, NEXT-UNIV to determine the originating node, NEW-UNIV? to check for prior processing, and NOTE-MOVE to process the originating node.

## Results

The shooting stars puzzle is solved by the program in 21 seconds. The output of the run is shown in listing 2. Figure 9 shows the eleven moves of the solution.

## References


[1] Moder, Joseph J. and Salah E. Elmaghraby. Handbook of Operations Research: Foundations and Fundamentals. Van Nostrand Reinhold Company, New York, 1978.

[2] Nico, Willard I. Shooting Stars. Byte, #9 (May 1976) 42-49.

[3] Pearce, Craig A. Desktop Wonders. Shooting Stars (for the SR-52 and PE-100 printer). Byte, #16 (December 1976) 92-93.

[4] The Software Farm. tinyFORTH User's Manual. Reston, VA, 1979.



Figure 1. The nine grid positions of the universe.


Figure 2. The initial configuration is one star surrounded by eight black holes.


Figure 3. The final configuration is one black hole surrounded by eight stars.

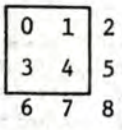



Figure 4. The corner galaxy around star 0. Other corner stars (2, 6, 8) have similar galaxies.

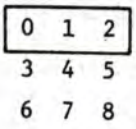



Figure 5. The side galaxy around star 1. Other side stars (3, 5, 7) have similar galaxies.


Figure 6. The center galaxy around star 4.

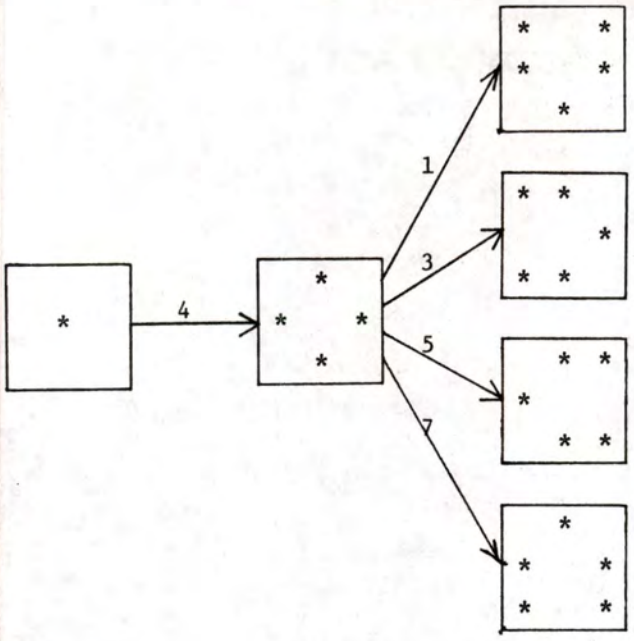



Figure 7. The portion of the network for the first two moves. Each node is labeled with a pattern of stars and each arc is labeled with a move (a star to shoot).

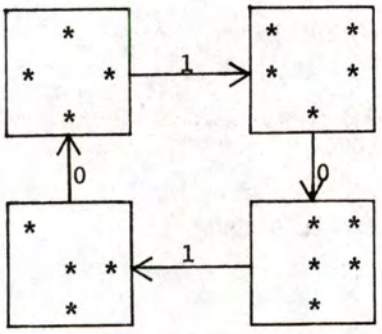



Figure 8. One of the loops in the network.

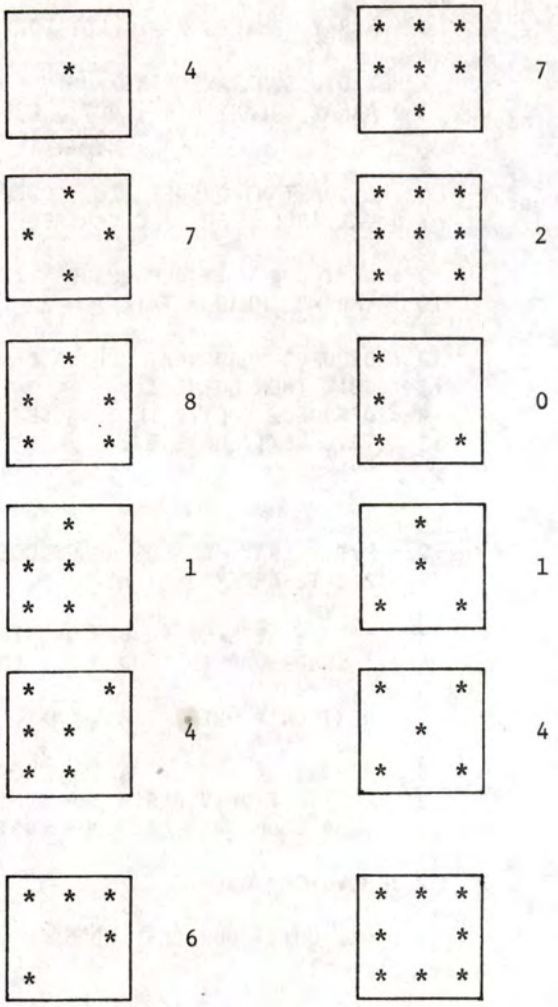



Figure 9. The solution to the shooting stars puzzle (read downward). The number of the star to shoot is shown for each pattern.

```
SCR # 1
  0 : ARRAY <BUILDS 2 * ALLOT DOES> OVER + + ;
  1
  2 HEX 010 CONSTANT FIRST-UNIV    1EF CONSTANT LAST-UNIV
  3 0 ARRAY GALAXY 01B , 007 , 036 , 049 , 0BA ,
  4                124 , 0D8 , 1C0 , 1B0 , DECIMAL
  5
  6 140 CONSTANT QUEUE-SIZE   QUEUE-SIZE ARRAY QUEUE
  7 0 VARIABLE HEAD    0 VARIABLE TAIL
  8
  9 : NEXT DUP @ 1+ DUP QUEUE-SIZE = IF DROP 0 THEN DUP ROT ! ;
 10 : EMPTY?  HEAD @ TAIL @ = ;
 11
 12 : ONQUEUE HEAD NEXT DUP TAIL @ = IF ." QUEUE FULL"
 13   QUIT THEN QUEUE ! ;
 14 : OFFQUEUE EMPTY? IF ." QUEUE EMPTY" QUIT THEN
 15   TAIL NEXT QUEUE @ ;                              -->


SCR # 2
  0 : BYTE-ARRAY <BUILDS ALLOT DOES> + ;
  1 512 BYTE-ARRAY BEST-MOVE
  2
  3 : BIT  ( REPLACE NUMBER ON STACK WITH BIT )
  4   1 SWAP -DUP IF 0 DO DUP + LOOP THEN ;
  5
  6 : NEXT-UNIV OVER OVER GALAXY @ XOR ; ( MAKE A MOVE )
  7
  8 : REPORT  ( DISPLAY A SEQUENCE OF MOVES )
  9   CR FIRST-UNIV BEGIN DUP BEST-MOVE C@ DUP . NEXT-UNIV
 10   SWAP DROP SWAP DROP DUP LAST-UNIV = UNTIL DROP ;
 11
 12 : MOVE-OK? OVER OVER BIT AND NOT ; ( CAN MOVE BE MADE? )
 13
 14 : NEW-UNIV? DUP BEST-MOVE C@ 255 = ; ( BEEN HERE BEFORE? )
 15     -->


SCR # 3
  0 : NOTE-MOVE  ( SAVE BEST MOVE, PUT NODE ON QUEUE )
  1   DUP ONQUEUE BEST-MOVE C! ;
  2
  3 : TRY-MOVE  ( TRY A SPECIFIC MOVE BACKWARDS )
  4   MOVE-OK? IF NEXT-UNIV NEW-UNIV? IF NOTE-MOVE
  5   ELSE DROP DROP THEN ELSE DROP THEN ;
  6
  7 : 9MOVES 9 0 DO I TRY-MOVE LOOP DROP ; ( TRY ALL 9 MOVES )
  8
  9 : SOLVE ( PROCESS ALL NODES ON QUEUE )
 10   BEGIN OFFQUEUE 9MOVES EMPTY? UNTIL ;
 11
 12 : INIT ( INITIALIZE ) 0 BEST-MOVE 512 255 FILL
 13   HEAD @ TAIL !  LAST-UNIV ONQUEUE ;
 14
 15 : SHOOT-STARS  INIT SOLVE REPORT ;          ;S
```

Listing 1. FORTH program to solve shooting stars puzzle.

```
SHOOT-STARS
4 7 8 1 4 1 6 7 2 0 1 4  OK
```

Listing 2. The solution to the shooting stars puzzle found by the FORTH program above.

# HOW TO PRODUCE RANDOM ACCESS VIDEOTAPES, VIDEODISCS AND OTHER INTELLIGENT WONDERS WITH YOUR MICROCOMPUTER

Robert V. Whitney
Whitney Educational Services
2071 Tenth Avenue
San Francisco, CA. 94116
(415) 681 - 4725

## Abstract

Recorded video instructional programming may be combined with interactive computer programs and presented to students via computerized videotape. If desired, the perfected programs of instruction may be transferred to the videodisc as that technology becomes useful.

The interactive training system, CATI (for Computer Assisted Television Instruction), makes use of existing videotapes or newly created video training material. Movies, slides, stills or video camera pictures are recorded on videotape, usually with a small video camera. The videotape machine is interfaced with a microcomputer, in this case the Apple II. The Apple is programmed to find desired segments on videotape and display them on the same screen used for presentation of computer data.

An instructional (CAI) program is written which requires periodic keyboard responses from the student who is presented with lesson material accessed from a combination of computer and videotape.

As the student interacts with the program, appropriate computer data and videotaped material are presented as the system's response to the specific needs of the student.

If the instruction has been prepared for wide distribution, taped programs, complete with computer programming, may be transferred from master videotape to the videodisc format for inexpensive, multiple copy production.

It is NOT possible to record directly to videodisc systems and advantages of the disc are achieved only when large numbers of copies are called for. It is necessary to build the instruction and work out its strategies on videotape initially.

The CATI system developed by the author and his son, David Whitney, gives the instructional programmer the tools with which to shape interactive video programs from existing or new videotapes which may then be presented directly to students with the same equipment.

This paper constitutes an explanation, description and demonstration of the CATI interactive instructional system.

## Background

The appearance of computer assisted instruction brought a new kind of learning activity to a few schoolrooms -- for a few hours at least -- during the past twenty years. Enormous amounts of money have been spent on such systems as Plato and TICCIT and a great deal has been learned from the relatively small number of students who have used computers in the schoolroom.

CAI, with its history of complex hardware, expensive interconnections, and hardwon courseware design has always, nonetheless, encouraged its adherents because of the learning activity it

always seems to generate. Students like the interactivity -- the challenge.

CAI is Socratic. It expects you to do something during your instruction and it will assert a firm and quick opinion about your performance. The feedback you get as a student is immediate, and learning theory holds that that is a good thing. It's also a good thing that the computer is quite willing to wait while you go to the refrigerator and is NOT willing to release you from a difficult or frustrating lesson until you have mastered the material. If it is well programmed, the computer will adjust the instruction to your level of competence -- which is something that films, books and teachers of large classes cannot do.

In spite of all of this, CAI has some formidable problems. Interactive as it may be, the computer talks to you by giving you something to read. Now in this world of noisy TV programs, professional football and stereo rock music, some students may find something missing at the CAI terminal. Developers have not been unaware of this possibility. As far back as 1966, designers started adding sound to data displays on the IBM 1500. Even the simplest of audio support can be quite helpful in capturing attention, especially among students who grew up near pinball machines or pong games. (What would be the effect if your best return shot lost its bonk on the opposite wall of the pong court?)

Speaking of sound, there are some things that can be said a lot better than they can be written, which is why we have teachers. Hearing and seeing a good teacher can certainly help you learn. And, speaking of seeing something more than alphanumerics on the CRT, CAI developers have long been concerned about visuals. Both the TICCIT and Plato projects developed a graphics ability a long time ago. A special plasma screen allows Plato to present interactive graphics with motion capability. And TICCIT uses TV transmission to show illustrative material from a central file. But neither of these sophisticated accomplishments in CAI has approached the audio visual wallop of a good living instructor -- even one recorded on videotape.

The trouble with film or videotape, course, is that it is not interactive. And tape or film are not media we think of as allowing individualized instruction. These media run their courses at their own speed, whether the student follows the instruction, gets hopelessly lost or falls asleep.

Traditionally, videotape and film are seen as expensive to produce, buy or rent. They are also inconvenient. If you don't think so, just ask the teacher who ordered the instruction, rescheduled it when it failed to arrive, hauled in the projector or video cart, dealt with the plugs and buttons and then returned everything. In spite of the enormous advantages of sound, motion and drama capabilities, the unwieldy presentation mechanics of many audio visual forms can waste many minutes of class time for each minute of relevant coursework absorbed. This problem is even more obvious in business and industry where paid time is lost when instruction is inefficient.

## Computer Assisted Television Instruction -- CATI

-- that's what we call it -- and we think is offers some answers. The new aspect is random access of sound and/or motion visuals. The CATI interface allows random access of both computer material and videotape segments at the same time so that interactive instruction flows along, presenting the student with individualized lesson material of any kind. All material is responsive to the individual student as anticipated by the program design.

We'll assign the access job to our trusty Apple II micro. The Apple is interfaced to a small, solenoid activated, half-inch cassette video tape recorder such as the Sony SLO-320 or SLO-323 or the Panasonic NV-8200. These are standard, industrial model Betamax or VHS videotape decks and require no modification whatsoever. Other video machines may be used also.

Our interface is a card which slips into an Apple port. It comes with all connectors, ready to go to work. It costs $ 368.

Control programs, documentation and instructional sample routines are part of the interface package. The only other component required is a standard color TV set to serve as display CRT for all visuals.

This arrangement of standard hardware is used to program and to present the instruction. Programs may include sound and motion visuals, demonstrations by good teachers, computer graphics and individualized CAI style interaction.

A major objective is to eliminate the media hassle. If a student requires material from movies, slides, stills or the horse's mouth to help him/her understand something, the computer will provide it from videotape easily and within seconds.

As with all instruction, the values are found in the course design and in the software. The challenge for the instructional systems designer is to make life as simple as possible for the instructional designer. Having written and gathered the teaching material, we think the teacher should be able to put visuals on tape and then sit down and program the instructional presentation quite easily. The routines have been written to lead the teacher through this process so that he/she may concentrate on the instruction instead of the media.

Another important objective in the CATI system is the optimum use of existing films and videotapes. In many cases, no editing is required. The entire film may be displayed on tape followed by CAI style drill or testing with remediation accessed from the original presentation. With the tape recorders we are using, you may change the audio for a segment used in a remediation branch simply by plugging in a microphone and audio dubbing the new audio on a separate audio track from the original. The computer will switch to the proper track when the time comes.

Our demonstration will actually produce some interactive audio/visual instruction. We will program the system using BASIC and will transfer still and motion visuals to videotape from still photos, slides, video camera and 8 mm. film.

Before we get down to production, we want to clarify, again, why we are not using the much heralded videodisc. A major objective of this demonstration is to show you actual production of interactive video material and you CANNOT record anything on a videodisc. You can only PLAY a videodisc.

The videodisc is cost effective only for quantity production of previously validated materials which must first be worked out, sculptured and tested on videotape. The CATI system is ideally configured for this task. The system will present the material to students and may be used to modify the material on a running basis. Conversion of the perfected material to the disc format can be accomplished from master videotape when the economies of play/only mass production are called for. There is only one significant difference between the two video media and that is the time required to find and play the recorded material. You can compensate for this differential in the production process by combining computer generated material and videotape content. In our instruction, the Betamax player seldom requires more than a few seconds to access needed material. Since system responses to students are combinations of computer and tape material, no delays in interactive activity occur. The computer/videotape interface allows the Apple to perform all functions simultaneously.

Some of the time, the student hears taped audio while working with computer generated pictures. Thus the designer has a talking computer as well as a videotape reproducer. The Apple is kept quite busy as it switches between audio tracks and video sources at any tape location called for in the interactive computer program.

This is the videotape player/recorder: a Sony SLO-323. It has all of the functions you'd expect and a few more: play, fast forward, audio dub, reverse, stop, pause (still frame), dual track audio selection and variable speed (slow and fast motion) in forward and reverse modes.

These new recorders are wonderful improvements over the older versions. You can now review a tape the way you thumb through a book since the picture remains on the screen no matter how fast you run the tape. As stated, there are two audio tracks which can be used for two levels of competence, bilingual instruction or alternative responses. You can even record on one while listening to the other.

Our Apple II has 32 K of RAM memory and a single disk drive. The disk is not necessary in giving instruction;programs may be saved on cassette and eventually stored on the videotape audio track. However, the disk does speed up the programming phases. Smaller user memories are also workable for many programs. If you elect to have your final tape carry audio, video and computer programming you have the advantage of putting the entire training module in one very small package. It also means that multiple copies of the tape can be copied from the original in one operation. And you can pick up a two hour tape at your local Fotomat store. The tape can even be broadcast, computer program and all, for copying off the air. This is an interesting future prospect which would make it possible for students to record each days individualized CATI lesson on yesterday's tape on a home recorder.

Demonstration

Now we are going to load our authoring program from our disk into the computer. This program is called SEARCH. Notice that -- now -- all of the tape machine functions may be operated from the Apple keyboard.

The interface and accompanying software allow the microcomputer to locate videotape addresses by counting "control track" pulses. These pulses are recorded on a special track (not an audio track) and are inherent in the video recording process. Ordinarily, they are used as part of the synchronization system in the recorder. The system produces 30 frames of video pictures each second. The control track is often used in electronic videotape editing.

The computer tells the tape machine to move in the direction of the desired tape address and stops the tape when it gets there. The tape is then either played or waits in "pause" until the appropriate moment called for in the computer program.

The interface allows the computer to do two jobs at once. It can locate tape segments at the same time that it is also carrying on a dialogue with a student.

The interface switches between video sources on computer signal. It can switch as often as called for and as fast as you wish, integrating material from the computer program text and the videotape. This "edit as it goes" feature greatly simplifies production of videotape material since no complex video editing need be done for most subjects. The tape audio tracks may be used to narrate any of the instruction, whether it originates from the computer or the videotape machine.

The video programming process is simple. Visual segments to be used are copied to tape from any source (slides, stills, motion pictures, video camera or another videotape). We will demonstrate each of these transfer methods.

The exact location of the segments placed on videotape is not important at this point, although some thought should be given to keeping related material in close proximity in order to minimize access time.

The programmer views the tape, controlling its functions from the computer keyboard. Upon locating a segment to be used in the instruction, the beginning and end of the sequence are entered into memory with one keystroke each. The segment is then assigned a name which will be written into the instructional CATI program.

Programs may be designed which combine video sources and student responses in such a way that new material flows to the student constantly so that he/she is engaged at all times with no technology-based delays.

The interface can be prepared to control any videocassette machine which

is solenoid operated (remote controllable) and which outputs the control track signal. This includes many Umatic ($^3/_4$ inch machines) as well as Betamax and VHS format devices.

Our designs to this writing have been accomplished in BASIC. We expect that programming in PASCAL and PILOT will occupy programming tests in the next few weeks.

We also expect to extend the control interface to compatibility with other microcomputers, probably the TRS 80 and the Pet in a short time.

# LESSON DESIGN IN PILOT

Robert N. Watkins
Customer Education Manager

The Computer Merchant
5107 El Cajon Blvd.
San Diego, CA 92115
(714) 583-3963 [voice] (714) 582-9557 [ABBS]

Abstract: Lesson design for Computer Assisted Instruction (CAI) consists of psychological and programming factors. This paper focuses on the programming task using the CAI language PILOT. A method of lesson design that makes construction of PILOT lessons easier and less prone to programming error is presented. This method is part of a class in using PILOT that is taught at The Computer Merchant.

Picture this scene: after exposing the dangers of a nuclear power plant, Jane Fonda (as an ace television reporter) examines a typical lesson written in PILOT. "Oh no!", she gasps. "It's the Spaghetti Syndrome!"

And well it might be. What I call the "Spaghetti Syndrome" is the tendency of computer programs to consist of such twisted, convoluted threads of logic that they resemble the proverbial plate of spaghetti. If this is not true from the very start, then certainly as changes and enhancements become necessary the program listing becomes less obvious and clear as to what is to be accomplished.

Whether the program in question is one written in BASIC or a lesson written in PILOT, this occurs too frequently. The comparison with BASIC is intentional, since although BASIC and PILOT were developed for different reasons, they are both prone to "Spaghetti Syndrome" programming. A look at these development reasons suggests why.

BASIC, for Beginner's All-purpose Symbolic Instruction Code, was written to teach computer programming. It is a simplified language, with a limited vocabulary (which has since been extended rather haphazardly by each and every vendor offering it). PILOT, for Programmed Instruction, Learning, Or Teaching, was written for the exact opposite: to avoid having to teach programming! Its purpose was to make lesson writing for CAI easy enough so that professional people-programmers (ie., teachers) would not have to become professional computer-programmers also. The approach was to make PILOT a simplified language, with a limited vocabulary (which also has been extended haphazardly by every vendor offering IT).

The weakness common to both, due to the limited vocabularies, is a lack of sophisticated mechanisms to control the sequence (or flow) of a program or lesson. These are called control structures, and the more comprehensive the control structures at your command, the easier it is to avoid tangled up, error prone programs (or lessons). But (a philosophical aside here) our concepts of structure are all in the mind anyway, attempts to impose order on a sometimes disorganized world. So we see magazine articles on bringing the "wonders" of structured programming to BASIC, and this paper which incorporates structured design techniques into lesson planning in PILOT.

PILOT: A Brief Summary. For those of you not already familiar with PILOT, it has been described in previous Proceedings volumes. But for the purposes of this paper, here is a brief summary.

A PILOT lesson consists of a list of statements in the form:

```
[*label] [command][condition]:[operand(s)]
```

The brackets are not typed; they are my way of indicating that the element contained within them is optional. Thus the only required portion of a PILOT statement is the colon (:).

The label, which must begin with an asterisk (*), is used by several commands to reference other points within a lesson. If a line is not referred to by other statements, it is optional. When present, it is followed by a space. Commands tell the computer what action is desired. If omitted, the command from the previous statement is assumed. A condition is either Y, N, or a conditional phrase such as (I<3) enclosed, as shown, in parentheses. The statement will be executed only if the condition used is true (see below). If omitted, the statement is always executed when reached in a lesson. The operands are different for each command, and sometimes is not used at all.

Eight (8) PILOT commands are called core commands. These are common to all versions of PILOT, and form the nucleus of its capabilities:

```
T:  Type text to user.          T:HI.  WELCOME TO PILOT.
A:  Accept answer from user.    A:
    Accept & store answer.      A:NAME$
M:  Match answer to keywords.   M:ANN,BOB,CATHY,DALE
C:  Compute as in BASIC.        C:Y=(X^2)+X+4
R:  Does nothing; remark.       R:PAUSE SUBROUTINE
J:  Jump to another statement.  J:*PART2
U:  Use a subroutine located    U:*PAUSE
    at label, then return.
E:  Return (if in subroutine)   E:
    End (if in main lesson)
```

Typical extensions to the core commands, which differ from one vendor to another, are:

```
   PR: Begin lesson program.      PR:LESSON PART ONE
CLEAR: Clear video screen.        CLEAR:
  LOC: Move cursor location.      LOC:15,1
 TONE: Sound musical note(s).     TONE:A8,G4,C8
    G: Do graphics command.       G:COLOR=15
                                  G:PLOT X,23
   SJ: Load and start another     SJ:LESSON PART TWO
       lesson from diskette.
```

A Skeleton Lesson. Appendix A is a dummy lesson written in MUSE Inc.'s APPILOT, which runs on the APPLE ][ computer. The following section on lesson design methodology will use it as an example. The lesson is divided into three main parts.

Immediately following the lesson header (PR:) statement comes a collection of utility subroutines. These save repetitive typing within the lesson and are called via the Use (U:) command. They perform such functions as accepting a yes or no answer, or delaying the lesson a fixed amount of time.

After the utility subroutines is the main lesson logic, which I call the "script". It is a sequence of statements that outlines the topics to be covered. No Jump (J:) commands are used in the script area -- it proceeds directly from top to bottom then quits with and End (E:) command.

Next come a second set of subroutines, which are called by the script. These can be individual screenfuls of text, called pages (or frames), or control structures that call pages as needed. An individual page can either present information to the user (a text page) or test the user on material previously presented (a quiz page).

The Lesson Design process. Lessons seem to collect in groups. When an author writes a lesson on a topic and either runs out of room in the computer or covers too much material, he/she decides to split the lesson into two. So let's start at the course level, and plan lesson series from the outset. One possible hierarchy (with an example) is as follows:

CAI LIBRARY
  Subject
    Course
      Lesson
        utility sub.
        script
        page sub.

THE COMPUTER MERCHANT
  Computer Programming
    Using APPILOT
      Lesson Design Methods

Once you have researched the course you want to present, and subdivided the material into subtopics of reasonably short length, it is time to design the lessons themselves. The procedure below can be applied to each lesson in turn. The result will be a coherent, consistent group of lessons.

1. Make a list of concepts the lesson is to cover. Don't worry about sequence yet, just put them all down as you think of them. And you can always add more later.

2. Evaluate each concept in terms of what they student must be able to do with it after the lesson is successfully completed. Give each a rating, as follows:

Level One: student need only be exposed to the concept.

Level Two: student must be able to recognize the concept when it is described.

Level Three: student must be able to explain the concept in his/her own words.

Level Four: student must be able to recall the exact statement of the concept.

Level Five: student must be able to recall and apply the concept in the solution of problems.

3. Eliminate Level One concepts in so far as it is possible. Generally, these will only confuse the student by forcing him/her to expend effort on relatively non-important items.

4. Sequence the remaining list in the order you want the concepts presented. Assign a PILOT label to each concept -- these will be used in the script to identify that section of the lesson. For a lesson on automobile parts, typical labels might be *OVERVIEW, *PISTON, *CARB, and *CYLINDER .

5. Start the script portion of your lesson by writing a Use (U:) command for each of the labels you defined above, in the sequence that you numbered them on your list. Follow these by an End (E:) command.

6. Start the pages portion of your lesson by writing one text page for each concept, each page starting with a remark that has the appropriate label on it. If the concept is very short, it can be fully explained on the page in detail. Usually, however, this will be an introduction for other text pages to follow.

7. Put your skeleton lesson on the computer. Note that up to this point, everything has been pencil and paper work. Now is the time to test your work so far, and assure yourself that the upper level framework is correct. This is easier to do now because a lot of detail is not yet present.

8. Flesh out the skeleton lesson by adding further text pages, quiz pages, and/or control structures (see Appendix B). Be sure to add a Use (U:) command to your script in the proper sequence for each new page or control structure.

Quiz pages should occur frequently (every two or three text frames at most) and be keyed to the mastery level you assigned each concept. Level One (exposure) concepts need not be tested at all. Level two (recognition) quizes may take the form of a multiple choice question, where the question text describes the concept and the choices are among the concept and several close imitations.

Level Three (own word repetition) and Level Four (exact repetition) can use

a similar scheme. The concept is given in the question text, and the user is asked for the definition. Your Match (M:) command will have many synonyms if you're testing for Level Three comprehension, since the user is allowed to use his/her own words. For Level Four, of course, no synonyms are allowed -- only the exact description will match.

Level Five is the most difficult to test. Two useful types of questions, though, are word problems (that we all learned to hate in Math class) and either/or questions (a favorite of IQ tests). In the former, you pose a problem that requires the user to apply the concept to get the correct answer. An example would be to display a simplified Balance Sheet for a business, and ask for the Current Ratio. The either/or type of question presents two statements involving the concept, then asks whether statement 1 only, or statement 2 only, or both, or neither is correct. The statements require the user to first remember the concept, then apply it to test the two statements in his/her mind.

Quiz pages should respond with enthusiasm or encouragement (based on the answer). There is no place for a smug computer that ridicules wrong answers with "You blew it, turkey!" or similar. Such dialog doesn't reveal an immature computer...just an immature lesson author. Encourage the user, build his/her confidence and skill together.

9. Packaging the finished lesson should receive some consideration. APPILOT itself makes a good example. It is distributed on diskette, enclosed in a 1/2" thick three-ring binder. In your binder you could have a short instructions page, followed by a plastic "floppy pocket" page containing the diskette, followed by any illustrations, diagrams, or supplementary reference material desired. Since a diskette will hold all the lessons for a course, use regular dividers to separate the printed material for each lesson. This keeps everything together and presents a good appearance.

10. Finally, after you have tested it, and it has been in public use for a while, review the lesson. Are the facts still true? Should certain key words be included in the Match (M:) commands, based on your experience? Maybe you have thought of a clearer way of describing a concept. If you have constructed your lessons properly, the effort to edit and even to add whole new pages will be minimal. And you'll be able to follow easily what you did when you originally wrote it.

11. And, may I suggest sending us a copy? We're actively involved in PILOT for the APPLE ][ computer, and hope to serve as a central exchange for courseware. Both public domain (free to all) and proprietary (royalties requested) courseware would interest us.

I hope these notes have been helpful to you who are using PILOT, whatever the version, and serve to entice you who have not yet discovered how powerful and easy to use PILOT really is.

APPENDIX A

A Skeleton APPILOT Lesson

This skeleton lesson can be used as a base on which to build your own APPILOT lessons. Because core commands were used throughout, you can probably adapt it for other PILOT versions as well. The only APPILOT dependent commands are explained in the text: you can substitute the facilities provided in your implementation.

```
PR:LESSON PART ONE
J:*SCRIPT
R:**************************************
R:*THIS IS THE UTILITY SUBROUTINE     *
R:*AREA.                              *
R:**************************************
*RETURN R:WAIT FOR USER TO PRESS RETURN
LOC:23,1
T:PLEASE PRESS RETURN TO GO ON.
A:
CLEAR:
E:
R:**************************************
*PAUSE R:WAIT A SHORT WHILE THEN GO ON
C:DELAY=0
*PAUS1 R:
C:DELAY=DELAY+1
J(DELAY<25):*PAUS1
E:
R:**************************************
*YESNO R:ACCEPT YES OR NO ANSWER
A:
M:YES!Y !YEAH!OK!YUP!AFFIRM!YUH
E:
R:**************************************
R:*     MAIN LESSON SCRIPT            *
R:**************************************
R:YOUR LESSON BEGINS HERE
U:*PAGE1
U:*PAGE2
R:    ETC.
E:
R:**************************************
R:* TEXT AND QUIZ FRAMES GO HERE      *
R:**************************************
*PAGE1 R:YOU MAY DESCRIBE PAGE HERE
T:THIS IS THE FIRST LESSON PAGE.
T:THE TEXT ON THIS PAGE WILL BE TYPED
T:TO THE SCREEN.
T:
T:THE U: COMMAND WILL ASK THE USER TO
T:PRESS RETURN WHEN READY TO GO ON,
T:THEN CLEAR SCREEN FOR NEXT PAGE.
U:*RETURN
E:
R:**************************************
*PAGE2 R:
T:THIS IS THE TEXT TO BE DISPLAYED ON
T:PAGE TWO.
T:
T:
T:QUICK NOW, ON WHAT PAGE (SPELL IT
T:OUT) IS THIS TEXT TO BE DISPLAYED?
T:
T:PAGE  - - -.
A:
M:TWO
TY:THAT'S RIGHT!  THIS IS PAGE TWO.
TN:SORRY, NO.  THE RIGHT PAGE IS TWO.
U:*RETURN
E:
R:**************************************
R:*  TH-TH-TH-THAT'S ALL, FOLKS!      *
R:**************************************
R:NO E: IS NEEDED HERE, SINCE THE TRUE
R:END OF PROGRAM HAPPENS AT THE END OF
R:THE SCRIPT PORTION.  SEE?  THERE'S
R:ONE RIGHT AFTER ALL THE U: COMMANDS.
```

APPENDIX B

Sample Control Structures for use within Page subroutines

In general, the Jump (J:) command should only be used to send control to a label WITHIN THE SAME SUBROUTINE as itself. A subroutine should have only one entry point -- the label -- and only one exit point at the End (E:) command.

The exception is at the program's beginning, where it is used to jump over the utility subroutines. Of course, the utility subroutines could be placed elsewhere. In APPILOT they are placed at the front because this speeds execution time on the APPLE ][ significantly.

The control structures discussed below are meant to give YOU greater control over your program. Limiting jumps to local areas only makes the most common source of errors much less likely. Following the design of the control structures likewise assures you that the lesson will flow as you wish, and not develop unpredictable quirks. And the lesson will be more readable to you (less Spaghetti Syndrome!).

```
R:*********************************
*SIMPLE R:
R:CONTROL STRUCTURE TYPE 1
R:THE SIMPLE YES/NO DECISION
R:
T:SET THE Y/N FLAG BY A MATCH
T:STATEMENT OR ANY OTHER WAY
T:AVAILABLE IN YOUR PILOT.
T:
TY:THEN PREFIX  ALL STATEMENTS
TY:THAT SHOULD EXECUTE WHEN A
TY:DECISION IS TRUE WITH THE
TY:Y CONDITIONER.  BE SURE TO
TY:DO THEM ALL!!
R:
R:REMARKS ARE USEFUL TO SEPARATE
R:CONDITIONS.  USE THEM OFTEN!
R:
TN:LIKEWISE ALL STATEMENTS THAT
TN:SHOULD BE EXECUTED WHEN THE
TN:DECISION IS FALSE SHOULD BE
TN:PREFIXED WITH THE N CONDITION.
R:
E:
R:**********************************
*WHILE R:
R:CONTROL STRUCTURE TYPE 2
R:LOOP EXECUTED 0,1, OR N TIMES
R:WILL BYPASS IF CONDITION FALSE
C:I=1
*WHILE1 R:
J(I>5):*WHILEND
T:THIS IS A LINE.
C:I=I+1
J:*WHILE1
R:
*WHILEND R:
E:
R:**********************************
```

```
*UNTIL R:
R:CONTROL STRUCTURE TYPE 3
R:LOOP EXECUTED 1 OR N TIMES
R:WILL ALWAYS EXECUTE AT LEAST ONCE
R:THIS ONE DOES THE SAME AS *WHILE.
C:I=1
*UNTIL1 R:
T:THIS IS A LINE.
C:I=I+1
J(I<6):*UNTIL1
R:
R:NOTE AT EXIT TIME I=6 IN *UNTIL,
R:BUT I=5 IN *WHILE.
E:
R:**********************************
```

```
R:**************************************
*COMPLEX R:
R:CONTROL STRUCTURE TYPE 4
R:COMPLEX DECISION (>2 POSSIBLES)
R:
T:NOTE THE CAREFUL USE OF THE J:
T:COMMAND IN THIS STRUCTURE.
T:ASSUME X ALREADY SET BY C:
T:COMMAND TO 1,2,3 OR 4
R:
J(X>1):*COMPLEX2
T:THIS GROUP GETS EXECUTED ONLY IF
T:X=1.  NOTE THE JUMP AT THE END
T:OF THIS AND EVERY SUBGROUP THAT
T:SENDS CONTROL TO THE ONE END.
J:*COMPEND
R:
*COMPLEX2 R:X=2 HERE.  USE REMARKS!
J(X>2):*COMPLEX3
T:NOTICE THE CREATIVE LABELLING
T:TOO.  USE EVERYTHING YOU CAN TO
T:MAKE YOUR LESSONS READABLE.  YOU
T:NEVER KNOW WHO WILL READ THEM
T:LATER...MAYBE EVEN YOU!
J:*COMPEND
R:
*COMPLEX3 R:X=3 IN THIS GROUP.
J(X>3):*COMPLEX4
T:YOU CAN DO OTHER THINGS IN A
T:SUBGROUP BESIDES TYPE.
U:*RETURN
U:*PAGE23
T:DO YOU UNDERSTAND NOW?
U:*YESNO
TY:GREAT. NOTICE THIS SIMPLE
TY:DECISION HERE TOO? SLICK, EH?
TN:THAT'S TOO BAD. I SUGGEST
TN:ANOTHER READING OF THE PAPER.
J:*COMPEND
R:
*COMPLEX4 R:X=4 IF WE GET HERE.
T:ERROR CHECKING LOGIC SHOULD BE
T:PLACED AS CLOSE AS POSSIBLE TO
T:THE A: COMMAND THAT OBTAINS AN
T:ANSWER.  WE ASSUME THAT X WAS
T:SET EARLIER BASED ON A VALID
T:INPUT.
T:
T:NOTE THE JUMP BELOW.  A GOOD
T:HABIT, ALTHOUGH NOT NEEDED HERE.
T:IF YOU WANT TO ADD ANOTHER SUB-
T:GROUP LATER, THE LOGIC IS ALREADY
T:IN PLACE.
J:*COMPEND
R:
*COMPEND R:
T:STATEMENTS COMMON TO ALL CAN BE PUT
T:HERE IF DESIRED, TO AVOID EXTRAS.
U:*RETURN
E:
R:**************************************
```

# AN APPLE FOR THE TEACHER - A Graphic CAI Authoring System

Ted Perry
San Juan Unified School District, KYDE TYME Project
2331 St. Marks Way, Sacramento, California 95825
(916) 487-7517, 485-0619

This is a progress report on two cooperating Title IV-C Projects, the KYDE TYME Project in the San Juan Unified School District in Sacramento, California, and the CHIPS Project at California School for the Deaf in Berkeley, California. These two projects are funded to develop a computer assisted instruction author language for the microcomputer. This language is to be easily usable by the "novice" teacher in CAI. The authoring system necessitates no programming expertise on the part of the teacher and literally walks the authoring teacher a step at a time through building a student curriculum. This authoring system makes full use of the graphic capabilities of the Apple computer.

At this time the "author language" is complete and running, and I would like to describe and demonstrate it and request constructive feedback. Our author language program consists of several parts:

1. TEACHER AUTHORING PROGRAM
   Allows the teacher to make use of the graphics library and combine graphic images with text for presentation to the student. The teacher also inputs the correct and incorrect answers and feedback appropriate to each response. This program is utilized only when writing a lesson.

2. STUDENT PRESENTATION PROGRAM
   Presents the lesson screen to the student and allows a student to interact with the previously authored program. It brings together the text and graphics and presents them in an orderly fashion.

3. GRAPHICS DEVELOPMENT PROGRAM
   Develops graphic images for the graphics library. It enables the author to quickly create new images, utilize portions of old images or combine images.

4. GRAPHICS LIBRARY PROGRAM
   A resource which is used when "authoring" a program. The teacher requests images from the graphics library and puts them into the lesson that is being written.

5. DATA MANAGEMENT PROGRAM
   Keeps track of student progress, does an analysis of student errors, and is actually a computerized lesson planner which enables the teacher to set up a sequence of lessons in an order that matches the individual needs of each student.

It is difficult to decide where to start in describing this multi-faceted author language. Which comes first is hard to determine. Perhaps the best way to describe the authoring program is to invite you to look through my eyes as I write a single computer assisted instruction lesson.

After the authoring diskette is put into drive one and a graphics library diskette is put in drive two, the authoring program is booted. The prompt then asks me if I wish to work on an old lesson or add a new lesson. In this demonstration, I would like to create a lesson from scratch so I respond with an "A", meaning "Add a new lesson".

I am now confronted with the lesson characteristic page which asks me to determine the following:

1. which area of the screen should be designated as the text window,

2. whether I want upper and lower case or only upper case student input,

3. what size and color of text, and

4. how fast should text be presented.

I choose to make the lower 2/3 of the screen the text window and to make upper and lower case indistinguishable as the student presents it to the computer. I will print in large letters (double size) that are orange. I choose to have normal text and graphics as opposed to inverse and I would like the printing speed to be 255 (as fast as it can appear on the screen).

The computer then asks me if the lesson questions should be presented in random order or sequential as I enter them and how many tries the student should have before the computer does not present that question again. I choose random presentation with three tries before the computer withdraws that question. I tell the computer to give the student the correct answer after two tries. (Actually, I made all these choices with two key strokes by using the default parameters I previously put in the program. These parameters can be easily changed.)

After printing "Next" on the computer, I am presented with a blank screen and the word "Image" at the bottom. The computer is asking me what graphic image I want. I can choose pictures from the graphic library to use in a lesson as I am building it. After typing the word "Tree" (which I know is on the graphics diskette), the computer responds by displaying the colored image of the tree on the screen and I manipulate its position utilizing the paddles. I repeat this sequence for two more trees, a frog, witch, and a car. The upper third of my screen is filled with graphic images.

The computer then asks me "Label?" and gives me an opportunity to place labels on or near each of the images. I choose to label the witch and the car.

In the next step as I am waiting for the computer to respond, a sign across the screen says "Saving Graphics Portion". After the diskette stops, the computer prints "Textual Portion" with a flashing cursor. I type as follows:

| | |
|---|---|
| TEXT: | "There are several objects on the screen. How many do you see?" |
| RIGHT ANSWER: | "5" or "Five" |
| FEEDBACK: | "You really did a good job counting. Yes, there are 5 objects on the screen." |

Next the computer asks me for expected wrong answers. I respond with other expected wrong answers and give feedback.

| | |
|---|---|
| WRONG ANSWER: | "1", "2", "3", "4" |
| FEEDBACK: | "Yes, you are right but I think there are even more objects on the screen. Please try again." |

The computer asks for more wrong answers and I press RETURN to state anything else will be considered a wrong answer and I give it feedback.

| | |
|---|---|
| WRONG ANSWER: | CARRIAGE RETURN (the empty set) |
| FEEDBACK: | "No, that is not correct. Please try again." |

Having finished with question one, the computer now gives me several choices. I may use the same graphics and add more text and questions or I may change the graphics as I add additional materials, or I can quit. I will add one more question in a shortened format, showing you only what I enter.

| | |
|---|---|
| TEXTUAL PORTION: | "Some of the objects are trees. How many trees do you see on the screen?" |
| RIGHT ANSWER: | "3" or "Three" |
| FEEDBACK: | "Sure is hard to fool you." |
| WRONG ANSWER: | "4", "5", or "Four", "Five" |
| FEEDBACK: | "Yes, there are that many objects but not all of them are trees." |
| WRONG ANSWER: | CARRIAGE RETURN |
| FEEDBACK: | "I think you need to look again." |

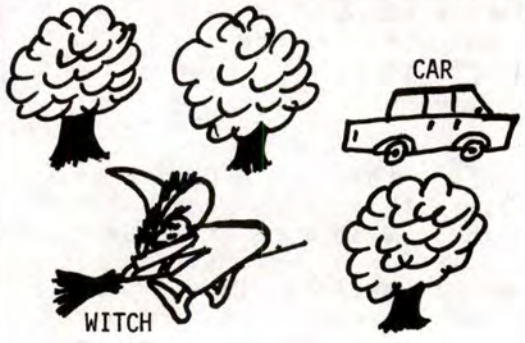
At this point I choose to end my low-level development task and as I tell the computer I am finished, it saves the text questions to the diskette, asks me to give the lesson a name, category, and define the approximate grade

level of usage. It also asks me to define author, date of development, and school of origin. Upon completion of those tasks, I am asked for a 70 character description of the lesson. We have termed this short description a "one-line zinger"! i.e., Given several objects, the student will respond with the appropriate number in each sub-group.

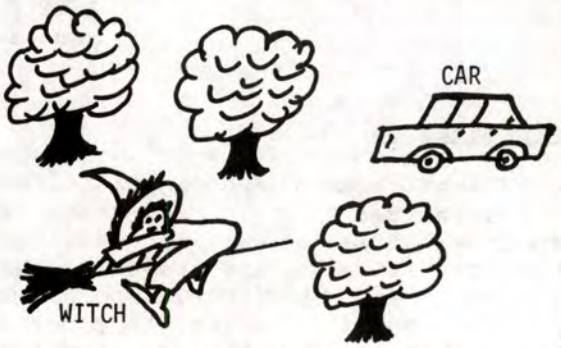
Having completed authoring a two-question lesson, I proudly press RETURN and watch the computer combine the text, graphics, and demographic portions onto a diskette to be presented to the student.

Unable to contain myself, I must run the student diskette to see my masterpiece in action!

The computer presents me with the graphic images across the top of my screen and asks me the appropriate questions.




COMPUTER: "There are several objects on the screen. How many do you see?"
MY RESPONSE: "7"
COMPUTER: "No, that is not correct. Please try again."




COMPUTER: "Some of the objects are trees. How many trees do you see on the screen?"
MY RESPONSE: "3"
COMPUTER: "Sure is hard to fool you."

Pleased with the performance of the computer but not the quality of my curriculum, I turn off the computer and begin revising the presentation.

This presentation is meant to be an introduction to the workings of our multi-faceted author language. In the future I will describe the workings of graphics author language which allows the development of graphic images. It has the capabilities of HIRES line drawing and manipulation of HIRES characters, shapes and images. Included are routines to fill irregular shapes with color and utilize graphic fonts.

We are nearing dissemination time. If you are interested, please write:

Ted Perry
KYDE TYME Project
2331 St. Marks Way
Sacramento, CA 95825

Geoff Zawolkow
Calif. School for the Deaf
2601 Warring Street
Berkeley, CA 94704

# CAI: A DIFFERENT WAY

Jeff Levinsky
Computer Systems Design Group
3632 Governor Drive
San Diego, CA 92122

## Introduction

The teaching machine has characterized much of computer assisted instruction in the past years: students are presented with "frames" or questions, and must provide short answers to proceed. This style of learning can be slow and uninteresting, especially as the subject matter and the method of presentation is fixed. In order to provide a richer CAI environment, a group of educators and computer scientists have developed the GROW system [1], in which the knowledge stored in the computer is mutable, and reflects opinions of students as well as those of teachers. Also, the GROW system allows CAI programs to be both written and used in a variety of styles and formats.

GROW currently operates on several microcomputer systems and on one large computer system. The first installation was on the CHAOS II system (8080-based) at Clairemont High School in San Diego, California [2,3]. A similar version now runs on the Northstar Horizon (Z80-based) and is used at several schools in Palo Alto, California. A yet more complex version operates on the APPLE II computer and can be used in conjunction with an APPLE network developed especially for GROW. Finally, an INTERLISP version (for the DEC 20) exists, primarily for experimentation. GROW may be used at simple or very advanced levels, and is currently utilized by students in grades 7 through 12, teachers, curriculum developers, and independent researchers in a number of different areas of education.

## Basic Structure of Grow

The usual explanation of GROW describes the system as a collection of "nodes", where a node somewhat resembles a typical CAI frame. Specifically, a node has a textual description, which is printed when the node is first entered by a student, and a set of responses which the node makes to various statements (requests or answers) by the student. Typically, the description might pose a question which the user would answer with some input. That response would be matched against the set of responses for the node - if a match is found, then some actions associated with that particular response of the node are taken. In a simple GROW system, there is a very small set of possible actions:

| action | shorthand |
|---|---|
| print a message | P |
| add points to user's score | + |
| subtract points from user's score | - |
| go to another node | G |
| quit | Q |

At this point, GROW somewhat resembles languages such as PILOT, although GROW is more structured in that nodes are separate entities instead of merely different parts of a common program. A typical GROW node, written using the above actions, might be:

```
description
   WHO DISCOVERED AMERICA?
response
   CHRISTOPHER COLUMBUS
actions
   + 10
   G NODE72
response
   LEIF ERICSON
actions
   + 15
   P VERY GOOD
   G ICELAND
```

Upon entering this node, one would be asked:

WHO DISCOVERED AMERICA?

If the student responded with:

CHRISTOPHER COLUMBUS

he or she would be awarded ten points and sent to node NODE72. If the student responded with:

LEIF ERICSON

he or she would receive fifteen points, the message "VERY GOOD", and be sent to the node named ICELAND. Upon entering any other response, a default node would be used to try to determine a match: the default node recognizes messages such as "HELP" and "QUIT". If even the default node cannot recognize the reply, then GROW prints a random remark such as "I DON'T UNDERSTAND".

Using this structure, traditional CAI programs can be constructed. However, GROW can be used in several advanced ways. One basic feature of all GROW systems is that, at any time while the student is using the system, the nodes may be extended by the student! There are two ways that this may happen in the simple level GROW: by adding a new response (and the corresponding actions to be taken for that response) to the current node, and by building new nodes. The first is accomplished by merely typing "EXTEND" and then the response(s) and action(s). The second form of extension occurs more or less automatically - whenever a node is entered for the first time, the person entering it is asked to provide the description for the node. Then, using "EXTEND", the person may provide responses for the node as well. By virtue of these extensability mechanisms, the 'knowledge' in GROW is constantly expanding as students use the system (hence the name). Some of the implications of this essentially unrestricted growth will be examined below.

To clarify how nodes are used, extended, and interconnected, we present an information booth with analogous behavior (see Figure 1). The booth has a name, a general description, and a book of responses and corresponding actions (Figure 2). Each page of the book might contain one set of responses and the actions to be taken accordingly. A person trying to get information from the booth first reads the description and then makes

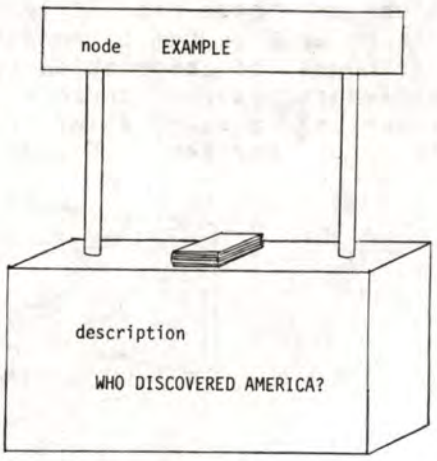



Figure 1

inquiries. These inquiries are looked up in the book, and the corresponding reply (action) is

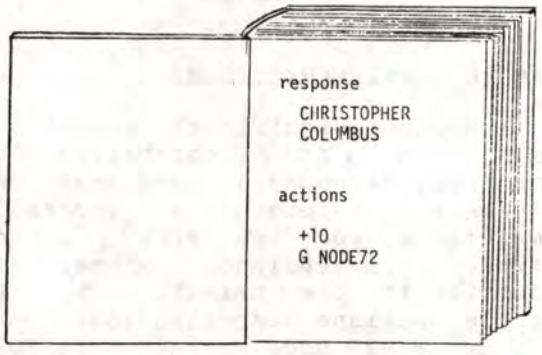



Figure 2

made. At the end of the book are blank pages - the person at the booth may add new responses and

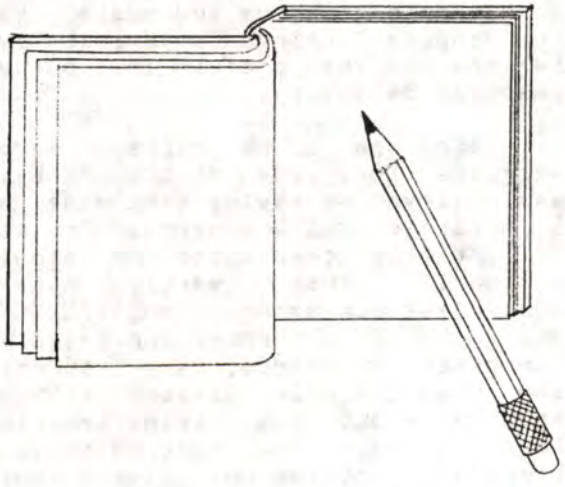


Figure 3

actions on these pages (Figure 3). An entire GROW system is essentially a warehouse of information booths, with perhaps several persons (each representing a user) going from one booth to another (Figure 4).

```
10 PRINT "WHO DISCOVERED AMERICA?"
20 KEY "CHRISTOPHER COLUMBUS"
30 SC=SC+10
40 ENTER "NODE72"
50 KEY "LEIF ERICSON"
60 SC=SC+15
```

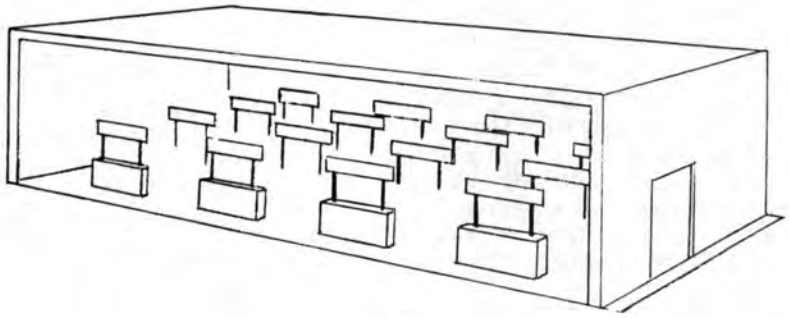


Figure 4

Significantly, in this analogy, each person has the freedom to wander about and to add information anywhere in the system.

## Higher Level Programming

Another significant aspect of GROW is the capability of constructing nodes in more than one language. This is currently implemented for the APPLE II and INTERLISP versions where, in addition to the primitive set of simple actions described above, one may code a node using APPLESOFT BASIC or INTERLISP, respectively. This immediately opens up a number of possibilities to the person creating or extending nodes. For example, nodes can be written to perform arithmetic calculations, to present randomized math exercises, to generate graphics and music, and to compare strings in various ways (which can be useful in better response handling).

The use of a higher level language for GROW on the APPLE II was achieved by having each node be a separate BASIC program. One may still extend nodes using the simple actions: this merely causes equivalent statements in APPLESOFT BASIC to be generated and added to the node. Of course, many students use the simple actions without knowing about the transformation that occurs. The node below is a translation of the one given above into BASIC:

```
70 PRINT "VERY GOOD"
80 ENTER "ICELAND"
```

As is evident, a few statements, such as "KEY" and "ENTER", have been added to the BASIC to permit GROW actions to be performed. "KEY" indicates to what remark the node will be respond and "ENTER" corresponds to the "G" action which one uses to go to a new node. Use of the high-level language to encode nodes does not change any of the semantics of GROW; in particular, a node written in BASIC may be extended or edited (in the normal BASIC manner) while the GROW program is running.

A few other features in the APPLE II version of GROW are of interest. In addition to the "KEY" statement, which merely gives a string that the user's response is to be checked against, there is also a "KEYIF" statement which allows the test to be made just as in an "IF" statement. For example, one might say:

```
90 KEYIF VAL(S$)=3
```

to test if the user's input (which is always in S$) is equal to three. Variables in these nodes are all local or global, that is to say, local to a particular node or known globally throughout the particular GROW program. Local variables retain their values over a node's lifetime, so that a node can 'remember' how many times it has been entered, for example. Nodes

may also call upon a standard set of subroutines to perform tasks such as statistics collection. Finally, there are ways to establish demons, which are special programs that are executed whenever certain events occur. For example, a demon might be executed whenever a student types in a response - the demon could then count the number of times that the student extends certain nodes. Besides monitoring usage, demons can also correct spelling mistakes made by users and represent information that is too global to belong in particular nodes.

## Teaching with GROW

Given the above technical description of GROW, the discussion will now emphasize how GROW is actually used in the classroom. To introduce the simple version of GROW to students, a short manual, known as THE GROW BOOK, was developed. It begins by introducing how to use a computer terminal and by asking a student to proceed through a sample GROW session. The first session is intentionally made to resemble the game of ADVENTURE, and has the student enter a castle and try to solve various problems to move from room to room. For example, to exit from the dungeon of the castle, the student must state the number of colors in the rainbow.

Learning how to extend GROW is postponed until the second lesson so as to give the students time to understand the action/reaction and modular basis of the system. The second lesson asks the student to return to the castle example and to expand a node to reply that "YOU WON'T GET ANY HELP HERE" if a user ever inputs "I NEED HELP" or "HELP ME". The general rules for extending nodes are then given, along with four of the primitive actions (and some pitfalls to avoid).

The third lesson provides some hints and simple tricks (that is, the detailed technical information that should not concern all students). In that lesson, topics such as the default node, the random responses given when GROW fails to match up an input, and the two remaining primitive actions are discussed. By the end of this lesson, the entire simple GROW system has been presented, and students are expected to explore and extend existing GROW systems.

More advanced students eventually grow bored with the primitive actions and are interested in using BASIC to write nodes in. As GROW permits the different languages for extensions to be used interchangeably, a student does not have to unlearn anything in order to use the advanced form. Naturally, the student must solidly understand BASIC programming and must feel comfortable with the idea of interconnected programs, each with its own preserved state. [Many students appear to believe programs retain the value of their variables after being stored and then retrieved via a disk, anyway.] Students who have already learned techniques in BASIC, especially for graphics or sound generation, can immediately apply those techniques when writing nodes in BASIC.

## Motivation

Early experience with GROW in various schools has been positive in that student interest is high. This may be attributed to various ideas in the design of GROW. One important consideration is that the student should perform an active rather than a passive role when using a computer. The student should not mindlessly press buttons, or type short answers, to respond to brief questions, especially to those posed equally mindlessly by the computer. Good examples of active uses of computers are programming and searching through databases for particular pieces of information, and GROW supports both of these functions. Another common form of active learning is the process of organizing and communicating (to others) one's knowledge or ideas: GROW supports this too. In fact, a common anticipated use for GROW is to have teachers develop a relatively straight-lined curriculum for a particular discipline or topic and then to have students edit and add to the nodes to clarify and strengthen the material.

The malleable nature of knowledge in a GROW system seems to be important in how students use the system. To be concise, GROW encourages a form of grafitti. Some teachers have objected to GROW because of this aspect, and have expressed the fear that the disks will be quickly filled with all manners of outrageous nonsense which will, in turn, obscure whatever base system the teacher or curriculum developer initially provided. While this is conceivable (and provisions for locking nodes do exist to counteract this), students generally play by the rules. Often, the 'grafitti' will serve a constructive purpose by indicating where in the curriculum students are becoming confused, irritated, or simply bored. Ideas presented by authority can be attacked (and even corrected) by students who can see the actual issues clearer. At any rate, emphasis should not be placed upon the students who do fill disks (and thereby are learning to program and communicate verbally) - rather the concern should be for the students who do not.

Emphasis of communication skills is one of the fundamental aspects of the GROW system. Communication between students is supported by having students share disks (in some versions, a number of students can be in the same system at once). As each student knows that others will use (and perhaps extend) his or her nodes, extending the network becomes a highly creative process. Students tend to enter jokes and clever traps (this is encouraged at first) but useful statements in the more traditional sense are also prevalent. Knowing that one's peers are going to scrutinize and evaluate what is entered forces some care in matters such as spelling and grammar (far more so than in the average composition course where only the instructor ever reads a student's writing). An issue here is whether a student's name should be tagged automatically to that student's extensions (anonymity is the current practice as no great offenses have occurred).

Finally, GROW does not depart from the "make it gamelike" axiom of educational personal computing. The system may be viewed as the combination of the common computer games of ANIMAL and ADVENTURE, as THE GROW BOOK indicates. This makes learning about GROW comfortable and familiar. Of course, the ANIMAL/ADVENTURE style is by no means the one with the greatest educational value, and students should go beyond it frequently.

## GROWing up

The APPLE II GROW system is currently being expanded to fill the needs that classroom experience makes evident. One facility being added is a network for APPLEs which will enable nodes to be shared between machines. In terms of the earlier analogy, a network allows information booths to be transferred between warehouses, as each warehouse represents a processor (Figure 5). Such a network could also support a hard disk (and thus help overcome the critical storage problems of floppy disks). Other improvements would permit GROW to preserve graphical pictures on the screen as well as the values of variables.

On another level, the basic structure of GROW is evolving towards similar ideas in artificial intelligence. It may be recognized that the nodes in a GROW system form a structure very much akin to "semantic networks" in that both try

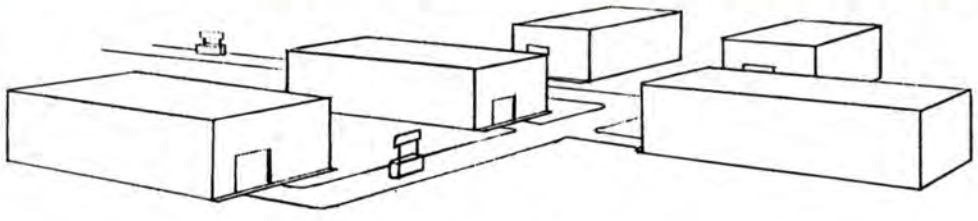


Figure 5

to represent through objects and interconnections some "knowledge". GROW does not allow the rich modes of links between nodes that most semantic networks have, but it does provide powerful procedural specifications for nodes. One possibility is to isolate the access methods used by students and teachers (known as "interfaces") from the actual nodes (which then form an "information network"). Such a separation would allow many different interfaces, including ones for strict database style access, game access, and tutorial construction, for example. At the same time, the representation of the information itself can be upgraded.

## Summary

GROW is an attempt to apply non-traditional solutions to some of the traditional problems of CAI. Specifically, to counter the sterile and passive nature of many CAI presentations, GROW encorporates several basic motivational techniques and allows for creative flexibility. This includes having the student participate in an active role, having 'knowledge' be mutable and controversial, and having evaluations be carried out by other individuals rather than by machines. With GROW, the computer need not play the role of teacher but can instead be viewed only as a tool for learning and for reference. And, it is hoped, the variety of presentation and the subjectivity of the communication will permit GROW to be used in educational disciplines in which CAI has been typically ignored. Finally, GROW is designed for many levels of users: beginners need not know even that the system is extendable, while advanced users can develop nodes and systems of nodes using the powerful tools and languages available.

## Acknowledgements

Mary Kroening, Robert Haas, Joan Targ, Peter Schorer, and George Sellman are among the individuals who have been instrumental in the development and classroom testing of GROW. John Harris and Hewlett-Packard Laboratories provided assistance with the preparation of this paper.

For further information on GROW, please write to the Computer Systems Design Group, 3632 Governor Drive, San Diego, CA., 92122.

## References

[1] Levinsky, Jeff L. "Grow". Creative Computing. To appear.

[2] Levinsky, Jeff L. "CHAOS: An Interactive Timeshared Operating System for the 8080". Dr. Dobb's Journal, XXXI (January, 1979), 6-13.

[3] Kroening, Mary E. and Levinsky, Jeff L. "CHAOS II: Design and Implementation of a Timeshared Microcomputer System", in NCC '79 Personal Computing Proceedings. AFIPS, 1979, 689-698.

# PROGRAMMING FOR EVERYONE:
# A RATIONALE AND SOME TEACHING STRATEGIES

William J. Wagner, Ph. D.
Mountain View High School
Mountain View, California 94041

Abstract

The importance of offering non-technicalinstruction about computers is discussed, and the importance of introducing these students to programming is argued. Two teaching experiences of the author are described in detail: a high school programming class with no math pre-requisite, and a community college short course about computers.

## INTRODUCTION

Inexpensive and easy to use microcomputers offer the possibility of expanded use of computers, both into new areas and by people who have never before considered using them. But the expansion beyond the naturally motivated population does not necessarily follow the introduction of new equipment or the offering of new courses. Certain individuals are naturally attracted to computers and quickly pick up the skills and vocabulary necessary to make progress. But most people, whether because of lack of opportunity, low confidence, or the high level of most computer offerings, find it difficult to gain this experience and knowledge about the field.

This paper addresses the problem of bringing computers to individuals who do not normally seek them out and whose experience and opportunities are limited. First, the reasons for broadening the scope of computer education are presented. Then an argument is presented for the inclusion of elementary programming in any such non-technical introduction to computers. Finally two experiences the author has had teaching such a course are described - in a high school programming class with no math pre-requisite, and in an inservice course for non-technical adults offered through the local community college.

## WHY INTRODUCE COMPUTERS TO MORE PEOPLE?

The fundamental reason for seeking to broaden the knowledge about computers in the general population is to work against the continued widening of the gap between technical and non-technical people. Even in the Santa Clara Valley, the symbolic center of the computer and electronics industry, the general population is distinctly stratified in terms of computer knowledge and opportunity. Despite the tremendous growth and sales of personal computers in this area, when the entire population is considered, the enthusiasm and interest in computers is still largely confined to technical people and hobbyists; and I daresay that it is common to find the same stratification of interest even within the families of these enthusiasts.

The population in need of increased education about computers is far from homogeneous. It contains future or present decision-makers who need to be more clear about the appropriate roles of computers in business and other endeavors. There are bill-payers to whom more computer knowledge might provide an alternative to "blaming the computer." (They might learn to blame the computer people!)

And there are employees who note the increasing use of computers around them at work, and who need a better understanding of where they stand vis-a-vis these new tools.

While it may be worse for one to be ignorant of computers within a technical community like 'Silicon Valley', I think it is safe to say that large segments of the population throughout the country are excluded from computer use and knowledge.

But everyone should have the right of access to information about such an important aspect of the world, and we should be offering an opportunity to gain awareness of this very technical aspect of modern society. Right now that access to computers is severely limited by a person's training and experience. Almost all available courses in programming require advanced math or are taught at a fast pace. Non-technical training in the uses of computers is finding its way into the schools, but not much is being done for adults.

The goal of this education is not vocational; although there are employment opportunities, we should not makeunrealistic promises, nor should this be our most important incentive: not only is there more to getting a job in the industry than just knowledge about computers, but also we must emphasize the importance of knowledge about computers for those who do not choose to work in the field. We owe it to the bystanders in the computer revolution to provide a chance to learn about computers in a way which is appropriate to their interest, expereince, and aspirations. Think of it as a liberal education.

Finally, it is important to the computer culture itself that the population of users be expanded. I do not mean that increase sales will somehow benefit us all. I mean that outsiders bring new ideas, new applications, and often test our long held beliefs about the way things should be. Any field can benefit from the introductionof new blood, but the world of computer users has been isolated so long from the general public that it will especially benefit.

## WHAT KNOWLEDGE ABOUT COMPUTERS?

The following list is based upon observations of the misconceptions which even educated people have about computers, upon estimates of what they need to know to deal intelligently with computers around them, and upon hunches about what makes them so angry and fearful about the machines. First, it is important to know what kinds of things computers can do best: what jobs they are particularly suited for, and which ones would be more appropriately done by other means. This leads to how computers are being used now, the prospects for the future, and finally to a discussion of the benefits and problems associated with their use.

We should also introduce certain key concepts at the right level of sophistication of the audience, e.g., input, output, storage, program, looping, branching, etc. Some vocabulary and jargon is also useful to pass on: byte, timesharing, mini, micro, chip, etc. Interweaved with all these topics should be the question, "How do you make a computer do something?" Programming, I will argue, should be central to instruction about computers.

## THE IMPORTANCE OF PROGRAMMING

The program is the single most important concept in understanding how computers might be used. When a person is taught the elementary concepts of programming and given thechance to write programs, he or she becomes truly "computer literate" in the sense of being able to produce something in the language, as suggested by Arthur Luehrmann of the Lawrence Hall of Science (Luehrmann, 1979). Furthermore, through programming a person is given a clearer picture of the capabilities of the computer, is given a chance to be in control of the computer, and is given experience with the actual operation of a computer system. Each of these will be discussed in detail below.

Capabilities of the Computer. It is difficult for even a sophisticated lay person to conceive of the kinds of problems which computers can and cannot accomplish or are most suited for. Further, I find

it very difficult to explain these distinctions without reference to the language used to give instructions to the machine. The commands of the language provide a vocabulary for discussing these matters. The notions of input and output are dealt with immediately in the very first programs people write; the range of the prossible calculations may be intraoduced early and at the appropriate level of sophistications with PRINT statements; the significance of stored data comes early with the READ statement; the notion of looping is explained easily if GOTO, FOR...NEXT, and IF...THEN have been encountered. And the latter command introduces comparison and branching.

An understanding of these few concepts - input, output, calculations, stored data, looping, comparisons, and branching - is necessary before one can appreciate the special role computers can play. Thus the structure of the language quickly reveals why the computer is better at searching long lists for a particular name than it is at understanding a simple English sentence; why it is more suitable for calculating the cost of housing materials than for designing the house; and why it is better used as a word frequency counter in Shakespeare's Sonnets than as a critic of their quality, or as a writer of poetry.

Once these concepts are introduced through writing elementary programs and studying others, the instruction may leave programming to concentrate on other goals. But programming provides the framework for understanding the capabilities and limitations of the machines.

Controlling the Computer. We live in a world in which computers seem to be in charge: errors are hard to correct, programs are difficult to change, machines impossible to question. Non-technical people particularly feel the disadvantage of not being able to cope with the computer and computer people. They appreciate a demsonstration that a computer is a lifeless piece of junk before it is plugged in or programmed, and that any difficulty we have in communicating with it is related to its own limited vocabulary. But if one learns a bit of its language, what patterns can appear on the screen and what calculations are possible!. To provide people with a little vocabulary with which to give instructions to the computer, and thereby control it, is to let them for once be in charge of the machine. Non-technical people particularly enjoy this opportunity.

How Does it Work? Programming requires that a person interact intelligently with the various components of a computer system. Through the sequence of writing, saving, loading, changing, and replacing programs on disk or tape, a person gains experience which can make it easier to understand the concepts of ROM, RAM, disk storage, and the various degrees of permanence of each. And since programming even the smallest microcomputer requires steps which are analogous to those on the largest machines, the student gains insight into computers which seem quite different from the particular one used for instruction.

## A HIGH SCHOOL COURSE IN PROGRAMMING

For three years at Mountain View High School the computer programming class has been open to all students, there being no mathematics or grade level requirement. I have described this class in detail elsewhere (Wagner, 1979). Briefly, the equipment consists of 7 microcomputer terminals: two Sols and two Horizons running Northstar BASIC (one Horizon timeshares three ports), and a TRS-80 with disk drive. The class has been completed by students at all levels from remedial math through calculus, including a few from special education.

The textbook (Golden, 1975) is the only available one, in my opinion, which is suited to classroom instruction in programming, but even it assumes a knowledge of algebra. Therefore the class is actually run on different levels: the basic assignments are usually divided into two levels of difficulty, with opportunity for students to choose from the wide range available in the text. Studetns write approximately 25 programs during the semester and gain familiarity with loading and saving onto floppy disk, and accessing the printer.

The class is primarily about programming, although

it is supplemented by a movie, a videotape of the 1978 NOVA production on computers, and informal discussions of the uses and abuses of computers. The programming covers through FOR...NEXT loops, plus disk commands. Students may retake the class for credit. A few choose to repeat the basic assignments, others work on projects which we agree upon, and some assist in the teaching of the class.

Since many students have difficulty scheduling an elective offered only twice a year, the computer offerings at the high school are supplemented in two ways. Informally at lunch and after school students at all levels use the computers for games, programming, or typing in programs from magazines or books. We also introduce students in some math classes to programming in a 1 or 2 week short course which is described by Wagner (1979).

## A COURSE FOR ADULTS

During the summer of 1979 I was invited by the Cupertino Union School District to give a non-technical introduction about computers to the district's central office staff. This was truly an enlightened decision, for while the district is a leader in instructional uses of computers and is planning extensive improvements in its data processing program, many employees who are involved only tangentially in these projects or whose jobs might be directly affected by the changes, knew little about computers. Such a situation could make these employees feel neglected, by-passed, or threatened.

I think that the three weeks (6 two-hour sessions) helped alleviate some fears, made the employees feel good about their employer, and made them better able to adjust to increase computer use around them. Over 40 people participated in the two sections of the course, which was offered through De Anza College at no cost to the school district. They were a truly diverse group - secretaries, maintenance men, clerks, bus drivers, gardeners, etc. - and they were a wonderful class. They were much less inhibited than professionals (particularly teachers) when first approaching the keyboard. They always had interesting questions and seemed to really appreciate the instruction.

In accordance with my belief in the importance of programming in such a class, about half the time was spent learning some elementary BASIC commands and studying and writing programs. Listed below are some of my teaching strategies for this class.

## TEACHING STRATEGIES

Questionnaire. In the first session they responded to a questionnaire wich asked for opinions about computer use - what are the good and bad aspencts, how might it affect your job, what would you like to learn, etc. This gave people a chance to express themselves and brought many more points to light for later discussion than a direct verbal discussion on the first day would have.

Access to Computers. There were 10 computers avaiable - Apples, PETS, and TRS-80's - each connected to a Nestar Disk Sharing system (reviewed by Stone and Wagner, 1979). This system provides quick access, through a menu program, to many programs and made for a comfortable introduction to the keyboard, to input/output, and to the range of things possible. The disk sharing system saves a lot of instructor time which often must be spent preparing cassettes or disks.

Blackjack: The BLACKJACK program on the TRS-80 created a lot of interest with this group and I used it to demonstrate non-perfection and changeability of a program. That program, like many others, allows the player to generate fantastic winnings by betting negative amounts and then losing. (I use this same program to discuss subtraction of negatives in Algebra.) The students were amazed that a bug like that would get such national distribution. I showed them that the program could be modified to reject such bets, and drew two morals from the example: if programs do not work the way you desire, you should complain to someone who can fix it; and contrary to popular belief, programs are not written in stone, but may be changed.

Phone Number Lookup: I used a program called

Phone List which is distributed with the Apple to demonstrate list searching and changing. Each person's name and a fictitious number had been entered into the file and each had a chance to correct the number, change the name if desired, and to add another name and number. This was close to the applications of computers which many of them would experience, and generated a good discussion of how the program might be improved from the user's point of view.

Elementary Programming. The enclosed program (Appendix 1) was used to introduce programming. Each person loaded and ran the program and then it was analyzed in detail with the whole class. Printing out a canned paragraph like this, with personal details embedded in it is another familiar example which adults can appreciate.

The sessions devoted to programming concentrated on PRINT, INPUT, and IF...THEN. A person with a little prior programming experience wrote a number guessing program, but the efforts primarily involved programs which created dialogues between person and computer. These ranged from a few lines asking for name and address and then repeating the given information, to a 40 line program which produced an interactive conversation about the merits of some modern movies. I was particularly proud of a brave attempt at generating poetry in a mad-lib format. Some students were too ambitious and a few were frustrated and a few never did get a program written. All in all, however, it was a very positive experience for them and for me. I feel that my opinion about the importance of teaching about computers to non-technical people was confirmed, as was my belief in the importance of including programming in the class.

REFERENCES

Golden, N., Computer Programming in the BASIC Language. Harcourt Brace Jovanovich, San Francisco 1975.

Luehrmann, A., "Computer Literacy in the Schools, a National Strategy", Proceedings of the 4th West Coast Computer Faire. Computer Faire, Inc., Palo Alto, 1979.

Stone, D., and Wagner, W., "A Review of the Nestar Disk Sharing System." Recreational Computing, September/October, 1979.

Wagner, W., "Microcomputers in the High School, Expanding our Audience." The Computing Teacher, September, 1979.

APPENDIX

The list and run of a program used to introduce students to programming concepts.

```
10 INPUT "NAME PLEASE? "; A$
20 INPUT " DEPT. IN C.U.S.D.?"; B$
30 INPUT " NO. OF YEARS THERE? "; N
40 INPUT " OFFICE PHONE NUMBER? "; C$
50 HOME
55 REM  on the apple, HOME clears the screen
60 PRINT "       THIS IS TO VERIFY THAT"
70 PRINT A$;", AN EMPLYEE IN THE"
80 PRINT B$;" DEPARTMENT OF THE"
90 PRINT "CUPERTINO UNION SCHOOL DISTRICT,"
100 PRINT "HAS WORKED THERE A TOTAL OF "; N
110 PRINT "YEARS.  THIS EMPLOYEE CAN BE"
120 PRINT "REACHED AT TELEPHONE # ";C$;"."
130 PRINT
140 PRINT
150 PRINT
160 INPUT "TYPE 1 TO DO THIS AGAIN AND 2 TO END
    THE PROGRAM";X
140 IF X = 1 THEN GOTO 10
150 END

RUN

NAME PLEASE?  JAMES EARLE CARTER
DEPT. IN C.U.S.D.?  FUEL CONSERVATION
NO. OF YEARS THERE? 3
OFFICE PHONE NUMBER?  PA 6 5000

        THIS IS TO VERIFY THAT
JAMES EARLE CARTER, AN EMPLYEE IN THE
FUEL CONSERVATION DEPARTMENT OF THE
CUPERTINO UNION SCHOOL DISTRICT,
HAS WORKED THERE FOR A TOTAL OF 3
YEARS.  THIS EMPLOEE CAN BE
REACHED AT TELEPHONE # PA 6 5000.

TYPE 1 TO DO THIS AGAIN AND 2 TO END THE PROGRAM 2
```

# INDIVIDUALIZED INSTRUCTION IN COMPUTER PROGRAMMING

Carl Grame and Dan O'Donnell, Instructors
De Anza College
21250 Stevens Creek Boulevard
Cupertino, California 95014

## Introduction

This paper describes how courses in three programming languages are conducted on an individualized basis at De Anza College. The languages are FORTRAN, BASIC, and IBM 370 assembly language. Since FORTRAN is our most completely developed course, we use a detailed description of it as representative of all three courses. The individualized approach, which includes audio tapes, uses a variety of teaching styles and accommodates a variety of student learning styles.

## What is Individualized Instruction in General?

Individulalized instruction is a process based on the premise that a student, given sufficient time and the availability of adequate human and educational resources, can master a given segment of instruction to meet an objective that has been specified in advance. Individualization recognizes that different students learn at different rates and in different ways and that instructors are as varied as the students they teach. In an individualized course the instructor is a facilitator and a resource person, providing supplemental assistance through tutoring sessions, small groups, on-the-spot answers to questions, selection of additional media, and development of innovative learning activities to extend the traditional functions of teaching and administration.

In individualized instruction the learning process becomes a joint venture between student and instructor, based on individual education and career requirements. Together, student and teacher may choose from alternative materials, select units or modules, and establish a schedule and an appropriate pace for working. Students with restricted schedules have freedom to learn without rigidly prescribed classroom attendance. Students who are motivated can move ahead rapidly and independently. Those who are handicapped, physically or educationally, are given adequate time and one-to-one help. The individualized approach enables the institution to experiment with open registration in which students enter or complete the course at any point during the term.

## Why Individualized Instruction?

When using the lecture approach to teach computer programming at De Anza, we noted a need for students to progress at various paces. We established an individualized instruction system as an integral part of an audio-tutorial class in FORTRAN and Beginning Assembly Language since 1970, and BASIC since 1978, and experienced excellent results. Student response has been very positive because they have been able to devote more time to writing computer programs and consulting with the instructor on an individual basis. We have come to the conclusion that any motivated student who at some time has passed a course in elementary algebra should be able to satisfactorily achieve the objectives of the FORTRAN and BASIC course. The Assembly Language course has FORTRAN or BASIC as a prerequisite. Of the three courses, FORTRAN is completely developed. We will use it, therefore, to describe the way we conduct all of our individualized courses.

## Course Components of the Individualized Course in FORTRAN

Computer Programming: An Individualized Course in FORTRAN IV is an audio-tutorial program divided into twelve units, plus a supplement entitled, Using a Keypunch. The course consists of the following components:

Tapes. Each audio unit explains an aspect of computer programming and guides the student through the corresponding unit of the Learning Guide. Instead of a formal lecture, the tapes present a dialog between two instructors, using alternating voices to sustain the listener's interest. The narrators explain the illustrations in the Learning Guide and occasionally instruct the student to make entries in illustrations that are incomplete. The tapes also provide feedback so that the student will know whether the entries are correct.

- Orientation tape introduces the student to various features of the Learning Guide and enables the student to become acquainted with the audio listening procedure before the FORTRAN instruction begins.
- Audio units are coordinated with the twelve units of the Learning Guide.
- Supplementary tape is used to introduce students to the keypunch or to review keypunching if necessary.

Learning Guide. The Learning Guide is the text workbook for the course. In addition to the illustrations that accompany the tapes, it includes the following:

- Objectives provide a clear statement of what the student should accomplish in each unit.
- Introduction explains the purpose of FORTRAN statements and techniques that are covered in the unit.
- Reference section aids the student in carrying out assignments by summarizing major points covered by audio tape.
- Application Exercises give the student an opportunity to apply the material in the unit.
- Answers to Application Exercises can be used for immediate feedback for the student's self-appraisal.
- Programming Problems provide the student with an opportunity to write executable programs.
- Quick Guide (the appendix) is an additional reference for FORTRAN IV statements, rules, and summaries.
- Index provides an alphabetical reference to major topics and the number of the unit in which each topic is first explained.

Scriptbook. At the instructor's discretion, Scriptbooks can be made available for the convenience of students who prefer reading to listening, for review, or for use when audio listening facilities are not available.

Instructor's Guide. To assist the instructor, the following features are included in this Instructor's Guide:

- Objectives are listed to give the instructor an overview of the course and criteria for judging students' progress.
- Flexible Course Calendars provide suggested time schedules for two-week, six-week, twelve-week, and sixteen-week sessions.
- Suggestions for Grading offer several measurements as alternatives to be used according to the instructor's preference.

- Bibliography lists selected FORTRAN textbooks for the instructor who wishes to provide additional resource material.
- Learning Guide illustrations are reproduced to show the entries that must be made by students.
- Solutions to Programming Problems are computer-produced. Both the program and the results are provided.
- Test Answers are included for each of the twenty-four tests.
- Reproducibles include a blank coding form and a flowcharting worksheet, which may be duplicated for the convenience of the instructor.

Testbook. The Testbook includes two alternate tests for each unit. Test questions are keyed to the objectives of the unit and are compatible with the Application Exercises and Programming Problems. The two test forms can be used for pre-testing and post-testing--or for testing and retesting to give the student a second chance to pass the test. In any school that adopts this course, tests may be reproduced without further written consent from the publisher.

Additional Course Resources

- Access to an instructor, a proctor, a tutor, or a laboratory aide
- User's guide for computer facility
- Manufacturer's FORTRAN manual
- Programming supplies, including cards, flowchart forms, coding sheets
- Access to a computer and keypunch or time-share terminal
- Audio-tape listening facilities
- Supplementary FORTRAN texts

Student Learning Activities

The twelve audio-tutorial units in the Learning Guide are presented in sequence, followed by the supplement entitled, "Using the Keypunch." The student follows the steps described below.

Get the appropriate tape for the unit to be studied. Before starting the tape, the student reads the objectives and the introduction to the unit in the Learning Guide. Then, while listening to the tape, the student refers to the designated figures in the Learning Guide. From time to time, the student is asked to make entries in the Learning Guide, and also to take notes. The student should stop the tape as often as he or she wishes in order to replay any portion needed for review.

After completing the audio portion of the unit, the student reviews the objectives, then completes the Application Exercise, and checks the results with the answers provided in the Learning Guide. Programming Problems begin in Unit 3. Before doing these problems, it may be necessary for the student to read the computer manufacturer's reference manual for any special details related to the computer installation being used. The student writes and executes the Programming Problems, and then checks the results with output provided with the problems in the Learning Guide. The student is encouraged to consult with the instructor as often as necessary. The Quick Guide in the Learning Guide provides a handy reference for the definitions, charts, and tables mentioned in the audio tapes and for all FORTRAN statements covered in the Learning Guide.

Finally, the student takes the test for the unit.

In addition to these activities, the student has the option of attending group sessions as needed. These sessions include test previews, and reviews and lectures on exercises and programs.

Student learning activities may be summarized as follows:

- Read the objectives and the introduction to the unit in the Learning Guide.
- Listen to the audio tape and refer to the Learning Guide; take notes, make entries in the Learning Guide when directed to do so; stop the tapes and replay them as necessary; consult the instructor when the need occurs.
- Complete the Application Exercises; refer to the Reference section of the Learning Guide as necessary; check results with answers provided in the Learning Guide.
- Refer to supplementary resources if desired.
- Write and execute Programming Problems and check results with output provided.
- Take tests as directed by the instructor.
- Consult the instructor and/or supplementary resources and review the unit if necessary.
- Attend optional group sessions other than tests as necessary (test previews, reviews, and lectures).

Advantages to the Student

Perhaps the most important advantage of audio-tutorial learning is that the student sets his or her own pace because he or she controls the tape player. The student can replay the tape, or any part of it, as often as desired or stop it at any point to review, to consult with instructor, or to use additional resources. The course offers several additional advantages. The objectives give a clear statement of the performance that is expected for each unit. While listening to the tape, the student is actively involved in studying illustrations or writing entries in this Learning Guide, and can concentrate without dividing his or her attention between the Learning Guide and a separate textbook. Immediate feedback is another advantage. The tape provides feedback whenever an entry is made in the Learning Guide during the audio portion of the unit. Answers to Application Exercises are available in the Learning Guide. This course also provides practical experience through the Programming Problems, with feedback coming directly from the computer.

Scriptbooks can be made available for the convenience of students who prefer reading to listening, for review, or for use when audio listening facilities are not available. Such alternatives increase the individualization of instruction.

Responsibility of the Student

Individualized instruction permits the student to make many choices. Because these choices must be made, individualized learning places more responsibility upon the learners. They pace themselves and set some of their own deadlines. They check their own progress and decide whether they need to repeat material, go on to new material, or consult with their instructor to meet the objectives of the course.

Students may or may not meet regularly in the group sessions since the audio-tutorial approach makes lectures optional and instructors have more time to devote to interaction with students on a one-to-one basis. Since students set the pace, the instructor may not always know what point they have reached in the Learning Guide or what problems they may be having. Therefore, they must ask for help when they need it.

Instructor Activities

The instructor provides administrative information such as assignments, test dates, grading policies, laboratory procedures, etc., and is available for

consultation with students on a scheduled basis--such as, laboratory sessions, office hours, etc., and conducts tests.

In addition, the instructor conducts optional group sessions which include test previews, test reviews and lectures on application exercises and programming problems.

Ways Student is Assisted

Orientation At the beginning of the course, the student is given a thorough orientation which includes:

- Orientation tape to introduce the features of the Learning Guide.
- Detailed information sheet.

Minimum Pace The student can choose when to read, listen to audio tape, complete application exercises, and work on programming problems or get help from the instructor, lab assistant or tutor. Our approach has a minimum pace with a due date for tests and programming problems. In a regular twelve-week quarter, for example, a student must complete one unit per week by submitting the programming problems and taking the test. Students may go as fast as they want, but not as slow as they want.

Because of the great deal of freedom afforded in carrying out course activities, we have found some students procrastinate until it is too late to satisfactorily complete the course. We feel the weekly due date helps to lessen this tendency by providing an incentive to act.

People The student can call upon the following people for assistance:

- Instructors: during classroom sessions, laboratory sessions or office hours.
- Lab assistant: during open laboratory hours.
- Tutors: individual appointments, drop-in hours or in class.
- Facilities for handicapped: in-class note takers for blind, deaf, or quadriplegic students and in-class sign language interpreter for deaf students.

Flexible Arrangements The student has considerable flexibility in completing course requirements.

- Open laboratory. A lab assistant is on duty to help students during the twelve-week quarter from 7:30 a.m. to 11 p.m. weekdays, and 9 a.m. to 3 p.m. on Saturday.
- Classroom laboratory. An instructor is on duty to help students during the twelve-week quarter from 7:30 a.m. to 2:30 p.m. on Mondays and Fridays.
- Audio Tape Listening. The student has the following options for listening to audio tape cassettes:
  - Listening room in Learning Center (library) from 7:30 a.m. to 10 p.m. weekdays, and 1 p.m. to 5 p.m. on Sundays.
  - Obtaining copies for personal tape player.
- Script. Scripts for audio tapes are available at the reserve desk in the Learning Center.
- Tests. Students working ahead of the minimum pace may schedule tests in Learning Center.
- Tutors. Students may avail themselves of tutorial help in a variety of ways:
  - Individual meetings with appointed tutor at a mutually agreeable time.
  - Drop-in tutor is on duty in Learning Center during scheduled hours.
  - In-class tutors are available during two- and six-week summer sessions.

Group Sessions. The student may attend any, all, or part of the following scheduled group sessions for each unit:

- Lectures on general subject, application exercises, and programming problems.
- Test previews and test reviews.
- Tape playing in clsss during two- and six-week summer sessions.
- Tests.

Bonuses and Penalties. We employ the following forms of bonuses and penalties which affect a student's score which is based on a total possible points of 200:

- The lowest test score is dropped.
- Additional points are given to those attending test reviews who have scores in the lower ranges.
- One point is deducted if a program is not turned in on time or if it is incorrect.
- Students may correct and re-submit programs; however, the course letter grade is lowered if the record shows there is more than one incorrect program at the conclusion of the course.

Status Reports. Student test scores and programming problems status are posted for each unit.

Miscellaneous Observations

Student Acceptance. We have conducted surveys of previous classes near the end of the course and find that generally students like this style of class and would like to take other subjects in a similar format.

Variety of Students. Many different learning styles or needs can be accommodated.

- Slow learners, subject to minimum pace, may replay tapes (or read script) as many times as necessary.
- Fast learners can finish course before the end of the term.
- Students who prefer listening can emphasize use of audio tape.
- Students who prefer reading can emphasize use of script.

Organization Required. To make the independent feature of the course effective, it must be very well organized. If you are developing your own materials, it takes a considerable amount of time. We are still working on the BASIC and Assembly Language course. Once organized, however, it runs rather smoothly.

Teacher Flexibility. We have found it possible to easily exchange coverage of classes. For example, during a twelve week quarter, we are assigned seven class sections of two or three languages between us that meet between 7:30 and 2:30. Two particular features of our approach are:

- Team teaching--To give students exposure to more than one instructor, we schedule ourselves so we each cover all sections sometime during the week. Also, if one instructor needs a block of time free to work on a project or attend a conference, the other can cover.
- Blocked room schedule--We have one room for all seven sections. Students are free to come in whenever an activity is taking place that would benefit them even though it may not be their officially assigned section.

Variety of Course Calendars. We have been able to easily teach the course on a twelve week, six week, and two week basis.

Planned Improvements. We are considering the following improvements as time permits:

- Complete BASIC and Assembly Language Learning Guides.
- Develop more versions of tests for all classes.
- Explore open entry/open exit.
- Explore flexible credit.

Summary

Based on student performance and acceptance, we believe the individualized approach has been quite successful in enabling a variety of students to learn computer programming. Our experience indicates that student success is directly related to the extent that

1. The course is well organized, including the availability of as many options and human and material resources as possible.
2. The student realizes and accepts personal responsibility to utilize as many of the various options and resources as necessary for his or her individual needs.

If we can be of further assistance, please contact us.

# YOU'D LIKE TO TEACH THE WORLD TO WHAT?
## A Guide to Writing Micro-Computer Courseware

Silas S. Warner
MUSE
331 N. Charles
Baltimore, Maryland 21201

ABSTRACT

The use of micro-computers in education is bringing computer-based education within reach of the average homeowner. A guide is presented for writing educational programs, testing them and making them available to students.

## WHAT IS COURSEWARE?

When a salesman talks about selling a home computer, one of the magic words mentioned is "education." In the near future, home computers are supposed to become total information centers, combining classroom, library and instructor in one portable unit. That hasn't happened. Educational programming for home computers remains in the "game" stage. Only a few full-scale curricula have been developed, and those mainly in micro-computer programming. The revolution in education is still around the same corner it's been for three years.

You can help push it around the corner. Everybody here knows something different, that it would be fun for others to learn. In many cities, there are now "education exchanges" where you can volunteer to teach a course or go to take one. Most people, though, don't want to take the time to teach. The computer can help you teach by making your knowledge available to anyone who wants it, at any time.

But don't expect the computer to do all the work. You must be able to prepare a program which will tell the computer how to teach, as well as what to teach. Unlike most computer programs, a teaching program should not be put together for the computer. The ultimate goal of a teaching program is the student. For that and other reasons, a special term is used by educators to denote teaching programs. They're called "courseware".

Courseware can be prepared for other teaching methods as well as computers. Films, audio tapes, even testbooks are courseware. Why computer courseware, then? What distinguishes computer teaching from other learning aids? I'll answer that by considering first what the computer is not.

## WHAT THE COMPUTER IS NOT

1. The Computer Is Not a Textbook.

As a courseware developer, I have had many requests along these lines: "I have written (or found) a great textbook in -------. If I put that book on the computer, what will I have to do to avoid copyright problems?"

To which my answer has always been, "If you want to put a textbook on a computer, put it on top of the keyset and tell the students to read it. It's cheaper that way, and the student can take the book home."

One of the worst things that a computer can do is display long passages from a book. It essentially wastes the power of the computer. There is no reason why a testbook can't be packaged as part of a computer-based course, and the computer assign "homework" in the book. This approach is called computer-managed instruction (CMI).

2. The Computer is Not a Movie.

Some computer-based education systems make a great fuss about the power and scope of their graphics terminals. Now a graphic display is almost a must for education - some concepts just can't get across without a picture. But if all your lessons consist of fancy graphics, you can do it cheaper and more effectively with film or video tape.

3. The Computer is Not an Instructor.

For all the talk about artificial intelligence, remember that the most complex computer whose existence is publicly known has a total complexity in terms of number of units, about equal to the brain of a frog. There isn't one computer system on the market, for instance, that can completely analyze a sentence for its meaning and construct an appropriate reply.

When a teacher asks, "What is the capital of Nigeria?" and the student responds, "Can I go to the bathroom?", or "Betty hit me with a spitball!", the teacher can reply intelligently, It will be a long time - at least 30 years - before a computer will be able to do that at a low cost.

## WHAT THE COMPUTER IS

For educational putposes, the definition of a computer that I am going to use is this:

"A device capable of presenting a predefined program of educational materials to one or more students, and varying that program as the student interacts with it."

This definition is not complete. Most large batch computers can't present educational material, or interact with a student. But you can't develop courseware for them.

And there are "teaching machines", based on microfilm, video, and audio tapes, which don't have computers and yet fit the definition. But writing courseware for these machines involves the same decisions as writing courseware for computers.

The operative words in this definition are "vary" and "interact." These two ideas - varying the presentation of the course, sometimes called "branching", and interaction between student and computer - distinguished computer courseware. The computer, unlike any other teaching aid, can decide what portion of its store of knowledge to present to the student next. It can decide, that is if you provide it with the choices and routes through the material in your courseware design.

WHAT ARE YOU GOING TO TEACH?

Most people assume that you start programming when you sit down in front of the computer. In educational programming, especially, nothing could be father from the truth. Your first step is to decide exactly what subject matter you want to teach. Focus on the student, not on yourself or the computer. "After this lesson, the student should be able to....". That is the way you should phrase your goals. Educators call this an "instructional objective." It is a statement of what the student should be able to do after completing the lesson.

Notice that you do not define what the student should know. The computer has no way of finding out what the student knows, and neither do you unless you're a mind reader. So you specify a way for the student to prove knowledge to you or the computer. This is the basis of an instructional objective.

Usually, the student will need to learn the fundamentals before going on to more advanced topics. You should decide on what the student should learn first - in educational jargon, an "objective hierarchy." Decide on the order in which you will teach your material, and write it down before starting.

HOW ARE YOU GOING TO TEACH?

That's a good question. Educators have been debating that question for their entire existence - roughly 3,000 years - and have come up with no right answer. Teachers generally agree today that they have come up with the right answer - that there is no one right way to teach. There are any number of sometimes right ways, and the right way for you depends on what you teach and how you and your students feel about it. There are a few tested forms for computer courseware, though. I will describe some of them.

The Tutorial - the computer presents text to the student, possibly with graphic illustrations. At frequent intervals, the computer asks questions. If the student persistently misses a questions, the computer goes back over the material again. The trick to a tutorial is to make the text interesting enough to keep the student's attention - the form, by itself, is boring.

The Drill and Practice - the most common form of micro-computer courseware on the market, because it's the easiest to program. The computer asks question after question, which the student answers while the computer keeps score. It fills a classroom need - no teacher likes to administer drills - but it's deadly dull for everybody except the computer.

The Simulation/Tutorial - first, the computer teaches a principle by rote. Then, it presents a simulation of a real situation to which the principle applies. The student can change the parameters of the simulation, and watch the principle take effect. This is one of the most effective teaching methods on the computer. The most highly praised educational lessons on every computer system I know are simulations. But a simulation is very difficult to program.

The Discovery Method - also involves a simulation. The student is not told the rules of operation, but allowed to discover them by operating the simulation. This is one of the most controversial of computer teaching methods. When it works, it far outshines any other method in both teaching effectiveness and fun. But measuring learning by discovery isn't easy. The student may have an intuitive grasp of the principle, and can run the simulation perfectly, but not be able to put that knowledge into words.

HOW TO RUIN COURSEWARE

No matter what teaching method you use, Murphy's Law applies to programming your courseware. It's a lot easier to produce a mess of traps and problems for your student than it is to produce a smoothly functioning lesson. This holds for any computer program. But you are writing courseware for students, not for computers. There are whole new classes of errors that you can make, and not be aware of until you test your lessons with real students. For instance,

Requiring too much typing. We computer people tend to forget that 90% of the world out there can't type more than 10 letters in 60 seconds. Force such a person to type, say, "aliphatic hydrocarbon radical," and that is the end of the course for that person! If your system allows it, best avoid typing altogether by using a light pen, touch screen or similar device. And whatever you do, avoid requiring exact spelling of any word longer than "four"!

Unnatural key coventions. We have all seen the BASIC kludge, which requires you to "TYPE 1 FOR YES, Ø FOR NO." Why should the student have to do the work of converting his answer into computer form? Why should he not just be able to type "YES" or "NO"? Using single keys to get out of typing is a coward's way out unless you keep the key conventions displayed constantly so that the student doesn't have to remember them. And even then, it's ugly.

Forgetting unexpected answers. A parable: a teacher asks a student, "How many pecks in a bushel?" The student replies, "I don't know," and gets the reply "***SYNTAX ERROR-RETYPE LINE***." Moral: don't let your programming language decide what happens to unexpected answers. Closely related to this is:

Not knowing when to quit. After a specified number of wrong answers, no computer lesson should just keep saying "TRY AGAIN" forever! Every lesson should either give the student the answer, or pack it in and go on to something else.

Recycling forever. Most computer programmers know of the dangers of an infinite loop. But the dangers are multiplied when a student is trapped inside. You can never predict the interaction between computer and student - a student may come out of your carefully designed instructional sequence firmly convinced of the wrong answer. If the computer cycles back over the same material over and over, the student is just going to get more frustrated each time. Have a way for the computer to give up or call for help if this happens.

## NICE TOUCHES

Here are some good things that you can do to make your courseware better.

Letting the student know. Wherever possible, you should show the student scores, precent completed and any other relevant student data. Some of the most effective courseware I've seen has a built-in map. At intervals, the student sees the map of the lesson, with "YOU ARE HERE*" in big, bold letters!

Varying your formats. Some lessons I have seen slap a whole page of text on the screen, wait for the RETURN key, slap up another whole page and so on. This leads to headaches, eyestrain and boredom. If you must show large quantities of text, change its shape, size and position from page to page. This also gives the student a sense of place - he can recognize a given page by sight.

Surprises. It can be a fatiguing business, sitting in front of the computer for hours on end - well, maybe not for us computer freaks, but certainly for our students. So you can throw something totally irrelevant into the middle of your lesson, just to wake the student up. Don't overdo it - once every 20 minutes is plenty - but if Haydn can do it, why can't you?

## PROGRAMMING YOUR LESSON

One question comes up frequently about programming courseware. "Do I have to use an educational language to do it?" My answer is, no but it sure helps. Most computer languages are designed for the convenience of computer programmers. They assume that someone operating the system at least knows something about the computer. As the examples of bad courseware practices above indicated, this can lead to real trouble when a complete novice sits down in front of the machine.

Of course, you can program around the limitations of your language. I have seen excellent courseware done in BASIC and FORTRAN, and I suppose good courseware could be programmed in assembler or even COBOL. But why bother with tricks to get around the quirks of your language, when people have already created software designed without quirks? Furthermore, most educational languages are partially or fully transferrable. That means that what you write on one computer, you can usually reprogram quickly for another, even if the language used by the other is different.

I won't say much about the actual programming process, because there are two schools of thought about it. The prevailing school, used by most large corporate training groups and some universities, is that all courseware should be written down on paper before it is programmed into the computer. The script, they say, will show up weaknesses in the lesson design before computer time is wasted on programming.

I feel that when you are doing a project on your own, writing every step down on paper is a waste of time. As users of word-processing equipment know, it is usually much easier to change a file on a well-designed computer system than it is on paper. Of course, as soon as the lesson runs even partially, you should make a backup copy so that if you accidently destroy your working copy, you still have something to work from. And when you are done programming, it is a good idea to print a copy of your lesson for reference as you test it.

## TESTING

Once you've debugged a piece of software, it's usually ready to use. Not courseware! Courseware isn't written for computers but for students. The next step after your program is "finished" is to test it with students.

The first test student, of course, will be yourself. You won't be a very good test.

You can test your lesson by very careful reading, for spelling and typing errors. You can test it for programming bugs. But you can't test its teaching ability, because you already know what it teaches.

To really test a course, you must use people who have never seen it before. These may be hard to find - your family, co-workers, fellow computer club members, whatever - but they will find problems in places you never thought to test. The courseware testing corollary of Murphy's Law is, "The things that seem clearest to you will be the things the student cannot understand."

If I were to recommend the ideal population for testing courseware, I would have to recommend junior-high and high-school students. Person for person, they are the most destructive force against computer programs that I know. Their ingenuity is matched only by their drive to test the limits of their world - in other words, to break your courseware. And high-school students usually know enough about computers today to recognize a goof and exploit it to its limits.

Here is how to set up your test. Get your student's time committed for at least four times the length of time it takes you to complete the course. The student will take at least twice the time you will. Set the student down in front of the computer. Start the course. Then sit down behind the student and SHUT UP!

The pressure will be immense to tell the student just where he's going wrong. Don't give in unless he's hopelessly snarled, even if he turns to you and asks. To make sure, tape-record the whole session, murmuring your remarks into the microphone. When the test is over, then you can interview the student and get his comments and reactions.

The information you get from testing almost surely means that you will have to rewrite some sections of your course. Then you will have to retest it - with a different student. The first student will know what to expect.

## OUT INTO THE WORLD

It takes time, of course. But eventually your test students will have nothing but extravagant praise for your course. Or to be realistic, at least they won't constantly get stuck and give up in disgust. What will you do then? (After singing Hallelujah, of course.)

If you are a teacher or educator, the road to publication is wide open for you. But for most of us, it's difficult at least to get something accepted in education. Fortunately, there are ways around the school system in most localities.

Your local computer club can be a big help. Most computer clubs have software exchanges, in which you can put programs and have them swapped. That will take care of your computer-owning neighbors. In addition, your club can give exhibits and demonstrations. Many people have heard about computer-based education. This can be your club's chance to show it in action.

Another outlet for your courseware that you probably haven't thought about is the local library. More and more libraries are getting small computers. Many home computer companies, such as Apple, will actually give grants to libraries for this purpose. The problem has been to convince library administrators that small computers are good for more than bookkeeping and games. Your courseware may be just the argument they need!

A new development in many areas of the country is the "educational exchange" that I spoke about earlier. Together with the alternative school, the educational exchange is, part of an exciting new network of learning units outside the traditional school. They're looking for new ideas, and your computer courseware might fit perfectly into their plans.

However, don't go to the local school system. The range of obstacles raised against any innovation in the traditional schools would make the Maginot Line look like a sand castle. Everyone - from teachers' unions to staff psychologists - will be mobilized against you and your courseware. I'm afraid the only way to move the schools off of this position will be through convincing demonstrations.

All this assumes that you just want to see your courseware used, and people learning from it. If you have something extra-special, though, you may want to profit from it by publishing your courseware.

Today, the courseware business is chaotic and disorganized. If I were to give one piece of advice to you, it would be; Don't make the business any more chaotic by trying to market the courseware yourself. You can't possibly compete with software sources marketing thousands a year for the attention of the public.

Of course, I can give more bits of advice to you. Some of your best allies are the people that sell other materials incidental to your courseware. The computer manufacturer, to start with. The vendors of your software system, for another. If your course is about some specific piece of equipment, write the manufacturers of the equipment. Computer-based education today is a hot subject. If you have a teaching tool that works, the world needs it!

# ALPHABETICAL VERSUS GRAPHOTACTIC CRT PAGE LAYOUT OF LETTERS FOR A VERSATILE PORTABLE SPEECH PROSTHESIS (VPSP)

Carol A. Simpson, Psycholinguistic Research Associates
2055 Sterling Avenue, Menlo Park Ca 94025

*Abstract*

Results of a project to develop a Versatile, Portable, Speech Prosthesis (VPSP) are presented. The project is the offspring of special interface technology for the motor-handicapped developed at the Rehabilitation Engineering Center at Childrens Hospital at Stanford, discussed by Maurice LeBlanc in 1976 [4] and of linguistic human factors research on speech synthesis for airline applications conducted for the National Aeronautics and Space Administration (NASA) Ames Research Center by Carol Simpson between 1974 and 1978 [6]. The system is designed to aid the severely motor-impaired, wheelchair-confined cerebral palsey population. The main constraints under which the system was designed are:
1) Wheelchair mountability
2) Power entirely from wheelchair batteries
3) User control interface logic assuming only 1 user movement.
The system consists of a Z80 CPU with S-100 bus, 16K RAM, a mini-floppy disk, a 5" CRT, a VOTRAX (Reg.TN) ML-1 (Reg.TN) phoneme speech synthesizer, and a specially designed CRT and parallel I/O board. The software allows the user to compose messages using items from a pre-coded "starter vocabulary" of words and phrases, words generated by the user from spelled or simplified phoneme entry, and/or items previously generated by the user and stored for future use.

Results are presented of a linguistic human factors study of alternnative layouts for CRT menu pages. Application of the results to the design of single-switch communication aids is discussed.

*Introduction*

The Versatile Portable Speech Prosthesis (VPSP) is an on-going project to develop a wheelchair-portable speech synthesis system capable of unlimited vocabulary and message construction and designed to simplify message construction for the user. This simplification was achieved via two methodologies. Linguistic analyses of language structure were used so as to limit the number of items the user must chose from at any point in the message construction process. Limiting list size will reduce search time for humans as well as computers. Additionally, a single switch (1 bit) user input requires that the system automatically present the user with successive alternatives until the user uses the switch to say "yes" to one of the alteernatives. The fewer alternatives, the faster, on the average, the system will arrive at the item the user wants. To this end, rules of syntactic and graphophonotactic constraints on choices for selections were incorporated into the system logic. Linguistic human factors experiments were also conducted to determine which of several alternative design principles produce the fastest visual search times for words [7] and for letters used in message construction.

*Visual Search Time Experiment*

One design aspect of the VPSP is the layout of letters on the CRT screen during sentence construction. To the extent that this layout matches or conflicts with the user's cognitive map [5] of the spelling of words in his/her language, the page layout would be expected to facilitate or hinder the user's ability to visually locate letters in a menu displayed on a CRT page.

A major goal of the VPSP project is to reduce message construction time. Reduction of the time required for the user to visually locate letters on the CRT display should further this goal. Parenthetically, another type of time reduction involves system search time for an item requested by the user. And still another type of time reduction has to do with the time used by the system in a scanning mode to display successive menu items to the user. This experiment was designed to investigate only the first of these — visual search time. Specifically this experiment was designed to assess different page layout patterns and ordering schemes for their effects on the time required by the user to visually locate in correct sequence of letters to spell a desired word.

*Approach.* It was hypothesized that a page layout pattern which corresponds to grapho-phonotactic structure will result in faster visual search times than an alphabetical block page layout having all currently possible menu items listed in a single alphabetical block. This hypothesis was based in part on the earlier finding that words laid out in a syntactic pattern produced faster visual search times for sentence construction than did words laid out in a single alphabetical block [7]. Names for these two patterns, coined for this experiment, are Graphotactic and Alphabetical, respectively.

Within the major frammework of Graphotactic and Alphabetical layout patterns, several other variables were tested for their effect on visual search time. These are listed below with a short rationalle for each one.

1) letter clusters: certain clusters of letters occur in syllable initial, medial, and final position. The constraints on clusters limit their number to some 50 of the some 8000 in the set of all possible cluster strings. If a user could obtain in one chunk an input string of letters that would otherwise consist of several chunks (letters), word spelling time might be reduced.

2) affixes: certain prefixes and suffixes, such as DIS-, CON- RE- -ING, -TION, -ED, AND LY are used frequently; like letter clusters, they might reduce the number of input chunks needed to spell words and thus improve word spelling time.

3) boxes: when letter clusters are used, it could be helpful to draw boxes around all clusters beginning with the same letter to visually group them.

*Experimental Design.* Eight different page layouts were derived by crossing the 2 possible patterns (Graphotactic and Alphabetical) with the 3 possible aiding variables (Clusters, Affixes, and Boxes) The Appendix contains the Alphabetical, Single Letters, With Affixes Layout and the Graphotactic, Clusters, No Affixes, No Boxes Layout. The others were variations of the basic elements of these two.

In order to limit the number of page layouts, half of the eight possible combinations of layout pattern with the other three variables were not used.

Four of these had single letters with boxes, a non-meaningful use of boxes. Three of the graphotactic layouts combined with the no affixes condition were omitted on the assumption that the effect of affixes would be tested with the alphabetical layout conditions. Finally, the Alphabetical layout x clusters x no affixes x no boxes condition was omitted. This resulted in one less comparison of the effect of boxes. Admittedly, this design precluded testing for effects of interactions among variables, but this sacrifice was made in the interest of performing the study in the time available.

The eight experimental page layouts were presented to each of eight subjects in one of four presentation orders. The presentation orders were designed so as to alternate between page layouts with the Alphabetical pattern and those with the Graphotactic pattern. For two of the orders, a Alphabetical page layout came first. The other two orders started with a Graphotactic page layout. Finally, the ordering schemes were distributed in different positions for thee four different orders. This was considered sufficient control for any transfer effects that might influence tthe results.

*Subjects.* Subjects were 8 normal-speaking adults. Nor-

mal-speaking subjects were chosen in order that the data be representative of users who have a developed cognitive grammar of the language and the writteen language correlates (knowledge of spelling rules, complex syntactic structures). While the VPSP user would not be expected to have a literate knowledge of language or even a language production grammar for spoken language, we assume that with the aid of devices like the VPSP the user will eventually develop these abilities. Tentative evidence supporting this assumption has been reported by Goodenough-Trepagnier 1976 [1] and by Grace and Ashby 1979 [2]. Thus it seemed reasonable to start by designing the VPSP to match the cognitive linguistic grammar of the normal language user. Once a working VPSP existed, data could be collected from actual users for comparison to the data obtained from normal language users.

*Test Material.* A list of 12 words was randomly selected from the 5000 most frequently used words in American English [3] to simulate the type of worrds a VPSP user would be likely to spell. The words are listed below in Table 2.

**TABLE 2**

| | |
|---|---|
| 1. AIRCRAFT | 7. OPTIMAL |
| 2. COVERS | 8. PHRASE |
| 3. DEALER | 9. SOON |
| 4. MOTHER | 10. SUPREME |
| 5. MOVES | 11. TRAVEL |
| 6. NOTE | 12. WORN |

Each word was typed individually on a 3" x 5" (7.62 x 12.70 cm) white, unlined index card. Starting vertical and horizontal position with respect to the upper left corner of the card was constant for all words. The six page layouts were typed on white 8½"y 11" paper. Each page layout sheet was enclosed in a clear plastic cover to protect it from finger marks. The Appendix contains examples of four of the page layouts: Alphabetical x clusters x no affixes x no boxes; Graphotactic x clusters x no affixes x boxes; Alphabetical x single x no affixes x no boxes; and Graphotactic x singles x affixes x no boxes. *Test Equipment.* Test equipment consisted of an electronic stopwatch with digital display to the nearest 0.1 sec., a pencil, and a prepared answer sheet, all for use by the experimenter.

Procedure. Subjects were tested individually, seated at a table beside the experimenter. They were told that the page layouts, not they themselves, were being tested so as to determine which would be best for the VPSP. The experimenter administered the different page layouts in one of the four presentation orders (PO) such that each PO was used for 2 subjects. For each page layout, the subject was asked to study the layout of the letters as long as desired. Meanwhile the experimenter shufflled the experimental test stimulus cards and placed them face down on the table. When the subject was ready, the experimenter turned over the top card so the subject could read it and the subject pointed to each letter on the page layout in the order it appeared in the word. Response times were measured from the time the word card hit the table to the time the subject pointed to the last letter. After all twelve words had been presented for a given page layout, the experimenter asked for comments on that layout. Then the procedure was repeated for each sucessive page layout. The word cards were reshuffled before each new page layout.

*Results*

Table 3 gives the mean times in seconds for each page layout, averaged across subjects and across stimulus words. (See Appendix) For each subject, for each page layout, a mean and standard deviation (using n-1) of the response times to the 12 words was computted.

Anallysis of variance of these data indicated that the differences among means for the 12 page layouts were significant ($F = 6.73$; $df = 7,56$; $p < 0.001$). And a T-test for critical differences among means, where a difference of 1.94 sec. between any two means was needed for the 0.05 confidence level (two-tailed) and a difference of 2.60 sec for the 0.01 level, yielded the four significant differences listed in Table 4. (See Appendix)

In words, the statistically significant differences were that 1) Alphabetical Layout was faster than Graphotactic Layout when effects of the other variables were held constant. 2) For the Alphabetical Layouts, the Single Letters condition was faster than the Clusters condition; and 3) There were no differences due to presence of affixes or boxes, nor for clusters versus single letters for the Graphotactic Layouts.

*Discussion.* The overwhelming superiority of the alphabetical Layouts over the Graphotactic Layouts was a surprise; as was the superiority of single letters over clusters of letters and the lack of effect for affixes. It would seem that unlike syntactic structuree (the grammatical order of letters in a word), graphophonotactic structure (the order of letters and their sound correspondences in spelled letters) is not apparent to the normal language user. Of course for these subjects, alphabetical order was well learned. Possibly given time to learn the positions of letters in a graphotactic layout, users might have performed faster with those layouts. Also the task of directly pointing to desired letters purposely eliminated the confounding factor of serial position and grouping of letters that would have been present with a scanning type of selection procedure. It remained to be seen how fast actual users would be using the scanning selection which makes them wait until the scanner reaches the item they want. In preparation for this possibility the VPSP was designed with both a graphotactic and an alphabetical page layout for spelling words. In accordance with the results, letter clusters were not used, only single letters, and boxes were therefore not applicable. Affixes were not placed on the spelling page. They were however included in a separate page so as to see if actual users would use them.

Initial results, based on the testing of the VPSP by 5 different users, indicate that the Graphotactic Layout is preferred by them after having used it for periods of 4 hours to 5 days. One exception is a user who now uses an alphabetical block layout for an eye position coded communication system. She prefers the Alphabetical Layout. At this time insufficient response time data is available to test the difference in message word spelling time with the two layouts.

One user who has used the VPSP experimentally for some two months regularly uses the affixes and has even requested additional ones. However, the affixes are required far less frequently than are the individual letters. This supports the decisiion to put affixes on a separate page.

*Summary*

The VPSP is a working prototype that is at this time being used to collect data essential to improving the human factors of its design. The present system has demonstrated feasibility. The current phase of the project is a comparative evaluation of alternative design principles with actual users.

*Acknowledgments*

This project has inspirational roots in the thoughts of Kenneth Colby, Director of the Intelligent Speech Prosthesis Project, Neuropsychiatric Institute, ULCA who first suggested the idea of a portable speech prosthesis to Carol Simpson in the fall of 1975 and in the ideas of John Eulenberg, Director of the Artificial Language Laboratory, Department of Computer Science, Michigan State University, who has been a proponent of easier "navigation in language space" since those early days. The initial contact between Maurice LeBlanc, Chief of Rehab Engineering for Chldrens Hosptial at Stanford and the author was coordinated by Charles Kubokawa, Chief of the NASA-Ames Technology Utilization Office. This project might not have received funding without the industry interest expressed by the long term loan for system development of a VOTRAX ML-1 (Reg. TM) speech synthesizer and the virtual donation of a second unit for the wheelchair system from the manufacturer, Vocal Interface Division, Federal Screw Works, Troy, Michigan. Charles Lingel, Independent Consultant, designed the system hardware and software. Thanks to

special software support from Wizzard Systems, we have a very powerful speech system rule development and test package and have also realized significant savings in memory and execution time for the letter-building and speech output driver software for the VPSP. Douglas Williams of Psycho-Linguistic Research Associates made many valuable suggestions for CRT page display design. We thank Peggy Christensen for her help as experimenter for the visual search time study of letter page layouts. This project was funded by a grant to Childrens Hospital at Stanford (NASA Grant No. NSG-2313) by the Technology Utilization Office of NASA Ames Research Center in cooperation with the Man-Vehicle Systems Research Division there. We are extremely grateful to the Stanford Biomedical Technology Transfer Team, supported by NASA, for the many long hours from Gene Schmidt, M. D. who has coordinated efforts between NASA-Ames and Childrens Hospital at Stanford.

*Bibliography*

1. Goodenough-Trepagnier, Cheryl 1976. Language development of children without articulate speech, paper presented at the Conference on the Psychology of Language, Stirling, Scotland, June, 1976 and also in *Recent Advances in the Psychology of Language,* Eds. Robin N. Campbell and Philip T. Smith, University of Stirling, Scotland.

2. Grace, Linda and Ashby, Faye 1979. The 1977-78 Narrative Analysis for the ESEA, Title IV, Part C Project: Non-Oral Communication Center, Project Number 3464, Fountain Valley School District, Plavan School, 9675 Warner Avenue, Fountain Valley CA 92708. Write Judy Montgomery, M. A., C. C. C., Project Director for copy of report.

3. Kucera, Henry and Francis, W. Nelson 1967. *Computational Analysis of Present-Day American English,* Providence, Brown University Press, 424p.

4. LeBlanc, Maurice 1976. Communication devices for non-speaking people, American Occupational Therapy Association 1976 Annual Conference, San Francisco, CA, October 11-15, 1976.

5. Neisser, Ulric 1967. *Cognitive Psychology,* New York, Appleton-Century-Crofts, 351p.

6. Simpson, Carol 1976. Effects of linguistic redundancy on pilots' comprehension of synthesized speech, *Proceedings of the Twelfth Annual Conference on Manual Control,* NASA TMX-73,170, pp. 294-308.

7. Simpson, Carol A., Lingel, Charles D., and LeBlanc, Maurice A. 1979. Design Principles for a Versatile Portable Speech Prosthesis, in *Personal Computing Proceedings,* National Computer Conference, June 4-7, 1979, New York City, pp 87-93. Edited by Jay P. Lucas and Russell E. Adams, U. S. Patent and Trademark Office

TABLE 1

| | |
|---|---|
| ALPHABETICAL X SINGLE X NO AFFIXES X NO BOXES | (AL,S,-A,-B) |
| ALPHABETICAL X SINGLE X AFFIXES X NO BOXES | (AL,S,+A,-B) |
| ALPHABETICAL X CLUSTERS X AFFIXES X BOXES | (AL,C,+A,+B) |
| ALPHABETICAL X CLUSTERS X AFFIXES X NO BOXES | (AL,C,+A,-B) |
| ALPHABETICAL X CLUSTERS X NO AFFIXES X BOXES | (AL,C,-A,+B) |
| GRAPHOTACTIC X SINGLES X AFFIXES X NO BOXES | (GR,S,+A,-B) |
| GRAPHOTACTIC X CLUSTERS X AFFIXES X BOXES | (GR,C,+A,+B) |
| GRAPHOTACTIC X CLUSTERS X AFFIXES X NO BOXES | (GR,C,+A,-B) |

TABLE 2

| | |
|---|---|
| 1. AIRCRAFT | 7. OPTIMAL |
| 2. COVERS | 8. PHRASE |
| 3. DEALER | 9. SOON |
| 4. MOTHER | 10. SUPREME |
| 5. MOVES | 11. TRAVEL |
| 6. NOTE | 12. WORN |

TABLE 3

VISUAL SEARCH TIMES FOR LETTERS PAGES

| | ALPHABETICAL | GRAPHOTACTIC |
|---|---|---|
| SINGLE LETTERS NO AFFIXES | 5.75 | --- |
| SINGLE LETTERS W/ AFFIXES | 6.27 | 8.87 |
| CLUSTERS W/ AFFIXES W/ BOXES | 7.61 | 10.53 |
| CLUSTERS W/ AFFIXES NO BOXES | 8.23 | 10.62 |
| CLUSTERS NO AFFIXES W/ BOXES | 8.32 | --- |

NOTE:
CRITICAL DIFFERENCE
1.94 SEC FOR P < 0.05
2.60 SEC FOR P < 0.01

TABLE 4

SIGNIFICANT DIFFERENCES BETWEEN MEANS

| | | | | |
|---|---|---|---|---|
| 1) | AL,S,+A,-B | faster than | GR,S,+A,-B | p = 0.01 |
| 2) | AL,C,+A,+B | faster than | GR,C,+A,+B | p < 0.01 |
| 3) | AL,C,+A,-B | faster than | GR,C,+A,-B | p < 0.05 |
| 4) | AL,S,-A,-B | faster than | AL,C,+A,-B | p < 0.05 |

ALPHABETICAL, SINGLE LETTERS, WITH AFFIXES

| | | | | | | |
|---|---|---|---|---|---|---|
| be | a | f | k | p | u | able |
| con | b | g | l | q | v | e |
| de | c | h | m | r | w | ed |
| dis | d | i | n | s | x | er |
| en | e | j | o | t | y | es |
| im | | | | | | ied |
| in | | | | | | ies |
| re | | | | | | ing |
| un | | | | | | ize |
| | | | | | | ly |
| | | | | | | tion |

GRAPHOTACTIC, CLUSTERS, NO AFFIXES, NO BOXES

| | | | | | |
|---|---|---|---|---|---|
| b | s | l | a | o | l |
| c | sc | m | ae | oa | m |
| ch | sch | n | ai | oe | n |
| ck | sh | r | e | oi | r |
| d | sk | w | ea | ou | w |
| f | sp | y | ei | u | y |
| g | st | | i | ue | |
| gh | t | | ia | ui | |
| h | th | | ie | | |
| j | v | | | | |
| k | x | | | | |
| p | x | | | | |
| ph | | | | | |
| ps | | | | | |
| qu | | | | | |

# MICROCOMPUTER/VIDEODISC CIA
# FULFILLING A PROMISE FOR HANDICAPPED STUDENTS

Ron Thorkildsen, Administrator, Exceptional Child Center
Assistant Professor, Instructional Media Department
UMC 68, Utah State University, Logan, Utah 84322
(801) 752-4100 ext 8273

## Abstract

This paper discusses the importance and need for individualized instruction for handicapped individuals and how CAI can influence this need. Special Educators have suspected for many years that the computer holds special promise in providing individualized instruction for handicapped students. This promise has not been fulfilled. The dream was ahead of technology. The advent of the low cost microcomputer combined with the rapid random access capabilities of the videodisc may very well provide the tools to fulfill this promise. A research project being conducted at Utah State Univeristy's Exceptional Child Center is investigating this premise. The major goal of the project is to develop a CAI system utilizing a microcomputer controlled videodisc to present CAI to mentally handicapped non-readers. The project is in its second year. One CAI program has been field tested and three more CAI programs are under development.

## Introduction

Early in the formal study of Special Education, teachers and researchers recognized the importance and effectiveness of individualized instruction for handicapped learners. There is typically such a wide range of intellectual experience among handicapped students that group based instruction is not effective. Many persons have long felt that the computer held a special promise in dealing with these individualized instruction needs. However, they also recognized the special communication problems associated with providing CAI to handicapped learners. This is particularily true with moderately mentally retarded learners who have little or no reading skills.

## CAI Limitations

A review of the literature has revealed little emphasis on using CAI with moderately mentally retarded or with non-readers. The lack of research in this area is due to the past limitations of computer hardware. To be effective for moderate and severely mentally retarded children, CAI must provide a full range of color, graphic, and auditory capabilities along with a variety of student response mechanisms. Computer auditory capabilities have been very limited, and because of the limited and costly nature of computer soft-generated graphics, CAI has been implemented with none or very primitive animation. Colby (1973) has stated that CAI allows only a limited number of responses and that we need to "....find a way to rapidly random access both sounds and pictures" (p. 260). Geoffrin and Bergeron (1977) in the conclusion of their report suggested that a larger variety of activities other than animation are needed and that these "....activities must be designed to train generalization reading skills to new words and situations other than computer animation" (p.12).

Hirschbuhl and Seeman (1975) have reviewed recent developments in CAI and give three recommendations for expanded application of CAI. They state that CAI must provide:

1. The ability to generate, program, and edit interactions with complex real images.
2. A technique for creating and delivering cost-effective graphics and visuals that allow for interaction.
3. The capability to provide overlays on images for sequential question frames. Such questions need to be communicated by audio messages.

Most of the limitations of CAI cited in the literature are related to video capacities. However, the most serious limitation with CAI for the mentally retarded has been the lack of random access audio. Considering the majority of the moderately handicapped have little or no reading skills, random access audio is important to a CAI system for this population. It is necessary that the CAI system have the capacity to provide instruction verbally. Random access is necessary to provide smooth and fast transitions to remediation material. A recent technological advance that has great potential to allow the implementation and manipulation of audio and video function is the random access videodisc player.

## U.S.U. Research Project

A research project currently being conducted

at Utah State University's Exceptional Child Center is utilizing this recently developed videodisc technology.[1] The major goal of the project is to develop and field test a prototype individualized instructional system for moderately mentally retarded learners using the videodisc player interfaced with a microcomputer system. The system developed by this project is referred to as the MCVD (Micro-Computer VideoDisc) system. This system will alleviate many of the problems mentioned by Colby (1973) and Geoffrin & Bergeron (1977). The videodisc offers a fast, economical and feasible way of storing and presenting audio/visual information. Specifically, the videodisc allows the random access of 54,000 frames of audio/visual information in less than eight seconds from any portion of the disc. Accessing segments within a program will require only a fraction of a second. The videodisc allows the use of both still pictures and motion pictures with or without the use of sound.

MCVD Hardware and Software

The hardware for the MCVD system consists of the following:

1. MCA Videodisc Model 7820.
2. Apple II microcomputer with 48k RAM and one floppy disc drive.
3. Sony 12" color monitor
4. Carol Mgf. touch panel

The MCA videodisc was selected mainly because of its rapid random access capabilities and its availability. The Apple II was selected because of its color graphics, portability and low cost. The touch panel was included to allow the child to interact with the system by touching the screen. This eliminates the abstraction of touching a key to represent an image on the screen. An interface board has been designed and developed which allows the microcomputer to control the videodisc and receive input from the touch panel. The interface board also has the capability of selecting a video signal either from the videodisc or the microcomputer. The source of the video is program selectable.

The computer language that enables the instructional material to be presented is PILOT. This language is designed specifically for CAI and is useful because educators with little or no computer experience can write CAI courses. PILOT has been modified so that programs written in this language can control the videodisc and accept input from the touch panel. This required the addition of three commands to the PILOT language:

1. P: *F1, F2* This command searches to the frame number of the videodisc specified by *F1* and plays to the frame number specified by *F2*.

2. L: $x,y,r$ This command allows a match from input from the touch panel. The coordinates of the center of an object on the screen are defined by $x$ and $y$ and the acceptable range from the center of the object is defined by $r$.

3. I: *T1, T2* This command allows the programmer to specify a time *(T1)* before input can be accepted from the touch panel and a time *(T2)* to wait before indicating that there has been no response.

Four instructional programs are currently being developed for use with the system. These programs are designed to teach:

1. Discrimination of sizes, shapes and colors.
2. Timetelling.
3. Recognition of survival words.
4. Understanding of functional words and phrases.

These instruction programs are being adapted from programs developed by the Outreach and Development Division of the USU Exceptional Child Center.

System Operation

The following briefly describes how the system interacts with the learner:

The system presents an audio instruction and an associated visual image via the television monitor. The learner responds by touching the television screen. The area touched on the screen is transmitted to the microcomputer via a touch panel and the response is evaluated. If the response is correct, a signal is sent to the videodisc which contains an audio and visual reinforcement (a film clip). Other possible response conditions are a wrong response, a close response, and a non-response. The non-response is detected by allowing a specified period of time to respond. There will be recorded segments on the disc for all possible response conditions as well as a variety of positive reinforcements varying in length and type. There is also remediation segments which can be branched to when appropriate.

Parameters are set in each lesson within a program which determine: (1) the number of times the learner must respond correctly before branching to a reinforcement segment; (2) the number of times the learner can

[1]The 2nd and 3rd year of this project is being funded by a grant from the Bureau of Education for the Handicapped of DHEWs.

respond incorrectly before branching to a remediation segment; (3) the number of times learner is allowed to cycle through the same lesson without reaching criterion before the teacher is signaled and asked to intervene. A time limit is set for each session at which time the teacher is signaled to end the session. The instructional segments are presented smoothly and without appreciable gaps between sections because of the random access nature of the videodisc playre. Each time a search is initiated, video is switched to orginate from the microcomputer and a computer generated graphic is displayed. After the search is completed, the source of the video is switched back to the videodisc.

## Conclusion

The project is progressing as planned. The matching, shapes, size and colors programs has been pressed on a videodisc. A preliminary field test of this program was conducted during the summer of 1979, and another field test will be conducted in January of 1980. The time-telling program is under development and will be field tested in March 1980. Data from the preliminary field test indicated a positive reaction by the child to the matching program and the system. Effectiveness data was not collected at that time, although the researchers feel confident that the system will prove both instructionally effective and cost effective. There is a strong feeling that the long awaited for promise of the computer to special education is now within reach.

# SOFTDOC

## A PROPOSAL FOR A MEDICAL SOFTWARE NETWORK

James Gagné, M.D.
President
DATAMED RESEARCH
1433 Roscomare Road
Los Angeles, CA 90024
(213) 472-8825

Abstract

Although computers have been utilized for health care applications for two decades, for a number of reasons widespread acceptance among clinicians has been disappointingly slow. The introduction of new microcomputer hardware, though clearly capable of supporting sophisticated medical applications, is not likely by itself to lead to a surge of medical computing.

The primary problem has been the indiscriminate throwing of masses of computer technology and software at a medical problem by those without an intimate understanding of the clinical process. By contrast, successful medical applications are most likely to stem directly from health professionals who have an interest in computing and who are willing to share their products with others.

Datamed Research is announcing the formation of SOFTDOC, a medical software exchange club. Interested physicians and other health professionals are invited to donate CP/M-compatable or UCSD Pascal source code medical programs on 8-inch flexible disks, in return for a free disk volume full of such donated software. Others will be charged a minimal fee ($15) per disk. Additional services related to medical software, such as compliations of user evaluations of commercial products, will also be offered.

This paper evaluates the history of medical computing, discusses the problems of the past, and offers suggestions for the creation of successful medical software.

Introduction

In the quarter century since computers first became generally available, their acceptance in society has grown exponentially. Few large businesses, for example, are without a data processing network. Yet in the health sciences, computers have taken root only in the laboratory and the business office. Despite innumerable articles on medical data base systems, diagnostic and treatment programs, comprehensive intensive care data monitors, and other electronic marvels, medical computing has not been accepted by health care professionals. With rare exceptions, physicians and other practicioners have utilized such systems only under pressure from their superiors. In fact, the clinical computing literature is replete with "good ideas" which were never even implemented (4; 6; 9).

Although it is reasonable to expect growing acceptance of microcomputer-based patient billing systems for individual practicioners' offices, there is little indication at present that clinical applications of these machines will necessarily result. In fact, we may be destined to perpetuate this ineffective use of medical data processing. A number of physicians are becoming enamoured of the newly affordable microcomputers and dream naively of a computer in every doctor's office. In the last decade alone, dozens of physician entrepereurs have launched medical computer systems based upon their pet computer projects. Their ventures sank both because they did not meet the needs of other physicians and they were poorly marketed.

The poor reception of medical computing is curious in light of the ease with which science has accepted computers in general. After all, medicine operates with a "technological imperative"--a nearly universal and often uncritical demand for newer and better medical technology, widely considered the mainspring of progress. For example, despite widespread use, there is growing doubt that coronary care units or routine electronic monitoring of mothers in labor or even the current wave of coronary artery bypass surgery have been shown to be medically useful and cost effective. Yet their popularity is undaunted (6).

Why haven't clinical applications of computers caught on in such a climate of technological largesse?

What Went Wrong?

The medical computing literature continues to be filled with glowing descriptions of systems. Typically, however, these projects are abandoned when exploratory grants run out, although this fact seldom appears in print. There are only rare reports critically evaluating the actual status of medical computing (4; 6; 9; 15; 19). In discussing past problems I will add my own speculations to what others have reported.

A. Physicians are inherently conservative, due to decades of discovering that last year's "miracle" treatment has turned out a dud, or causes an occasional but deadly side effect. They require considerable seduction to accept a new technique not widely adopted by their fellows or praised in a major journal, particularly in areas in which they feel awkard. And many M.D.s are intimidated by computers

B. No one likes to be ordered about in an area where she/he feels competent, particularly physicians, whose individuality at times borders on monomania (15). As a result,programs designed to enhance clinical decision-making run the risk of establishing a "heads you win, tails I lose" situation. If the computer made the correct decision when the doctor didn't, then obviously the doctor was "dumber than that stupid machine." If the program was in error, then what good was it? Moreover unfortunate topics were chosen: MYCIN, a brilliant program created at Stanford University, was designed to inform the physician of the correct choice of antibiotics in patients critically ill with infections (16). However, because surgeons and internists have disagreed about antibiotics for years and have firm opinions on the subject, there was little likelihood the program would have been consulted. In fact it was rarely used (17).

C. Friedman and Gustafson (4) pointed out that the majority of computer applications in health care accomplish nothing more than "automating" a process already performed adequately by health professionals on their own. Hence their lack of acceptance. By contrast, computer-assisted tomography could not exist without computers; its acceptance is unprecedented.

D. Medical data base systems (where all the data pertaining to the care of each patient is maintained on line) are generally considered to have been failures. Of the more than 100 systems described in the literature, only 17 were implemented. Of these 17, none have lived up to their designers' expectations; many have been abandoned. In a study of health practicioners acturally using a medical database system (22), clarified the problems inherent in paperless medical records. While some clinical data is easily expressed in terms a computer can understand, such as the white blood cell count or the serum bilirubin, it is difficult to describe via a menu of characteristics the appearance of a patient who may be depressed but you're not sure, or a bizarre shadow on a X-ray. The range of clinical data is simply too vast to fit comfortably in a structure where descriptors must be selected from a list. Moreover, to enlarge the data base so that it could contain a barely adequate description of the patient's condition noticeably lengthened the time required to record a patient's condition during a visit. Dictating or even hand writing was both quicker and more accurate than the computer utilizing sophisticated light-pen input.

E. In contrast with the competitively commercial origins of most business software , where ineffective programs failed, the vast majority

of medical applications programs were born in ivory towers, shielded from the demands of the users by their grants. The history of medical computing can be summarized in an oversimplified manner as a series of false starts. It is my impression that too much was attempted too soon, with little attention paid to working closely with the intended users of the programs. Others agree (4; 15).

This situation corresponds with the most common cause of failure of any applications program. It does not work to throw a computer at a situation without determining both exactly what is needed and wanted, and whether a computer will add sufficiently to the way things are currently being done to warrant the change.

## What Is the Current Potential For Change?

We are at a turning point in computer history. For the first time, low-cost systems are available with sufficient speed and mass storage capability for serious applications programming, with the promise of even greater capacity at a lower cost in the near future.

Thus, we are no longer confined to large minicomputers or expensive mainframe systems, which in the past has restricted medical computing to academia and large institutions. A new vista has opened for the health professions. The hardware beckons us within every practicions office if we can produce quality software of sufficient utility.

To reiterate the magnitude of recent advances in electronic technology, note that several manufactures are adapting the video disk system to serve computer storage purposes. A single $15 video disk will contain roughly two billion characters, equivalent to the contents of several hundred medical journals stripped of advertising. Obviously, this much storage can be adapted to a multitude of other uses.

Although medical computing can boast of few currently accepted applications beyond technical roles, such as computer-assisted tomography and on-line electrocardiograph and pulmonary function test interpretations, there are a few clinical areas where it works well. For example, there appears to be good patient and practicioner acceptance of an on-line Past Medical History and Review of Systems questioning system (i.e., the long list of routine questions asked of every patient undergoing a thorough medical examination) (1; 18). Utilizing direct patient interaction with a simplified terminal, these systems utilize a highly branching questioning scheme; for instance, if a patient says he has no headaches, the program skips all queries about the details of headaches. In the University of Wisconsin system, the patient's appointment with the computer takes about an hour. The data is then summarized and printed out as a report for use by the practicioner.

One could make a case for this type of system to be a model for medical computing programs. Note the advantages;

A. Practicioners rarely subject their patients to every appropriate question in a Review of Systems history because it is so lengthy; as a result many important areas are left out, particularly psychological and social functioning. There is excellent evidence that this occurs to the detriment of patient care (23). In contrast, there is no reason to cut corners in the computerized system, and there is empirical evidence that computer-assisted screening histories are, in fact, more complete(1)

B. One may easily add validation questions to the system to ensure the patient is using the system properly and to enhance the reliability of the data (18).

C. There is no direct practicioner interaction with the computer required.

D. The system relieves the practicioner of a routine, repetitious task, leaving him/her free to validate and interpret the results. A substantial saving of expensive practicioner time results.

E. There is excellent patient acceptance of computer-taken histories, particularly if they are fun to use (18).

F. If one wishes to expand the program, it is not difficult to counsel the patient while taking the history. This is best done at the patient's option ("Do you want to learn more about...") (18).

In summary, there can be little doubt that in this application a computer has the potential simultaneously of improving the patient data

THE TEN COMMANDMENTS OF MEDICAL COMPUTING

I. Start with smaller scale projects until it is learned what works.

II. It is critical that those intending to write medical software be intimately familiar with the needs of the people the program is designed to serve. For all practical purposes, then, for now medical applications should be coded by the physicians themselves.

III. Minimize direct physician interaction with the computer; if possible have the patient or nurse or receptionist enter the data required.

IV. Ensure that computer-generated reports are clear, accurate, and succinct.

V. Do not try to tell someone something he/she thinks he already knows. When one does offer recommendations to the physician, ensure that one does so carefully. Be gentle. Suggest rather than command. Hedge. Offer several options. Intend to support the practitioner rather than directing.

VI. Take into account the substantial differences in substance and style among health care practitioners and accomodate their idiosynchracies.

VII. Pilot your software before attempting to peddle it: offer it to other health professionals for their comments and suggestions. See if your offfice manager, nurse, spouse, and their friend can readily comprehend your menus. Sit several people down at the terminal to see if anyone can win $10 by crashing the program.

VIII. Collaborate with others doing similar projects.

IX. In addition to ensuring that your programs are correct, make them friendly, fun and ease to use. Communicate with the user in full, rich English phrases rather than code. Intend to relieve boredom and support the user's psychological state as well.

X. Above all, ensure that your applications program substantially improves upon the manual method currently employed. Preferable, there should be a direct and obvious impact upon patient care.

Table 1.

base, enhancing patient care, and saving time and money.

SOFTDOC-A Good Way to Get Started

SOFTDOC is intended to serve a key function in getting started in medical applications software: sharing our materail with each other. It differs from other physicians computer associations in that it exists primarily distribute medical applications software on machine-readable media, and to assist its members in evaluating which programs work best.

The club functions as follows: software articles and comments will be donated by physicians for general distribution. It will be collected on 8-inch diskettes, edited, and distributed at a price of $15 per diskette. Software will be the main emphasis.

Disk formats supported include CP/M (i.e. CBASIC, Microsoft BASIC, FORTRAN, COBOL, and, if you must, 8080 or Z-80 assembly language) and UCSD Pascal. Both must be on single-density, soft-sectored, IBM-compatible 8-inch diskettes, though if someone wished to assist in translating this standard format to another (North Star, Apple, TRS-80, Micropolis, etc.), we could accommodate physicians without larger disks drives. (However, if you're serious about medical microcomputing, you should have at least one 8-inch floppy drive).

Because of its greatly improved capabilities for serious programming, Pascal is the preferred language. Users of SOFTDOC will also receive notices of the exciting happenings within the users group established for UCSD Pascal, also supported by Datamed Research. Limited assistance will be available for those who wish to get a Pascal compiler going on

an existing system.

Datamed Research also offers consulting to physicians who wish to begin an office system de novo.

## Objectives

SOFTDOC has been established to achieve the following goals:

1. To explore the valid uses of microcomputers in the clinical health sciences, while aboiding the naive trumpeters of technology that plague many of the medical computing clubs.
2. To support physicians, dentists, psychologists, nurses, pharmacologists, and other health professionals, whose full-time commitment is to the clinical practice of the health sciences, yet who have a serious interest in developing practical medical computing.
3. To be an easy-to-use forum for sharing health-related software.
4. To allow constructive feedback from other users regarding programing efforts, in order to create the best programs possible.
5. To encourage high quality evaluation and outcome research exploring the efficacy of medical computing in reducing costs and improving patient care.
6. To collect from members user evaluations of commercial medical software, including practice accounting packages, and to assemble these into reports for the users.
7. On occasion, to alert users to important new developments in medical computing.

## A Word About MUMPS

There is no high-level language accepted as standard for medical computing purposes today. The proponents of one language, MUMPS (M assachusetts General Hospital Utility Programming System, developed in 1966), have suggested that it is ideal for medical computing and should become the standard (2). And although there is a fair amount of medical applicaitons software written in MUMPS, I disagree that it is ideal for much of anything.

MUMPS is, in fact, quite similar to BASIC--much more primitive in some ways, but also offering enhanced features. First implemented on a PDP-11 (on which the majority of MUMPS programs still reside), MUMPS is an interpreter desighed to run multiple small programs or procedures in a multitasking environment. All data is in the form of variable-length strings, to which a series of basic functions can be applied. MUMPS supports an elegant, heirarchical data base system that offers ease of access and automatic clearing of variables which are no longer needed. It is designed to be highly sparing of memory.

But MUMPS is not a friendly language. Because key words must begin with different letters of the alphabet to allow one-letter abbreviations, they do not precisely describe the functions they indicate. Critical elements of structured program flow are lacking, such as WHILE..DO and REPEAT...UNTIL. Rigid MUMPS syntax leads to awkard, hard-to-understand programs. Data structures are limited. In sum, it appears to be an early high-level language that is now outmoded.

In my opinion, Pascal offers most of the advantages of MUMPS with few features lost, principally the ease of establishing trees of strings and automatic "garbage collection" of memory occupied by discarded variables. With a proper library of procedures Pascal could easily assume the functions of MUMPS, trading noticably faster execution for increased data memory requiremenrs, though one would save the memory reserved for the MUMPS interpreter.

I have enumerated the advantages of Pascal in writing reliable software to say that the software already available in MUMPS is little argument to me to support the language.

## Future Directions for Medical Computing

Let us discuss several potentially useful computing applications for health care.

A. We covered on-line routine screening histories earlier. Clearly this is one area where development may begin at once, allowing for the necessary "learning time" for the program to mature with use. Griest (7) mentions the potential for simple data analysis within such a program. For example, Gustafson (8) showed that a simple on-line questionnaire could identify potential suicide attempters more accurately than could many physicians. Dr. Slack (18) discussed the importance of maximizing the patient's sense of control

of the program by allowing them to skip questions they chose not to answer. Questions that patients frequently answered "don't understand" were modified or eliminated, even though a back-up explanation was alread included in the program. Finally, Griest (7) agrees that there are advantages to giving patients a copy of their printed summary, if only to correct errors.

B. Computer-assisted instruction (CAI) may be useful both for patients and practitioners. Medical students have learned anatomy (10) and fetal monitoring (21) on line. Teaching patients preventive medicine is an ideal topic for CAI (20). Ellis, et al (3), taught health risk factors on an Apple, starting the program by reading the patient's blood pressure, serum cholesterol, smoking history, etc., whcih were then used to calsulate health risks. One hidden advantage for physicians is that the program may print out the patient's correct answers to test questions, unequivocably documenting for legal purposes that the patient understood the instructions.

A variation of computer-assisted instruction is computer-assisted informed consent. Here, too, the advantage would be iron clas proof that the patient understood the material offered. However, such a program used alone may be deadly to the doctor-patient relationship; it would be better utilized an adjunct, somewhat similar to a movie.

C. Word processing is certainly an accepted application of microcomputers. It remains to be seen if a ine medical word processing system, run directly by the practitioner to document the events of a patient visit, would be faster than handwriting or dictating. The potential for saving time leis in utilizing a scheme for greatly condensing the keystrokes required to enter a standard medical phrase, while aboiding making the code too hard to remember or the resultant text too much the same (11).

D. Medical diagnosis may be much simpler than the literature would have us believe, provided one limits one's scope to reminding the practitioner of possibilities he/she may have forgotten. A simple pattern-matching scheme, such as is utilized by INTERNIST (13), is highly effective. A good way to begin may be a program that listed likely rare diseases for a given symptom/sign complex, along with differentiating data.

## For Further Information

Please send a stamped, self-addressed envelope to SOFTDOC at the address given at the beginning of this paper.

## References


1. Angle, H.V., Ellinwood, E.H., & Carroll, J. Computer Interview problem assessment of psychiatric patients. Proceedings of the Second Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1978.
2. Bowie, J., & Barnett, G.O. MUMPS-An economical and efficient time-sharing system for information management. Computer Programs in Biomedicine, 1976, 6, 11-22.
3. Ellis, L.B.M., Raines, J.R., & Brown, J.W. Teaching health risk concepts using microcomputers. Prodeedings of the Third Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1979, 158-163.
4. Friedman, R.B., & Gustafson, D.H. Computers in clinical medicine, a critical review. Computers and Biomedical Research, 1977, 10, 199-204.
5. Gagne, J.L. An introduction to Pascal. Kilobaud Microcomputing in press.
6. Glantz, S.A. Computers in clinical medicine: A critique. Computer, 1978, 11, 68-77.
7. Greist, J.H. Computer Interviewing Beyond data collection. Proceedings of the Second Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1977, 52.
8. Gustafson, D.H. A probabilistic system for identifying suicide attemptors. Computers and Biomedical Research, 1977, 10, 83-89.
9. Johnson, J.H. Computers in mental health: Where are we now? Proceedings of the Second Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1978, 104.

10. Kerns, J.M., Snell, R.S., & Tidbal, C.S. Anatomical- Correlation: A computer Simulation for freshman medical students. Proceedings of the Third Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1979, 164-169.

11. McLeod, D.J., & Meldman, M.J. SELECT: An interactive minicomputer subsystem for the generation of specialized narrative text. Computers and Biomedical Research, 1977, 10, 91-99.

12. Mesel, E., & Wirtschafter, D.D. On-line medicaid billing system for physicians's services. Computers and Biomedical Research, 1975, 8, 479-491.

13. Myers, J.D., & Pople, H.E. INTERNIST: A consultative diagnostic program in internal medicine. Proceedings of the First Annual Symposium on Computer Application in Medical Care. Washington, D.C. 1977, 52.

14. Oldfield, H.R., Jr. Modern methodology for the self-administered medical history. Proceedings of the Second Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1978, 238-240.

15. Ronis, N. My girl sends out my bills--Doctors and computers. Proceedings of the Second Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1978, 488-497.

16. Shortliffe, E.H., et al. Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system. Computers and Biomedical Research, 1975, 8, 303-320.

17. Shortliffe, E.H. MYCIN: A knowledge-based computer program applied to infectious diseases. Proceedings of the First Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1977, 66-69.

18. Slack, W.V. Patient counseling by computer. Proceedings of the Second Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1978, 222-226.

19. Sobolewski, J.S. Toward the implementation of successful medical computer applications. Proceedings of the First Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1977, 84-89.

20. Van Cura, L.J. A self-administered health hazard appraisal. Proceedings of the Second Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1978, 231.

21. Votaw, R.G., & Miller, L.W. Intrapartum fetal monitoring: A computer-based approach to development and assessment of clinical competence. Proceedings of the Third Annual Symposium on Computer Application in Medical Care. Washington, D.C., 1979, 170-174.

22. Zielstorff, R.D., et al. Experience with a computer-based medical record for nurse practitioners in ambulatory care. Computers in Biomedical Research, 1977, 10, 61-74.

23. Duff, & Hollingshead. Sickness and Society. New Haven: Yale Academic Press, 1968.

# THE COMPUTER IN THE PRACTICE OF MEDICINE: AN OVERVIEW

Mark H. Spohr,M.D.
Medsoft
P.O. Box 7049
Tahoe City, CA 95730
(916) 583-3097

## Introduction

We are at the beginning of a decade which will see computers enter most phases of our daily lives. Inexpensive and easy to program "appliance" computers are now available for small business and home use. There are many possibilities for their application in the medical office.

What can this unique tool, the computer, do for you, the physician? Its skills lie in the areas of data collection, storage and manipulation. It performs repetitious and tedious tasks without tiring. Computers offer physicians increased organization, efficiency and productivity. Better management improves the quality of patient care and reduces medical costs.

Estimates of physician time spent on maintaining medical records range from 25% to 35%. Administrative functions occupy yet more time. These tasks are necessary. By performing them more efficiently, they require less time. This allows more time for patient care. Quality of patient care increases, while record keeping, administration and management costs decrease.

The large data storage and manipulation capabilities of the computer can be used to our advantage in making decisions. Differential diagnosis routines, treatment protocols and drug interactions are potential areas of use. In addition, the interactive capabilities of the computer can make programmed learning a reality for patient and physician.

This paper outlines four areas in which computers can aid a physician in performing his duties more accurately and efficiently:

- medical records (creation, access Storage)
- business management (billing, accounting, payroll, word processing, patient scheduling)
- diagnosis and care information (decision making)
- education (physician and patient)

## The Medical Record

The medical record is the physician's data base. It gives a patient's past medical history and is the record of past encounters. The story that it tells of symptoms and signs, laboratory values, of treatment and response to treatment is vital to good medical care.

The traditional medical record contains this information but has its problems. Because of the tedium of writing detailed descriptions, it is often too brief. A handwritten chart may be partially illegible even to the creator. Its organization may be poor with a hodgepodge of laboratory slips, radiology reports, letters from consultants and progress notes.

It lacks organization, it may lack legibility, it may even lack completeness. The effect of this is to make it frustrating (and at times impossible) to extract information from the record. More complex data analysis is almost impossible because of the time involved in gathering information from the chart. For example: a gragh of a patient'sblood pressure over the past year with notations indicating dates of various medications would allow response to treatment to be seen readily. To do this manually is very time consuming, but a computer can do it easily.

The electronic medical record is a 5" minifloppy disk which can hold about 100K bytes of information. This is approximately 100 typed pages. Of

course, there are very few office medical records that reach 100 pages in spite of the many pages which contain very little information, such as a page of laboratory results with only a few values. A minifloppy disk has more than enough room for most medical records.

The electronic medical record can contain the history and physical exam, laboratory values, X-ray reports, a problem list, and interval notes. The problem oriented medical record provides a good structure for the chart. The record is organized around a problem list. Following initial history and examination, a list of problems is made. Notes for each office visit are entered in terms of these problems. New problems can be added to the list and the old removed as they are resolved. The notes are stored on the disk in sequential order by date but can be searched randomly to extract information. As an example, all notes concerning a single problem such as environmental allergies can be displayed. Data routinely obtained on each visit such as blood pressure, pulse, temperature and weight can be collected and a graph made showing change over time. Other data such as laboratory values and X-ray reports are filed in their own section and can be accessed by date or type of data.

There are several areas in which increases in productivity can be made using the electronic medical record. A patient, sitting at a computer in a waiting area can complete a computer guided history. This use of the computer allows a large number of questions to be asked and enables branching to more detailed questions based on the patient's responses. The results are automatically stored on that patient's medical record disk. When the physician reviews the history, the computer can edit the data to display only positive responses.

The physical exam is then performed and entered on the record in the following manner. The computer displays a menu of normal and common abnormal findings which are selected to be entered on the patient's chart. Provision is also made for individual entry of unusual findings. On subsequent visits, the same format is followed. The computer has a large library of common problems and their findings and treatment. These are selected from a menu and entered into the patient's record.

One can see that this system allows complete, legible, well organized charts to be created and referenced with a minimum of effort. The chart is reduced in size and storage space for patient records can be reduced proprotionally. In the event a paper copy of the chart is needed outside the office, it is easy to list the record on a printer. The electronic medical record also permits the unique function of analysis of information in the chart.

Office Management

The efficiency and accuracy of the computer can help you minimize the time spent on tasks assiciated with running the business of a medical practice. One of the main functions of the business is billing. There are a multitude of forms to be sent to insurance companies, and bills need to be sent to patients. The computer excells at filling out forms. It puts the proper information in the appropriate spaces and checks for completeness with tireless enthusiasm. This tedious task is ideally suited to the computer.

The computer can also be used as a word processor. It can type letters to referring physicians and to patients and also letters referring patients. If you will analyze your correspondence you will find it contains a number of common paragraphs with minor differences. These paragraphs can be stored in the computer. A letter can then be composed on the video display from these standard paragraphs with identifying and personalizing information inserted in the appropriate areas. The computer then types the letter quickly and accurately.

The computer can type a complete copy of a patient's chart in minutes if it

is needed outside the office by another physician or in the hospital when a patient is admitted. The original is always kept safe in your office.

Other functions such as an electronic appointment book can recall information for easy reference such as chart number, address and reminders of lab tests or X-rays.

Payroll and accounting are standard business functions that when done with a computer can improve the accuracy of your books while less time is spent on their maintenance.

## Diagnosis and Care Information: Aids to Decision Making

The office computer has the ability to store, recall and analyze large amounts of data as an aid to diagnosis and treatment. One applecation is differential diagnosis. Routines can be stored for easy retrieval. A patient's appropriate signs, symptoms and lab are entered into a protocol with branches calling for more information when needed. The output is a differential diagnosis and plan for further tests to narrow the differential. These diagnostic programs have the advantage that they are readily available, self-prompting, individualized, can consider a large number of possibilities, and remember the differential for each in great detail.

Another aid to diagnosis is analysis of laboratory tests such as glucose tolerance and thyroid function. The data can be analyzed by uniform criteria and appropriate action suggested.

Drug dosage which depends on a patient's age, sex, weight, surface area, renal or liver function can be calculated easily to take these differences into account. Similarly, patients on multiple drugs can be screened for possible interactions. This can be performed quickly and easily by the computer. At present it is not usually done because it is not accomplished easily or quickly.

The computer can also be used to perform health assessments of risk factors individualized for each patient's situation. For example, this may motivate him to alter his behavior when it can be shown that for each pound overweight he is, 0.2 years are subtracted from life expectancy.

## Physician and Patient Education

The computer presents the opportunity for programmed learning on an individualized basis. Programmed learning has the advantage that it is active rather than passive. The program can tailor itself to the level of knowledge of the user. It can instruct then ask questions. It can display graphics as well as text for a complete presentation. Depending on the answers to questions it can repeat or expand on specific areas of knowledge. Areas in which the student demonstrates competency can be passed over. The program can give rewards for correct responses or completion of certain tasks. In the case of patient education, areas of residual uncertainty can be identified for further explaination by the physician. A post-test can assess the knowledge gained and areas in need of further study.

Teaching programs can be developed and distributed on a subscription basis and credit assigned for completion of the course.

## Implementation Philosophy: Hardware and Software

The hardware to perform these functions consists of small, inexpensive "appliance" computers which can be used by people with little training. These have been chosen because they are readily available, reliable, easily serviced and low in cost. A large number of these have been produced and there is a fair amount of software available from multiple sources.

The low cost of the computers makes it possible to have a complete computer dedicated to each task. The hardware is redundant in that each unit is composed of standard building blocks and there are several of these in each system. This gives an extra measure of reliability. In the event of a failure of one system, modules from another system can be substituted. The defective module can be sent for repair on a low priority (and lower cost) basis while the more critical computing functions can continue.

The software is modular in design. Each function is implemented independently. The modules are complete in themselves and can be linked together, passing information from one program to the next. As an example, the self-administered history becomes part of the electronic medical record or it can be printed and used as part of a conventional medical record. This allows the system to be gradually integrated into one's practice. The modules can be customized to each physician's mode of practice. Areas of specialization can be accomodated.

Summary

We can improve the quality of medical care with the computer by allowing the physician more time for his patients, giving him better access to information about his patients and better information about diagnostic choices and therapeutic regimes. We can also give the patient better information about his illness and educate him about the steps he can take to improve his health.

There are many functions which lend themselves to computer assistance. These are functions which utilize the large storage capacity, speed, meticulous accuracy or tenaciousness of the computer to perform tasks requiring these skills. The areas of computer use are:

- the medical record
- office business functions
- diagnosis and care information
- education

We are at the beginning of the decade which will see computers enter most phases of our daily activities. The inexpensive appliance computers which are now available are prompting many people to become computer literate. The physician of the next decade will be first exposed to computers in grade school, will use and program them throughout high school, college and medical school, and will expect to employ them in his practice. There will exist an internalization of computer use. While present day physicians have not had this advantage, we can recognize current trends and take steps to implement computer use in our practices with benefits to ourselves and our patients.

# PERSONAL COMPUTERS IN THE OFFICE: AN EXAMPLE

**Clarence A. Ellis** and Gary J. Nutt
Xerox Palo Alto Research Center
3333 Coyote Hill Road / Palo Alto / California 94304

## ABSTRACT

One facet of office automation research is presented by describing an experimental office information system prototype developed at the Xerox Palo Alto Research Center (PARC). The discussion is of interest because of our position that many future office systems will be implemented in the environment of a network of personal computers, sharing information and facilities in a loosely coupled fashion. We first provide an intuitive description of an Office Information System, and then we give some details of the Officetalk-Zero prototype.

## INTRODUCTION

The automated office of the future is stimulating new interests within the computer science research and development world, (e.g., see [HAMM79, HEWI79, MORG76, NEWM79a, TSIC79, WHIT77, ZISM78]). Instead of emphasizing traditional business data processing, or management information systems, the new thrusts emphasize systems and facilities to aid the office worker in the more basic aspects of his/her job. Word processors address the problem of document preparation, but the worker must also organize, file, copy, transform, analyze and transmit that information effectively. The automated office can greatly aid the user in performing these functions, and present these options to the user via a simple, integrated interface.

Automated office systems offer a new area for applying results, techniques and methodologies of classic computer science; solutions to a large number of difficult problems must be obtained before such systems can become a reality. Many such problems are a result of three more general problems: the complexities of distributed systems that implement the automated office, the necessity for simple, yet complete human interfaces, and the need for knowledge-based systems to aid the user. These issues are discussed at length in a survey paper that appears elsewhere [ELLI80]. We have also included an extended bibliography at the end of this paper to show other relevant work, and to indicate further breadth and depth of office information system research.

The *office* is that part of a business that handles the information dealing with operation, accounting, payroll, billing, etc. In particular, office work consists of activities such as document preparation, filing, performing simple computations, checking information, intraoffice communication and external communication. Such processing within the office is usually stimulated by the arrival of a request for service such as an order, a bill, a complaint, a message to order more materials, or the date changing to Friday. The office, then, can be viewed as a mechanism that maintains the state of the business, by means of a series of activities that cause transitions.

An *automated office information system (OIS)* attempts to perform the functions of the ordinary office by means of a computer system. Automation in the office particularly aids the office worker in document preparation, information management and decision making. Such systems may be as modest as a group of independent word processors, or as complex as a distributed set of large, communicating computers. Within this spectrum is a central computer with several interactive terminals, or a set of interconnected personal computers. In either system the office worker would use a *work station* to perform his work, and that work station would be capable of electronically communicating with other work stations.

We distinguish office information systems from data processing systems both by the autonomy of the system's parts, and by function. A data processing system is used to implement algorithms with a single locus of control in which there are ordinarily not collections of autonomous parts; the algorithm ordinarily proceeds without the need for human interaction. Typical data processing systems compute payrolls, implement accounting systems, manage inventories, etc. An OIS is made up of a collection of highly interactive autonomous tasks that execute in parallel; the tasks include document preparation, document management, communication, and aids in decision making.

As an extended example, the remainder of this paper is a discussion of a prototype system, Officetalk-Zero, which explores the user interface facet of office information systems. Officetalk-Zero is an experimental distributed office information system, designed and implemented by William Newman, Tim Mott and others from the Office Research Group at Xerox PARC.

The discussion of Officetalk-Zero, or Officetalk for short, is divided into sections which illustrate a hardware view, a user's view, and a software view of the implementation.

## THE OFFICETALK PROTOTYPE

The Officetalk designers took the position that the new OIS should be based on the data objects of single page forms and files of forms; intercommunication is accomplished by electronically passing forms among the work stations. The user's model of the system is that Officetalk is merely an electronic aid for carrying out his normal tasks. A primary difference in the user's model (as opposed to his pre-OIS model) is the lack of real paper at the user's work station. Each work station provides a graphical window onto a worker's desk, allowing the worker to manipulate electronic forms by employing a pointing device.

Many of the individual facilities needed to implement Officetalk already existed as separate programs on several computer systems. The user must have a text editor, a graphics package, electronic mail, a filing facility and a forms data entry capability. However, an OIS must offer all of these facilities to the user via a *simple, uniform interface*.

### The Hardware Environment

The Officetalk hardware environment is based on the Xerox Alto personal computer [THAC80], a local network communication system (informally called the "Ethernet") [BOGG79, METC76, SHOC79], a remote hardcopy device, and a remote file system [PAXT79, SWIN79].

The Alto personal computer was designed to be used in a research environment in which properties of personal computers and distributed systems could be studied and tested. The basic configuration of the Alto used for implementing Officetalk incorporates 128K 16 bit words of 850 nanosecond memory, a 2.5 megabyte disk, an Ethernet interface, a CRT display, a keyboard, a keyset (function keys), and an x-y coordinate input device called the *mouse*. The Alto processor is based on a multiplexed microprogrammed machine which supports I/O and provides emulators for different abstract machines. The microprogrammed tasks are particularly suited for handling network communications; keyboard, keyset, and mouse input; and supporting the display screen.

The display screen is a 606 x 808 point grid with two-level (black and white) pixels; the display is defined by a bitmap kept in the primary memory. Areas on the screen are pointed to by a cursor under the control of the mouse. The mouse is operated by a button which is depressed, then released; the Alto firmware determines the state of the button as well as the x-y coordinate addressed by the mouse.

The filing system, called a *file server*, is implemented by an Alto with a large disk and software to allow remote clients to request (page-level) access to the system's local files. The file server is not designed to be distributed over multiple Altos, although extensions can be made to the basic facility to allow such distribution [PAXT79]. Thus all personal computers (there are several hundred around PARC) can access the file server via the Ethernet.

The Ethernet [BOGG79, METC76] is a three megabit per second local communication system for connecting computing facilities. Our Ethernet uses tapped coaxial cables to carry packets of digital data among personal computers, large file storage devices, printing facilities, large central computers, longer-haul communication equipment, and other experimental facilities.

The above facilities were used within our OIS experiment because of availability, and because of our belief that many future automated office systems will be designed around a similar physical environment [CREA78].

### The User's View of Officetalk

Officetalk is a distributed program that executes on at least one work station in conjunction with a hardcopy server and a file server. The user's view of the system makes this distribution explicit. The file server maintains a database describing all pending electronic transactions, e.g., electronic mail, information about each authenticated user of the system, or a set of tailored blank forms to be used in the particular application.

To use Officetalk, a set of blank forms for the particular application must be designed and entered into the database. Officetalk includes a forms editor to design the artwork of a form and to specify the style of each field on the form. The forms editor requires that the forms satisfy certain rules, such as no overlapping fields; it also permits certain fields to be designated as signature fields. (Signature field entries can only be filled in with the image of the current user's signature.)

Upon logging into Officetalk, the user is shown an image of a desktop containing parts of forms, as shown in Figure 1. The user employs the mouse to manipulate the forms on the desktop. Each form is displayed in a rectangular *window* on the CRT device. The form may be larger than the window; hence, the user is allowed to enlarge (shrink) the window or to scroll the form within the window by pointing to appropriate parts of the window frame. The user can also move the window around on the display screen by "picking up" the window with the mouse and then moving the mouse. If the new window position overlaps another window already on the screen, Officetalk treats the two windows as pieces of paper. The last window that is "laid down" is wholly visible, while intersecting windows are at least partially "covered up". Each window includes a menu of Officetalk commands which can be applied to the form that is visible in the window. The particular menu that is used is a function of the type of the form showing in the window. The mouse is used to point at commands in order to invoke them.

An Officetalk desktop contains four basic forms: an *in-basket* of incoming mail; an *out-basket* specifying mail to be sent and mail that has been recently sent; a *file index*, used as an index of forms that the user has saved; and a *blank stock index* of the set of available forms. When the user wishes to generate a document, he or she selects a blank form from the blank stock index; the form is drawn in a new, fully visible window. The user may then enter information into the form by pointing at a field and typing a character string (or causing a signature to be entered). The editor restricts the data types to match the form's field definitions, e.g., a signature field can contain only a signature. Officetalk also allows the user to draw freehand on a form; the mouse is used as a "brush" which can take on several different styles. Freehand illustrations can later be removed without harming the form's layout or previously typed information. Once a document has been prepared, it can be filed in the user's personal file, in which case it will have an entry in the personal file index mentioned above. Or perhaps it may be copied, the original filed, and the copy placed in the out-basket for mailing. The contents of the out-basket are actually mailed (placed on the central file server) when the user points to a *transmit* selection in the menu of the out-basket.

The user can work on an existing document by retrieving a previously-filed form from any file index. Electronic mail is routed from the sender to a mailbox on the file server; the mail is moved to the local in-basket by pointing to a menu selection. Forms that have been mailed can be traced by the user. When the trace option is chosen, Officetalk opens a window on the electronic desk and then describes the current location of the form and an audit trail describing its route to that location.

### Some Software Issues in Officetalk

Officetalk integrates a number of facilities that exist in many different systems into a single interface. In particular, implementations of windows [GOLD79, TEIT77], electronic mail [HEND77], file servers [SWIN79], and remote hardcopy facilities existed before Officetalk was designed. The interface takes advantage of the interactive graphics capability of the Alto to provide a uniform user interface. For example, the user can shuffle paper, read mail, or read previously filed documents using only the pointing device.

There are several aspects unique to the Officetalk design, but only a small sample are mentioned here: form space management, display management, and forms editing. Officetalk is designed to save the majority of the user's information state in the file server and as little as possible in the local personal computer; this reduces the dependency of a user on a particular work station or disk. However, it introduces a memory management problem among the file server, a work station's local disk, and the work station's primary memory; the primary memory can be used more effectively if parts of a form are "demand paged" from the local disk, which, in turn, is paged from the file server. A careful structuring of graphical display information, forms template information, and form-specific data allowed experimentation with tradeoffs of network traffic versus local disk space utilization.

The basic software under the graphics package implements portions of Level 4 of the Core System developed by the ACM SIGGRAPH Graphics Standard Committee, [ACM78]. The Alto environment provides low level implementation of the *pick*, *locator* and *keyboard* input devices. The *viewing transformation* is defined by a bitmap for the screen; in order to place an image on the screen, it is necessary only to set the appropriate bits in the bitmap. Officetalk designers implemented the two-dimensional notions of *windows* and *view ports*, so that clipping, scrolling and moving windows could be handled efficiently. The techniques used in the display maintenance are described in NEWM79b.

Each window in the user's domain has a descriptor indicating the current size and location of that window, as well as other information about the window's content. Window descriptors are placed in a list in the order in which their windows should appear on the screen. Thus, window movement amounts to placing the window descriptor on the front of the window list and then updating the bitmap by first clearing the area in which the top window appears, then placing the window content in the bitmap, and then drawing (with clipping) the window whose descriptor is next in the list. When the mouse points to a window, the program searches the list for the first window to contain the x-y coordinate input. The menu selection is determined by both the identity of the window and the location within the window. (The location within the window specifies the function to be performed.)

Intelligent forms editing requires some thought. There are the usual low level interface problems: for example, how to select and replace text. Additionally, field types must be checked for proper values. While some fields may be unalterable after they have once been written into (e.g., the "amount" field of a pay voucher), other forms are copies and thus cannot be written upon. One important problem that arose here was how to visually identify a copy from the original. The approach taken was to provide a different set of capabilities for manipulating a copy than for manipulating an original; the menus for the two types of forms differ.

## CONCLUSION

Officetalk is a prototype office information system that integrates a set of common facilities into a single system with a simple user interface. It builds upon the users model of paper forms to allow added functionality with electronic forms images. The original Officetalk-Zero, as described in this paper, does not include any decision support facility, a desirable feature of a production OIS. Decision support can perhaps best be incorporated by providing a means for defining procedural specifications of office activities. Although this was a goal of the Officetalk-Zero study, the user interface turned out to be a hard enough problem to absorb the full energies of its designers. In order to increase the reliability of a distributed OIS, production systems are likely to incorporate more sophisticated database systems than that used in Officetalk. The designers chose to use an existing facility which does not allow a distributed database, which supports no query system, and which uses overly simplistic locking mechanism for data consistency.

Officetalk is intended to be used by office workers to aid in document management, preparation and communication. Part of the reason for restricting interest to clerical work was the desire to investigate office procedure specification and interpretation; the designers recognized that the procedural specification of "routine clerical work" was an unsolved problem, and that a solution to that problem would be a step toward the solution of the more general problem.

Modern office information systems may use several of the ideas employed in Officetalk-Zero, although a fully operational office information systems may also need additional features. Officetalk emphasizes the user interface and the integration of diverse facilities; other experimental prototypes have tended to concentrate on areas such as procedure specification [ZISM77], specialized programming languages [HAMM77], and cost/benefit analysis [WHIT77].

Our work on office information systems has shown us the need for *functional integration*, and for *system integration*. Functional integration refers to the need for the user's model of a system to be complete and consistent; the clerical user must be able to work in an environment that provides all of the facilities he or she will need in order to perform his or her work without having to learn several different command languages or subsystem models. System integration refers to the need for operating systems, programming languages, architecture, databases and artificial intelligence systems that converge into a single, uniform environment; for example, researchers at PARC have experimented with the Smalltalk environment as an integration of operating system, programming language, debugger, and text editor [KAY77, INGA78].

We have not discussed the social implications of the office information systems technology. There is a possibility for added freedom from drudgery and added functionality; there is a possibility for drastic alienation and unemployment. The technology in and of itself is neither good nor bad; it is the use of technology that determines its effect on the worker and on society. Clearly computer scientists need to interact with workers in psychology, sociology, management science, political science, and other disciplines; Wynn's work [WYNN79] is a good example of such interdisciplinary integration.

As a result of particular constraints on OIS applications, we may see several new and radical system designs emerge. For example, local networks of personal computers provide a physical medium for the design of exotic systems of work stations that share compilers, consistency-checkers and databases, while autonomously performing other tasks with private facilities. One can hypothesize an intelligent form which guides itself through various work stations and measures its own progress, utilizing the facilities of particular work stations within their own domains. The field is open, and the next few years will see heavy use of personal computers to implement office information systems.

## EXTENDED BIBLIOGRAPHY


ACM78 --, "Special Issue: Graphics Standards," *ACM Computing Surveys*, Vol. 10, No. 4, (December, 1978), pp. 363-502.

BOGG79 Boggs, D., J. Shoch, E. Taft and R. Metcalfe, "Pup: An Internetwork Architecture," to appear in *IEEE Transactions on Communications*, 1979.

CREA78 Creative Strategies International, "Office Automation," Creative Strategies International, Report No. 27804, July, 1978.

ELLI79 Ellis, C. A., "Information Control Nets: A Mathematical Model of Office Information Flow," *ACM Proceedings of the Conference on Simulation, Modeling and Measurement of Computer Systems*, August, 1979, pp. 225-240.

ELLI80 Ellis, C. A. and G. J. Nutt, "Office Information Systems and Computer Science," *ACM Computing Surveys*, Vol. 12, No. 1, (March, 1980).

GOLD79 Goldberg, A. J. and D. J. Robson, "A Metaphor for User Interface Design," *Proceedings of the Systems Sciences Conference*, University of Hawaii, January, 1979, pp. 148-157.

HAMM77 Hammer, M., W. G. Howe, V. J. Kruskal and I. Wladawsky, "A Very High Level Programming Language for Data Processing Applications," *Communications of the ACM*, Vol. 20, No. 11, (November, 1977), pp. 832-840.

HAMM79 Hammer, M. and M. D. Zisman, "Design and Implementation of Office Information Systems," *Proceedings of the NYU Symposium on Automated Office Systems*, May, 1979.

HEND77 Henderson, D. H., Jr. and T. H. Myer, "Issues in Message Technology," *Proceedings of the Fifth Data Communications Symposium*, (1977), pp. 6-1 to 6-9.

HEWI79 Hewitt, C., "Behavioral Characteristics of Office Systems," *M.I.T.-I.A.P Course on Electronic Office of the Future*, January, 1979.

HEWL78 Hewlett-Packard Company, *HP 300 Computer System General Information Manual*, September, 1978.

INGA78 Ingalls, D. H., "The Smalltalk-76 Programming System: Design and Implementation," *Proceedings of the Fifth Annual Symposium on Principles of Programming Languages*, January, 1978.

KAY77 Kay, A. C., "Microelectronics and the Personal Computer," *Scientific American*, Vol. 237, No. 3, (September, 1977), pp. 231-244.

METC76 Metcalfe, R. M. and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM*, Vol. 19, No. 7, (July, 1976), pp. 395-404.

MORG76 Morgan, H. L., "Office Automation Project: A Research Perspective," *AFIPS Proceedings of the NCC*, Vol. 45, (1976), pp. 605-610.

MORG79 Morgan, H. L., "Database Alerting and Corporate Memory," *M.I.T.-I.A.P Course on Electronic Office of the Future*, January, 1979.

NESS78 Ness, D., Office Automation Project, Decision Sciences working papers, Wharton School, University of Pennsylvania, 1976-1978.

NEWM79a Newman, W. M., "Office Models and Office Systems Designs," *Proceedings of the International Workshop on Integrated Office Systems,* November, 1979.

NEWM79b Newman, W. M. and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill Book Company, New York, second edition, 1979.

NUTT79 Nutt, G. J. and C. A. Ellis, "Backtalk: An Office Environment Simulator," *Proceedings of the 1979 ICC*, Vol. 2, (June, 1979), pp. 22.3.1 - 22.3.5.

PAXT79 Paxton, W. H., "A Client-Based Transaction System to Maintain Data Integrity," to appear in the *Proceedings of the Seventh Symposium on Operating Systems Principles*, December, 1979.

SHOC79 Shoch, J. F., "An Annotated Bibliography on Local Computer Networks (preliminary edition)," Xerox Palo Alto Research Center, Report SSL-79-5, May, 1979.

SWIN79 Swinehart, D., G. McDaniel and D. Boggs, "WFS: A Simple Centralized File System for a Distributed Environment," to appear in the *Proceedings of the Seventh Symposium on Operating Systems Principles*, December, 1979.

TEIT77 Teitelman, W., "A Display Oriented Programmer's Assistant," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, (1977), pp. 905-915.

THAC79 Thacker, C. P., E. M. McCreight, B. W, Lampson, R. F. Sproull and D. R. Boggs, "ALTO: A Personal Computer," to appear in *Computer Structures: Readings and Examples*, edited by D. Siewiorek, C. G. Bell and A. Newell.

TSIC79 Tsichritzis, D., "A Form Manipulation System," *Proceedings of the NYU Symposium on Automated Office Systems*, May, 1979.

WHIT77 White, R.B., "A Prototype for the Automated Office," *Datamation,* Vol. 23, No. 4, (April, 1977), pp. 83-90.

WYNN79 Wynn, E., *Office Conversation as an Information Medium*, Ph.D. dissertation, Department of Anthropology, University of California, Berkeley, 1979.

ZISM77 Zisman, M. D., *Representation, Specification and Automation of Office Procedures*, Ph.D. dissertation, Wharton School, University of Pennsylvania, 1977.

ZISM78 Zisman, M. D., "Office Automation: Revolution or Evolution," *Sloan Management Review*, M.I.T., Vol. 19, No. 3, (Spring, 1978), pp.1-16.

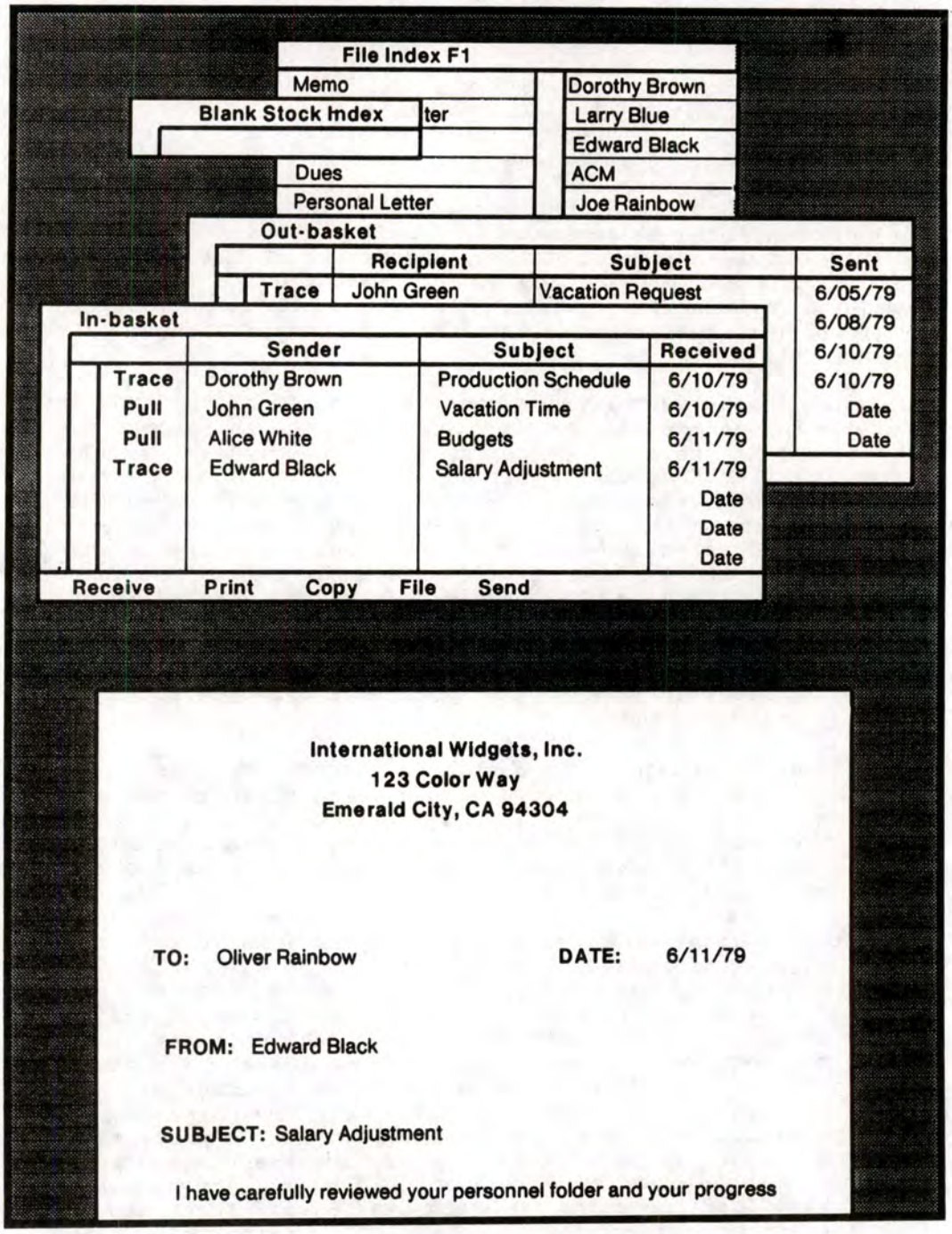

File Index F1
Memo
Blank Stock Index
ter
Dues
Personal Letter
Dorothy Brown
Larry Blue
Edward Black
ACM
Joe Rainbow
Out-basket
Recipient
Subject
Sent
Trace
John Green
Vacation Request
6/05/79
6/08/79
6/10/79
6/10/79
Date
Date
In-basket
Sender
Subject
Received
Trace
Dorothy Brown
Production Schedule
6/10/79
Pull
John Green
Vacation Time
6/10/79
Pull
Alice White
Budgets
6/11/79
Trace
Edward Black
Salary Adjustment
6/11/79
Date
Date
Date
Receive
Print
Copy
File
Send
International Widgets, Inc.
123 Color Way
Emerald City, CA 94304
TO: Oliver Rainbow
DATE: 6/11/79
FROM: Edward Black
SUBJECT: Salary Adjustment
I have carefully reviewed your personnel folder and your progress

**FIGURE 1**

# FOUR PROGRAMS FOR USE WITH LISTED OPTION AND COMMON STOCK INVESTMENT STRATEGIES

by
Alfred A. Adler Ph.D.
10360 E. Flintlock Tr.
Tucson, AZ 85715

Several years ago the author asked himself the following question. How can money be used to make more money, without becoming involved in a product or a service? By this is meant consistent, long term income, not sporatic profits interspersed with long periods of loss. The main thrust of effort in attempting to answer this question has been directed toward the security markets.

During the past several years considerable effort has gone into researching methods of tilting the odds in the investment game. Out of this has come the discovery that not only can the odds be tilted but that they can be tilted drastically, and in either direction. In particular, the strategy of hedging listed options against common stocks, when properly applied, can be proven to be more conservative and more consistently profitable than the simple buying and selling of stocks; so much so in fact that the Securities and Exchange Commisssion has recently ruled it a legal operation for trust and pension funds. The idea of an investment being more conservative and at the same time more profitable of course violates one of the widely 'known' tenets of Wall St. However in recent times much that was widely 'known' has been found to be wrong.

The only disadvantage of this strategy is its complexity. Since certain tactics, by their very nature, tend to shift the odds in your favor, while other tactics, by their nature, make it almost impossible not to lose, there is really no viable alternative to a large initial investment in self-education plus a continuing expenditure of time and effort.

The author's interest in stock market operations is primarily from the point of view of a mathematician. He firmly believes that the market is inherently unpredictable and that strategies based on hedging and the mathematics of probability are far more likely to be successful than those based on 'fundamentals', 'technical factors', or the reading of tea leaves. The ongoing study of investment strategies has included a series of computer programs which were written primarily for study purposes. The more useful of those have evolved into production programs which are used in the everyday management of investments. The programs were originally developed in PolyMorphic Basic, and have recently been revised and converted to North Star Basic. These are available from the author and a TRS-80 16K Level II version is available from Creative Computing Software.

The four programs to be presented are designed to be used in the real world, and include the effects of commissions, margin interest, and dividends, where applicable. The first presents the important indices for both opening and closing call option

transactions, including hedge ratios from zero to infinity, not inclusive. Another presents a graph or a table, as the user chooses, of profit from any combination of six basic positions: long or short a stock, long or short a call, and long or short a put. The third program enables the user to predict the future price of an option at user chosen future times based on user chosen future stock prices. The output may be displayed as either a chart giving future prices of options with three different exercise prices at three expiration dates, or a graph giving the future price of one option over a range of user chosen future stock prices. Finally, the fourth program enables the user to determine on an item by item basis the cost, current value per share, total current value, and capital gain of a portfolio consisting of long and short stock, and long and short option positions.

Program OPTION

When considering either an opening or a closing option transaction, an investor usually wishes he could readily compute many indices of risk and reward. A few of these indices are easy approximated but most require a bit of effort, and including the effects of broker's commissions, margin interest and dividends, although highly desirable, becomes very tedious. Program OPTION is designed to compute the important indices for both opening and closing call option transactions, including the effects of commissions, margin interest, and dividends. For an opening transaction these indices include, but are not limited to, option proceeds, net increased debit, lower breakeven point, upper breakeven point if a ratio write is involved, net profit in $, %, and annualized % if not called, and stock capital gain and net option profit in $, %, and annualized % if called. For a closing option transaction similar indices are computed for cases where the total position is closed out, where only the options are closed out, or in the case of a ratio write, where only the naked options are closed out. In addition, for the closing option transaction, the ratio of final profit to that contemplated at opening time is computed, along with the ratio of time actually in the position to that originally contemplated, as well as the ratio of these two ratios.

Program OPGRAPH

When dealing with stocks and put and call options, there are six basic positions one can hold: long or short a stock, a put, or a call. These positions are commonly held in combinations of two or more, and determining the stock price ranges over which a profit or loss will be incurred, and the magnitude of the profit or loss, is often tedious, particularly when broker's commissions are included. Since in these strategies, commissions wipe out a large part of the profit, and often convert a profit to a loss, it is mandatory that they be included. Program OPGRAPH presents a graph or a table, as the user chooses, of profit, including commissions, versus stock price, at option expiration.
A request for a graph will provide the user with the total net profit picture. However, a request for a table will provide not only the total net profit, but also a breakdown of the profit at each stock price into each of the six basic positions mentioned at the beginning of this discussion.

Program NEWPREM

It is well known that option premiums move up and down with stock price and decrease nonlinearly as expiration approaches. It is equally well known that option premiums are highly volatile and can change drastically as investor confidence waxes and wanes, even in the absence of significant changes in stock price or time. It is precisely for this latter reason that prediction of future option premiums is impossible. It is possible however, based on a stable reference period, to predict what an option premium should be. This future premium of course is dependent on the assumed future time and stock price at that time.
A function has been found that represents the variation of option premium with stock price at an arbitrary fixed time. Another function has been found that represents the variation of option premium with time for an arbitrary fixed stock price. Given option premium data over a user chosen reference period, program NEWPREM uses these functions to predict option premiums at user chosen future times and corresponding to user chosen stock prices. With these predicted premiums, an investor can quickly spot over or underpriced options as well as determine the probable future course of a contemplated opening option transaction.

Program PORTVAL

PORTVAL enables an investor to determine on an item by item basis the cost, current value per share, total current value, and capital gain of a portfolio consisting of long and short stock, and long and short option positions. The portfolio itself and the cost data are contained in DATA statements within the program. Current stock prices per share and current option premiums per share are entered during execution in response to prompting. Following the RUN command the user need only respond to the prompts as they arise. Output consists of a specification of the stock or option, the number of shares or options, the cost, current value per share, current dollar value of position, and current capital gain. Subtotals are provided for long stock position, long option position, short option position, and net debit. Total equity is evaluated as the sum of the current value of the long stock position and the long option position, minus the current value of the short option position and the net debit. This represents the current liquidation value of the portfolio, not including sales commissions.

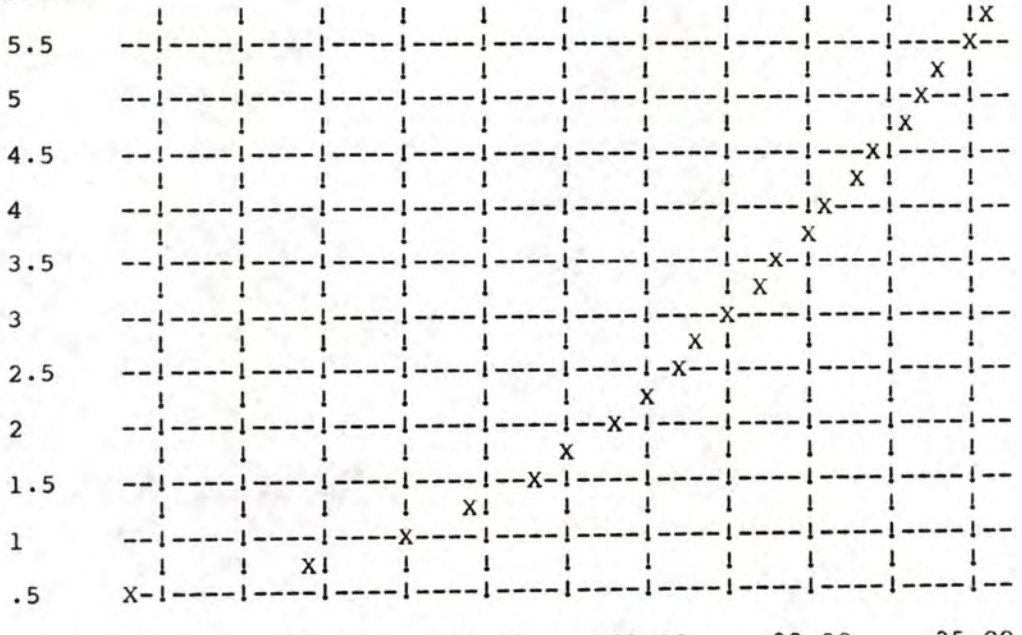
```
$$$$$$  Program  OPTION  -  by  A. A. Adler,  Ph.D.  $$$$$$$$

Do you want instructions? NO

$$$$$$$$$$$$$$$$  O P E N I N G    D A T A  $$$$$$$$$$$$$$$$

Stock Symbol, Remarks : SAF
How many shares owned? 500; and how many bought? 1000
Stock price? 22; and dividend from now to expiration? 250

Hedge ratio (how many calls sold per 100 shares of stock)? 2

Opening day no.? 200; and Expiration day no.? 300
Option striking price? 25; and Premium? 2

$$$$$$$$$$  O P E N I N G    T R A N S A C T I O N  $$$$$$$$$

TIME = .27397 YEARS TO EXPIRATION
STOCK PURCHASED =    $22223.    STOCK COMMITTED =    $33223.

OPTION PROCEEDS =     $5829.
NET PURCHASE =    $16394.    NET COMMITTED =    $27394.
LOWER BREAKEVEN =     $18.26 PER SHARE ; =    16.99 % DOWNSIDE
UPPER BREAKEVEN =     $31.18 PER SHARE ; =    41.72 % UPSIDE

                IF OPTIONS EXPIRE
INCOME =     $5697.     % =    20.80     %/YR.=     75.91
(Income = Option proceeds + Dividend - Margin interest)

          IF CALLED, STOCK GAIN/LOSS =    $3439.
PROFIT =     $9136.     % =    33.35     %/YR.=    121.74
(Profit = Stock gain + 'Income')

Do you want to compute a close? YES

$$$$$$$$$$$$$$$$  C L O S I N G    D A T A  $$$$$$$$$$$$$$$$

Closing day no.? 250
Option premium? .5; and stock price? 21
Reduced dividend? 125
Do you want to close ALL options or only UNCOVERED ones? ALL

$$$$$$$$$$  C L O S I N G    T R A N S A C T I O N  $$$$$$$$$

TIME = .13699 YEARS AFTER OPENING; OPTION PROCEEDS =     $4222.

        IF ALL OPTIONS BOUGHT BACK
INCOME =     $4156.     % =    15.17     %/YR.=    110.75
RATIO OF CLOSING INCOME TO OPENING INCOME :       .73
RATIO OF CLOSING  TIME  TO OPENING  TIME  :       .50
RATIO OF INCOME  RATIO  TO  TIME   RATIO  :      1.46

   POSITION CLOSEOUT, STOCK GAIN/LOSS =  $-2012.
PROFIT =     $2144.     % =     7.83     %/YR.=     57.14
RATIO OF CLOSING PROFIT TO OPENING PROFIT :       .23
RATIO OF PROFIT  RATIO  TO  TIME   RATIO  :       .47

Do you want to compute a close?
```

```
+++++++  Program  OPGRAPH  -  by  Alfred  A. Adler,  Ph.D  +++++

Do you want instructions? NO

************  S T O C K   &   O P T I O N   D A T A  ***********

Stock Symbol, Remarks : HBL - 5/25/78
No. shares owned? 0; and price per share? 0
No. shares bought? 0; and price per share? 0

No. calls sold? 10; exercise price? 30; and premium? 2.375
No. calls bought? 0; exercise price? 0; and premium? 0

No. puts sold? 10; exercise price? 25; and premium? 1.875
No. puts bought? 0; exercise price? 0; and premium? 0

Would you like a graph or a table, G or T? T

Readability of table is best when difference between
starting and ending stock price is evenly divisible by 10.
At what stock price should the table start? 18
At what stock price should the table end?   38

*************************  T A B L E  *************************

          <- - - - - P R O F I T     F R O M - - - - - ->
STOCK     STOCK     STOCK     CALLS     CALLS      PUTS      PUTS      TOTAL
PRICE     OWNED     BOUGHT     SOLD     BOUGHT     SOLD      BOUGHT    PROFIT

 18.00       0.        0.     2299.        0.    -5435.        0.     -3136.
 20.00       0.        0.     2299.        0.    -3435.        0.     -1136.
 22.00       0.        0.     2299.        0.    -1435.        0.       864.
 24.00       0.        0.     2299.        0.      565.        0.      2864.
 26.00       0.        0.     2299.        0.     1803.        0.      4102.
 28.00       0.        0.     2299.        0.     1803.        0.      4102.
 30.00       0.        0.     2299.        0.     1803.        0.      4102.
 32.00       0.        0.     -230.        0.     1803.        0.      1573.
 34.00       0.        0.    -2230.        0.     1803.        0.      -427.
 36.00       0.        0.    -4230.        0.     1803.        0.     -2427.
 38.00       0.        0.    -6230.        0.     1803.        0.     -4427.

Would you like a graph or a table, G or T? G

Readability of graph is best when difference between
starting and ending stock price is evenly divisible by 10,
and difference between maximum and minimum profit is evenly
divisible by 5.
At what stock price should the graph start? 18
At what stock price should the graph end?   38
What minimum profit should the graph show? -5000
What maximum profit should the graph show? 5000
How many lines would you like to skip so that the graph can
   start at the beginning of the next page? (If this RUN
   started at the top of the page, about 33 should work
   very nicely; if you printed a table first, 9 would be
   much better.) :9
```

```
**************************** G R A P H ************************

STOCK
PRICE                                PROFIT
        -5000.     -3000.     -1000.      1000.      3000.      5000.
 18      --!----!---O!----!----!----!----!----!----!----!----!--
           !    !    ! O  !    !    !    !    !    !    !    !
         --!----!----!---O!----!----!----!----!----!----!----!--
           !    !    !    ! O  !    !    !    !    !    !    !
 20      --!----!----!----!---O!----!----!----!----!----!----!--
           !    !    !    !    ! O  !    !    !    !    !    !
         --!----!----!----!----!---O!----!----!----!----!----!--
           !    !    !    !    !    ! O  !    !    !    !    !
 22      --!----!----!----!----!----!---O!----!----!----!----!--
           !    !    !    !    !    !    ! O  !    !    !    !
         --!----!----!----!----!----!----!---O!----!----!----!--
           !    !    !    !    !    !    !    ! O  !    !    !
 24      --!----!----!----!----!----!----!----!---O!----!----!--
           !    !    !    !    !    !    !    !    ! O  !    !
         --!----!----!----!----!----!----!----!----!----!O---!--
           !    !    !    !    !    !    !    !    !    !O   !
 26      --!----!----!----!----!----!----!----!----!----!O---!--
           !    !    !    !    !    !    !    !    !    !O   !
         --!----!----!----!----!----!----!----!----!----!O---!--
           !    !    !    !    !    !    !    !    !    !O   !
 28      --!----!----!----!----!----!----!----!----!----!O---!--
           !    !    !    !    !    !    !    !    !    !O   !
         --!----!----!----!----!----!----!----!----!----!O---!--
           !    !    !    !    !    !    !    !    !    !O   !
 30      --!----!----!----!----!----!----!----!----!----!O---!--
           !    !    !    !    !    !    !    !    O    !    !
         --!----!----!----!----!----!----!----!--O-!----!----!--
           !    !    !    !    !    !    !    O    !    !    !
 32      --!----!----!----!----!----!----!--O-!----!----!----!--
           !    !    !    !    !    !    O    !    !    !    !
         --!----!----!----!----!----!--O-!----!----!----!----!--
           !    !    !    !    !    O    !    !    !    !    !
 34      --!----!----!----!----!--O-!----!----!----!----!----!--
           !    !    !    !    O    !    !    !    !    !    !
         --!----!----!----!--O-!----!----!----!----!----!----!--
           !    !    !    O    !    !    !    !    !    !    !
 36      --!----!----!--O-!----!----!----!----!----!----!----!--
           !    !    O    !    !    !    !    !    !    !    !
         --!----!--O-!----!----!----!----!----!----!----!----!--
           !    O    !    !    !    !    !    !    !    !    !
 38      --!--O-!----!----!----!----!----!----!----!----!----!--

Would you like a graph or a table, G or T?
STOP IN LINE 50
```

```
<<<<<  Program  NEWPREM  -  by  Alfred  A. Adler  Ph.D.  >>>>

If you want instructions, LOAD Program INSTRUCT.

Do you want to see the normalized premiums? : YES
Do you want screen or printer type output, S or P? P

###################  INTRODUCTORY  DATA  ####################

Stock Symbol, Remarks : NWA
Do you want predictions of puts or calls? CALLS
Can you state the time function constants? NO

How many days will you use for your data base? 2
And how many exercise prices per day? 2
Choose exercise price No. 1 : 25
Choose exercise price No. 2 : 30

#############  THE DATA BASE WILL NOW BE INPUT  #############

FOR DAY 1 OF DATA BASE,
-------------------------
Give day no. : 255; and stock price : 29
List the 3 expiration day nos. (in chronological order) for
   options available on this day : 321,48,139
Give the 3 premiums corresponding to these expiration day nos.
   for an exercise price of 25 : 4.375,5.125,7.125
Give the 3 premiums corresponding to these expiration day nos.
   for an exercise price of 30 : 1.75,2.8125,3.875

              TABLE OF NORMALIZED OPTION PREMIUMS

EXERCISE                DAYS TO EXPIRATION
 PRICE          66              158             249
 25         1.2808689       2.4011716       4.7186465
 30         2.1937411       3.2745467       4.3463346

FOR DAY 2 OF DATA BASE,
-------------------------
Give day no. : 248; and stock price : 25.5
List the 3 expiration day nos. (in chronological order) for
   options available on this day : 321,48,139
Give the 3 premiums corresponding to these expiration day nos.
   for an exercise price of 25 : 2.25,2.9375,3.875
Give the 3 premiums corresponding to these expiration day nos.
   for an exercise price of 30 : .6875,1.1875,2.125

              TABLE OF NORMALIZED OPTION PREMIUMS

EXERCISE                DAYS TO EXPIRATION
 PRICE          73              165             256
 25         1.9843135       2.6758468       3.616369
 30         1.8884932       2.5988278       3.7520827

##############  THE DATA BASE IS NOW COMPLETE  ##############

THE TIME FUNCTION CONSTANTS ARE :
A =  47.60297
B =  2.6467181
```

```
DAYS TO EXPIRATION          ADJUSTED NORMALIZED PREMIUMS

      5                          .10442933
      10                         .20767304
      15                         .30977118
      20                         .41076078
      25                         .51067736
      50                         .99527864
      100                        1.8999948
      150                        2.7351262
      200                        3.5146189
      250                        4.2483028
      300                        4.9434146
      350                        5.6054657

#####  ENTER PARAMETERS FOR PREDICTED OPTION PREMIUMS  ######

***  Do you want a table of premiums for 9 options on a  ***
     chosen day for a chosen stock price, or do you want
     a graph of the premium for 1 option on a chosen day
     over a range of stock prices : T or G? G

Choose the exercise price desired : 30
Choose the expiration month : AUG
What is the expiration day no. for this month? 229

What is the day no. for which premiums are to be predicted? 139

Readability of graph is best when difference between
   starting and ending stock price is evenly divisible by 5.
At what stock price should the graph start? 25
At what stock price should the graph end?   35
How many lines would you like to skip so that the graph can
   start at the beginning of the next page? : 0


###########  GRAPH FOR AUG 30 CALLS ON DAY NO. 139  ###########

PREMIUM
```



# THE MICROCOMPUTER MARKET AND USERS IN JAPAN

Seiichiro Yahagi
Managing Director of NIPPON TIME SHARE CO., LTD.
17th Mori bldg., 1-26-5, Toranomon,
Minato-ku, Tokyo, Japan

## 1. Microcomputer Market in Japan

Microcomputer market is one of the growing markets in Japan. In spite of low growth economy in tne last several years after the oil shock, microcomputer market in Japan has been grown very rapidly.

Major Japanese microcomputer manufactures are as follows:

* Adtech System Science Co., Ltd.
* Fujitsu Ltd.
* Hitachi Ltd.
* Matsushita Electric Industrial Co., Ltd.
* Mitsubishi Elelctric Co., Ltd.
* Nippon Electric Co., Ltd.
* Sharp Co., Ltd.
* Sord Computer System Co., Ltd.

Nippon Electric Co., Ltd.(NEC) is the pioneer of this market. And recently Sharp Co., Ltd. entered to this market, and its new machine became a best seller because of its cost/performance.

There are many small companies which specialize in the specific fields in the Japanese microcomputer market.

Imported microcomputer such as PET, TRS, APPLE II, COMPUCOLOR and IMSAI 8080 are also popular in Japan, and are accepted by Japanese microcomputer hobbiest.

There are several micrcomputer magazines in Japan. Their names are "MICON", "I/O", "RAM" and "INTERFACE". Beside these magazines, some other computer magazines such as "COMPUTOPIA" provide articles relating microcomputer.

Microcomputer shops also enjoy the fruit of the growing market. In the beginning, the location of these microcomputer shops were restricted in two big cities, Tokyo and Osaka. But recently microcomputer shops have been increased in local cities.

The first international microcomputer conference in Japan was held in 1978. It is called as "IMAC78". "IMAC 80" will be held in this year. Other microcomputer exhibitions at Ryutsu Center and department stores are held in several times in a year.

In conclusion, Japanese microcomputer market just started to take-off era. Business use of microcomputer were introduced recently. Japanese people understand slowly difference between the myth and reality of microcomputer applications.

## 2. Microcomputer Users in Japan

Microcomputer users in Japan are remarkably increasing in the past few years. Recently, Japan Microcomputer Club summarized the result of questionaires. This club is the largest microcomputer club in Japan. The number of club members is 2,212 including 649 students as of Sept. 30, 1979. The following facts and figures are summarized the above-mentioned questionaires by Japan Microcomputer Club.

Figure 2 shows the proportion of members by age. The age of club member is widely spread out from 5 to 74. The proffessions of club members are varied from computer specialist to students shown in figure 3.

The types of occupations such as production are shown in figure 4. Engineer is the biggest occupation, and its share is the one third. The use purposes of microcomputer are summarized in figure 5. Hobby and game is 37 % which is the largest part among various purposes.

Figure 6 shows microcomputer models which are used by club members. The most popular microcomputer is TK-80, manufactured by Nippon Elctric Co.,Ltd. and it is noteworthy that 70 sets of microcomputers are of their own making.

The above-mentioned facts and figures are small sample of microcomputer users in Japan. Therefore these figures may have some bias. But I believe that the answers for questionaires can provide useful information concerning the situation of microcomputer users in Japan.

Figure 1. Microcomputer manufactures & models

Adtech System Science Co.,Ltd.: "ORANGE"

Fujitsu Ltd.: "LKIT-8", "LKIT-16"

Hitachi Ltd.: "H 68"

Matsushita Electric Industrial Co., Ltd.: "MY BRAIN"

Mitsubishi Electric Co., Ltd.: "PCA 0801"

Nippon Electric Co., Ltd.: "PC8001", "TK-80BS"

Sharp Co., Ltd.: "MZ-80K"

Sord Computer Systems Co., Ltd.: "M100"

Figure 2. Age Distribution

| | |
|---|---|
| 5 - 9 years old | 0.1% |
| 10 - 14 | 1.2 |
| 15 - 19 | 12.7 |
| 20 - 24 | 22.6 |
| 25 - 29 | 17.0 |
| 30 - 34 | 14.6 |
| 35 - 39 | 11.1 |
| 40 - 44 | 8.4 |
| 45 - 49 | 4.2 |
| 50 - 54 | 3.6 |
| 55 - 59 | 2.5 |
| 60 - 64 | 1.4 |
| 65 - 69 | 0.5 |
| 70 - 74 | 0.1 |
| Total: | 100.0% |

Figure 3. Type of Industries

| | |
|---|---|
| Student | 32.4% |
| Electronics | 17.5 |
| Education | 9.0 |
| Machinery | 6.3 |
| Computer | 5.9 |
| Petrochemicals | 3.6 |
| Government | 3.3 |
| Hospital | 2.6 |
| Service | 2.5 |
| Proffessional | 1.4 |
| Transportation communication | 0.9 |
| Mass communication | 0.9 |
| Banking & Security | 0.8 |
| Non-steel | 0.7 |
| Steel | 0.1 |
| Others | 12.1 |
| Total: | 100.0% |

Figure 4. Type of Occupations

| | |
|---|---|
| Enginner | 33.0% |
| Reserch & Development staff | 16.6 |
| Teacher | 11.3 |
| Manager | 8.0 |
| Salesman | 5.7 |
| Factory worker | 4.8 |
| Programmer | 4.7 |
| Medical Doctor | 3.9 |
| Office Clerk | 3.6 |
| Planner | 3.3 |
| Others | 5.1 |
| Total: | 100.0% |

Figure 5. Purpose of Using Microcomputer

| | |
|---|---|
| Hobby & game | 37.0% |
| Business use | 17.0 |
| For the future use | 5.5 |
| Education | 5.0 |
| To learn microcomputer | 4.0 |
| Private use | 2.9 |
| Amateuer wireless | 2.3 |
| Research & development | 2.3 |
| Statistics | 1.2 |
| Measurement & control | 1.1 |
| Scientific calculation | 0.8 |
| Data Base | 0.4 |
| No answer | 13.2 |
| Others | 7.3 |
| Total: | 100.0 |

Figure 6. Owner 's Microcomputer Models

| | | | |
|---|---|---|---|
| TK-80 | 90 (8)* | CROMEMCO | 5 |
| TK-80+BS | 189 (9)* | IMSAI | 4 |
| COMPO BS | 24 (1)* | SWTPC | 4 |
| PC-8001 | 4(193)* | AIDACS | 4 |
| EX-5 | 14 | PFC-15 | 3 |
| EX-80 | 25 (7)* | MDS | 3 |
| EX-80+BS | 22 (2)* | AIM 65 | 2 |
| H 68/TR | 53 (5)* | MARVEL 2000 | 2 |
| Basic Master | 17 (3)* | COMPUCOLOR | 2 |
| MZ-80K | 95(40)* | MCZ-80 | 2 |
| LKIT-8 | 9 (3)* | PDA-80 | 2 |
| LKIT-16 | 62 (3)* | PDC-80 | 2 |
| PET | 41(18)* | M& BRAIN | 2 |
| APPLE II | 28(25)* | HAND-MADE | |
| TRS-80 | 36(21)* | 8080 | 34(8)* |
| SMB-80T | 4 (1)* | 6800 | 22(6)* |
| M-100 | 8(13)* | 6500 | 2 |
| M-120 | 1 | Others | 12(5)* |
| M-180 | 2 | ORANGE | (4)* |
| M-220 | 7 | ATARI 800 | (5)* |
| COMKIT | 6 | TI 99/4 | (3)* |
| MEK 6800 DII | 11 | MC 6800 | (5)* |
| SDK-80 | 10 | RMC-1007 | 4 |
| SDK-85 | 12 | | |

NOTE: ()*...No. of purchase under consideration

# TURNKEY OR TURKEY ?

Thomas P. Bun, COMPUMAX, INC., 505 Hamilton,Palo Alto,CA 94301 (415) 321-2881
and
Paul J. Terrell, EXIDY, INC., 390 Java, Sunnyvale, CA 94086 (408) 734-9410

## Abstract

Why should a small business use a computer? What are the alternatives? How can the pains and the costs associated with computerization be minimized?

These issues are dealt with in simple terms, oriented towards the prospective novice computer user.

An innovative approach is described, based on a set of computer programs that come in a form completely ready to use, yet can be understood and set up rapidly, with minimal restrictions and great ease of change and extentions of the particular requirements of an individual business.

## Reasons for the current interest in computers

With the rapidly rising number of microcomputers made available, at ever decreasing cost, it has become apparent to the small businessman that the computer is the equalizer that will allow even the smallest of businesses to compete effectively with the horsepower of large corporations.

The small business computer is an electronic filing cabinet, an appointments calendar, a report generator and a sophisticated bookkeeper, to name only a few of the many applications within its capability.

A small business computer can reduce the time and expense of manual paperwork, eliminate the many sources of error and overlaps present in manual operation and at the same time increase the efficiency and accuracy of the existing staff.

Both general applications - like the complete bookkeeping - and specific tasks, like word processing, can be accomplished by the same equipment equally well.

## What size of business can afford a computer?

The same electronic technology that put the calculator in every shirt pocket, T.V. games and personal computers in every home, is now putting a small business computer on every desk.

With the cost of a microcomputer including disk storage and printer under $4,000 - and available on a lease term under $200 per month - today's businesses with annual revenues as low as $100,000 and a single proprietor, will undoubtedly benefit from a small easy-to-use computer. In fact,to stay competitive in business, he cannot afford not to have the computer in his business.

## The hidden cost of computer ownership

If the prospective user of a small business computer simply walks out into the marketplace and compares the price tag on the many available machines to make a decision, he is committing a grave mistake.

He may select the very best hardware ever made at the lowest price in history, and still make the worst possible buy.

The true question to be examined has very little to do with hardware sophistication. The question really is quite different. How will the computer be used in the business?

The name of the vast domain of procedures, programs and documentation required for the purpose of using a computer is SOFTWARE. In the old times when computers were first born 30 years ago, "SOFTWARE" represented about 5 % of the investment; the rest, 95 %, went towards the "HARDWARE", that is the equipment itself.

Since these old times, often referred to as "First Generation" times, this situation has changed radically. It is not difficult to find computer users today, who have just about inverted those two percentages. If a computer is acquired with little or no software, it is very likely that the total cost of software, by the time something useful is developed,will be a multiple of the original cost of the hardware.

## What are the alternatives?

The two extreme cases that can be observed in todays market are quite easily identified. At one end is the brand new,shining microcomputer that has just been brought out. The only programs readily available are a few demonstration programs, a couple of games. The user manual of the computer itself consists of mimeograph sheets in a 3-ring binder, entitled

"PRELIMINARY RELEASE".

There is a prepaid post card to be returned, with the promise of substitution for the final

document, as soon as it is available. All actual user programs will have to be written from scratch.

At the other end of this rainbow, a small business computer is offered with the claim that it is READY TO DO ALL YOUR APPLICATIONS. This approach is often described as the

"TURNKEY SYSTEM".

What is meant by the phrase is that you get a machine that is ready to process your business applications, whatever they may be!

The claim is, obviously, somewhat difficult to understand, since each business has quite different requirements.

For the matter of clarity and objective analysis, let us state that within the turnkey system approach itself, there is a wide variety of scopes. Some of them are built around old and fairly proven programs, that is, programs run many times and probably pretty useful in many situations.

Others are characterized by elaborate structures and procedures. Look out for particularly restrictive assumptions. They may assume that your accounting is on the "accrual basis". If you are on a "cash basis", you simply cannot use their system at all.

Often the programs are really very good and have just about everything that you may want, all the reports, all the data are taken care of. But when you want to make any change at all, you discover that the programs are written in

"ASSEMBLER LANGUAGE"

that is, the machine language of your computer. Thisisquite often the case with large, professional programs. It turns out, however, that to make a change,in this case, is at least very time consuming and costly, since you need a specialist to do it. It may be outright impossible, since you will sometimes find that

"THE SOURCE CODE IS NOT RELEASED".

This means that the author has locked away the code of the program and is not making it available to the users. In this case, to make any little change at all may be just as costly as to have to produce a new program from scratch.

The Ideal Approach

It is now possible to stay clear from both of these extreme cases. "CANNED PROGRAMS" are available off the shelf that are based on SIMPLICITY. The concepts of small business bookkeeping should be boiled down to the basic essentials. Programs should be produced specially for the novice, first time computer user.

The programs should be CONVERSATIONAL. That is, a "MENU" of alternative steps should be shown on the screen; as the user makes his selection, other levels of selection should come up, until a working routine is reached.

The programs must be INTERACTIVE. This means that when you enter an INVENTORY TRANSACTION, the information is made available to the GENERAL LEDGER: you do not need to enter the same information twice!

The programs need to be documented in several ways:

USER MANUALS should have a chapter in plain ENGLISH describing how the packages are to be set up and run in the first place. Actual sample runs should be reproduced in abundance.

The source code must be reproduced in its entirety in the User Manuals. There should be FLOWCHARTS and BLOCK DIAGRAMS showing the basic elements of program logic and the data structures employed in the data files. Last but not least, it is essential that many REMARK statements be in the program code itself, showing where a particular routine is found; where a particular data table (e.g. federal and state withholding tables in the PAYROLL program) is stored.

Effort should be made to avoid going to extremes of professionalism, where a program would become so compact, that it would be very difficult to change it. On the contrary, the purpose must be one of utter clarity and simplicity.

By the way,the Ideal Approach is the CompuMax Approach. The original versions of the software are now available for eight different families of computers (TRS-80, APPLE, PET, CROMEMCO, MICROPOLIS, MICROSOFT CP/M, CBASIC2 CP/M and DYNABYTE), all of them in this same vein of simplicity, and will print payable checks AND paychecks, W-2 forms and 941 amounts; aged trial balances of accounts receivable AND management science routines like optimize economic order quantities across an entire inventory. At the same time, TOTAL FLEXIBILITY and EASE of CHANGE are maintained.

The points made in the preceding chapter are illustrated in the following sample runs.

The computer runs were produced by the MAXILEDGER program, the extended version of the COMPUMAX job stream.

MAXILEDGER is currently available in the MICROPOLIS BASIC version. The program can be run on any 48k microcomputer interfaced with a MICROPOLIS 1053 Mod II dual floppy disk drive, such as the 48k model of the SORCERER by EXIDY, with the VIDEODISK.

```
PLOADG"MAXILEDGER"

WELCOME TO MAXILEDGER  SERIAL # 670

MAXILEDGER PROGRAM SELECTION
  1 - CHART OF ACCOUNTS DATA ENTRY
  2 - JOURNAL DATA ENTRY
  3 - SUBTOTALS UPDATE AND FILE LISTING
  4 - TRIAL BALANCE & FAST POSTING
  5 - TRIAL BALANCE & SLOW POSTING
  6 - REPORT OF CHANGES IN POSITION
  7 - PROFIT & LOSS STATEMENT
  8 - BALANCE SHEET
  9 - FILE BACKUP/RECOVERY   OR  0 - QUIT
ENTER THE NUMBER WANTED (0-9)? 1

1.  CHART OF ACCOUNTS PROGRAM
----------------------------

YOU MAY NOW
  11 - CREATE NEW CHART OF ACCOUNTS RECORDS
  12 - UPDATE EXISTING CHART OF ACCOUNTS RECORDS
  13 - CLEAR CURRENT FIELDS
  14 - CLEAR CURRENT AND MTD FIELDS
  15 - CLEAR CURRENT, MTD AND YTD FIELDS
  16 - DO THE ANNUAL RESTART OF THE CHART OF ACCOUNTS FILE
  17 - PURGE THE CHART OF ACCOUNTS FILE
  18 - CONVERT MICROLEDGER CHART
  19 - RUN ANOTHER PROGRAM
WHICH ONE DO YOU WANT (11-19)? 11
ENTER DATE (MMDDYY) ? 123178
ENTER TITLE FOR REPORTS? MICROMAX INC.
'CHART' FILE IS CREATED.

DO YOU WANT ACCOUNT EXPLANATION  Y OR N? Y

YOU WILL BE PROMPTED TO ENTER THE FOLLOWING DATA, FOR EACH RECORD:
    ACCOUNT NUMBER
    ACCOUNT NAME (MAX. 29 CHARACTERS)
    STARTING AMOUNTS FOR CURRENT, M-T-D, Y-T-D FIELDS
         (ALL OF WHICH MAY BE 0)

ACCOUNT NUMBERS MUST FOLLOW THE RULES:

    1001 - 1999 FOR CURRENT ASSETS
    2001 - 2999 FOR NON-CURRENT ASSETS
    3001 - 3999 FOR CURRENT LIABILITIES
    4001 - 4999 FOR LONG-TERM LIABILITIES
    5001 - 5999 FOR OWNERS EQUITIES
    6001 - MUST BE RETAINED EARNINGS
    6002 - 6999 FOR CUSTOM PROGRAMMING
    7001 - 7999 FOR REVENUE ACCOUNTS
    8001 - 8999 FOR DIRECT EXPENSE ACCOUNTS
    9001 - 9999 FOR G. & A.  EXPENSE ACCOUNTS.

LAST DIGIT OF A DETAIL ACCOUNT # MUST BE 1 THROUGH 9.
LAST DIGIT OF A SUBTOTAL ACCOUNT # MUST BE 0.
```

*MAIN PROGRAM SELECTION - often called "MENU" -*

*Having selected Program 1, you now see the next level of selection, that is, options in Program 1.*

*You now select option 11, the Chart data entry routine.*

COMPUMAX ASSOCIATES, INC.

*Chart of Accounts data entry using option 11.*

```
ACCOUNT #? 1011
ACCOUNT NAME (MAX. 29 CHARS. )? SECURITY PACIFIC BANK
CURRENT AMOUNT? 0
M. -T. -D. AMOUNT? 7203
Y. -T. -D. AMOUNT? 8723

1 - ACCOUNT # 1011
2 - ACC'T NAME: SECURITY PACIFIC BANK
3 - CURRENT AMOUNT 0
4 - M. T. D.  AMOUNT  7203
5 - Y. T. D.  AMOUNT  8723

IF RECORD IS CORRECT THEN ENTER Y? Y
NEW RECORD # 1  CREATED IN 'CHART'.
     MORE, Y OR N ? Y

ACCOUNT #? 1012
ACCOUNT NAME (MAX. 29 CHARS. )? UNIN BANKL
CURRENT AMOUNT? 0
M. -T. -D. AMOUNT? -6340
Y. -T. -D. AMOUNT? 13826

1 - ACCOUNT # 1012
2 - ACC'T NAME: UNIN BANKL
3 - CURRENT AMOUNT 0
4 - M. T. D.  AMOUNT -6340
5 - Y. T. D.  AMOUNT  13826

IF RECORD IS CORRECT THEN ENTER Y? N
WHICH ONE DO YOU WANT TO CHANGE, 1-5 (0 IF NO MORE CHANGE)? 2
NEW DESCRIPTION? UNION BANK
WHICH ONE DO YOU WANT TO CHANGE, 1-5 (0 IF NO MORE CHANGE)? 0
NEW RECORD # 2  CREATED IN 'CHART'.
     MORE, Y OR N ? Y

ACCOUNT #? 1013
ACCOUNT NAME (MAX. 29 CHARS. )? UNITED CALIFORNIA BANK
CURRENT AMOUNT? 0
M. -T. -D. AMOUNT? 0
Y. -T. -D. AMOUNT? 4079

1 - ACCOUNT # 1013
2 - ACC'T NAME: UNITED CALIFORNIA BANK
3 - CURRENT AMOUNT 0
4 - M. T. D.  AMOUNT  0
5 - Y. T. D.  AMOUNT  4079

IF RECORD IS CORRECT THEN ENTER Y? Y
NEW RECORD # 3  CREATED IN 'CHART'.
     MORE, Y OR N ? N
  THERE ARE NOW 3  ACCOUNTS IN 'CHART'.

YOU MAY NOW
  11 - CREATE NEW CHART OF ACCOUNTS RECORDS
  12 - UPDATE EXISTING CHART OF ACCOUNTS RECORDS
```

etc.

COMPUMAX ASSOCIATES, INC.

*Next, you select Program 2 from the main "MENU". This is an example of journal data entry using option 21.*

```
2. JOURNAL DATA ENTRY PROGRAM
-----------------------------

YOU MAY NOW
  21 - CREATE SETS OF DOUBLE ENTRIES
  22 - ADD SINGLE JOURNAL ENTRIES
  23 - UPDATE EXISTING JOURNAL RECORDS
  24 - PURGE JOURNAL FILE
  25 - RULES OF THE ACCOUNTING EQUATION
  26 - RUN ANOTHER PROGRAM
WHICH ONE DO YOU WANT (21-25)? 21
START DATE FOR PERIOD (MMDDYY)? 010179
END DATE FOR PERIOD (MMDDYY)  ? 013179
'JOUFIL' IS CREATED.


DOUBLE ENTRIES # 1  AND 2

DATE (MMDDYY)? 010179
DESCRIPTION (MAX. 30 CHARS. )? VECTOR SALE
AMOUNT (NO MINUS SIGN, PLEASE)? 700
DR ACC'T #? 1011
CR ACC'T #? 7011
ANOTHER DOUBLE ENTRY  Y OR N? Y


DOUBLE ENTRIES # 3  AND 4

DATE (MMDDYY)? 011079
DESCRIPTION (MAX. 30 CHARS. )? BYTE SHOP HAYWARD
AMOUNT (NO MINUS SIGN, PLEASE)? 3500
DR ACC'T #? 1013
CR ACC'T #? 7011
ANOTHER DOUBLE ENTRY  Y OR N? Y


DOUBLE ENTRIES # 5  AND 6

DATE (MMDDYY)? 010279
DESCRIPTION (MAX. 30 CHARS. )? PAPER PURCHASE
AMOUNT (NO MINUS SIGN, PLEASE)? 240.86
DR ACC'T #? 8011
CR ACC'T #? 1012
ANOTHER DOUBLE ENTRY  Y OR N? Y


DOUBLE ENTRIES # 7  AND 8

DATE (MMDDYY)?
DESCRIPTION (MAX. 30 CHARS. )? STOCK ISSUE
AMOUNT (NO MINUS SIGN, PLEASE)? 10000
DR ACC'T #? 1013
CR ACC'T #? 5101
ANOTHER DOUBLE ENTRY  Y OR N? etc.
```

```
TRIAL BALANCE FOR PERIOD STARTING 01-01-79 TO 01-31-79
------------------------------------------------------

EXPENSES        240.86      REVENUES      6060.00

INCOME         5819.14

TO ASSETS      15685.64     TO LIAB.      9866.50

                            TO R. E.      5819.14

ASSETS BALANCE LIAB. +O. E.

DO YOU WANT TO POST, Y OR N? Y
```

*Since thre Trial Balance ended with a satisfactory balance, you answer Y and post your transactions.*



```
GENERAL LEDGER RUN - MICROMAX INC.
----------------------------------

ACCOUNT                                  OPEN BAL.      CURR. AMT.      ENDING BAL.

  1011    SECURITY PACIFIC NATIONAL BANK
                                          8,723.00
01-01-79 VECTOR SALE                                       700.00
01-15-79 E. P. R. I. CONTRACT                            2,000.00
01-20-79 MORTGAGE PMT.                                    -133.50
                                                                        11,289.50

  1012    UNION BANK
                                         13,826.00
01-02-79 PAPER PURCHASE                                   -240.86
01-02-79 CHANGE                                            -13.50
01-02-79 PRINTER                                        -1,200.00
                                                                        12,371.64

  1013    UNITED CALIFORNIA BANK
                                          4,079.00
01-10-79 BYTE SHOP HAYWARD                               3,500.00
01-02-79 STOCK ISSUE                                    10,000.00
                                                                        17,579.00

  7011    MICROPOLIS SALES
                                         18,491.00
01-10-79 BYTE SHOP HAYWARD                               3,500.00
08-07-79 SALES RETURN                                     -140.00
01-01-79 VECTOR SALE                                       700.00
                                                                        22,551.00

  7201    SERVICE CONTRACTS
                                         17,728.00
01-15-79 E. P. R. I. CONTRACT                            2,000.00
                                                                        19,728.00
```



```
  8011    STATIONARY STOCK
                                             0.
01-02-79 PAPER PURCHASE                                   240.86
                                                                           240.86

 18  JOURNAL TRANSACTIONS POSTED TO CHART OF ACCOUNTS FILE.
```

CHANGES IN FINANCIAL POSITION - 12-31-78 THROUGH 01-31-79

MICROMAX INC.

| | | | |
|---|---|---|---|
| CURRENT ASSETS | 39,822.00 | 54,207.64 | 36.12 % |
| NON-CURRENT ASSETS | 10,240.00 | 11,440.00 | 11.71 % |
| CURRENT LIABILITIES | 576.00 | 576.00 | 0. % |
| LONG-TERM LIABILITIES | 6,340.00 | 14,217.50 | 124.25 % |
| OWNERS EQUITIES | 20,000.00 | 30,000.00 | 50.00 % |
| RETAINED EARNINGS | 23,046.00 | 28,865.14 | 25.25 % |
| REVENUES | 75,254.00 | 81,314.00 | 8.05 % |
| DIRECT EXPENSES | 35,730.00 | 35,970.86 | 0.67 % |
| G. & A. EXPENSES | 0. | 0. | |

PROFIT & LOSS STATEMENT - MICROMAX INC.

FOR PERIOD ENDING 1 - 31 - 79

| | CURRENT | MONTH-TO-DATE | YEAR-TO-DATE |
|---|---|---|---|
| REVENUES | | | |
| MICROPOLIS SALES | 4,060.00 | 7,788.00 | 22,551.00 |
| | 66.99 % | 26.23 % | 27.73 % |
| CROMEMCO SALE | 0.00 | 11,035.00 | 23,035.00 |
| | 0.00 % | 37.17 % | 28.32 % |
| MERCHANDISE TOTAL | 4,060.00 | 18,823.00 | 45,586.00 |
| | 66.99 % | 63.40 % | 56.06 % |
| SERVICE CONTRACTS | 2,000.00 | 10,864.00 | 19,728.00 |
| | 33.00 % | 36.59 % | 24.26 % |
| CONSULTING | 0.00 | 0.00 | 16,000.00 |
| | 0.00 % | 0.00 % | 19.67 % |
| SERVICE TOTAL | 2,000.00 | 10,864.00 | 35,728.00 |
| | 33.00 % | 36.59 % | 43.93 % |
| TOTAL REVENUES | 6,060.00 | 29,687.00 | 81,314.00 |
| | 100.00 % | 100.00 % | 100.00 % |
| EXPENSES | | | |
| STATIONARY STOCK | 240.86 | 240.86 | 240.86 |
| | 100.00 % | 1.33 % | 0.66 % |
| MATERIALS | 0.00 | 11,525.00 | 23,050.00 |
| | 0.00 % | 63.65 % | 64.07 % |
| STAFF WAGES | 0.00 | 6,340.00 | 12,680.00 |
| | 0.00 % | 35.01 % | 35.25 % |
| TOTAL DIRECT EXP. | 240.86 | 18,105.86 | 35,970.86 |
| | 100.00 % | 100.00 % | 100.00 % |
| AMORTIZATION EXPENSE | 0.00 | 0.00 | 0.00 |
| | 0.00 % | 0.00 % | 0.00 % |
| TOTAL G. & A. EXP. | 0.00 | 0.00 | 0.00 |
| | 0.00 % | 0.00 % | 0.00 % |
| TOTAL EXPENSES | 240.86 | 18,105.86 | 35,970.86 |
| | 100.00 % | 100.00 % | 100.00 % |
| INCOME | 5,819.14 | 11,581.14 | 45,343.14 |
| | 96.02 % | 39.01 % | 55.76 % |

COMPUMAX ASSOCIATES, INC.

*Finally, you run Program 8 and get your updated Balance Sheet.*

PAGE 1

BALANCE SHEET - MICROMAX INC.
DATE OF 1 - 31 - 79

| | CURRENT | MONTH-TO-DATE | YEAR-TO-DATE |
|---|---|---|---|
| CURRENT ASSETS | | | |
| SECURITY PACIFIC NATIONAL BANK | 2,566.50 | 9,769.50 | 11,289.50 |
| UNION BANK | ( 1,454.36)( | 7,794.36) | 12,371.64 |
| UNITED CALIFORNIA BANK | 13,500.00 | 13,500.00 | 17,579.00 |
| BANKS TOTAL | 14,612.14 | 15,475.14 | 41,240.14 |
| PETTY CASH | 13.50 | 1,048.50 | 3,118.50 |
| ACCOUNTS RECEIVABLE | ( 140.00) | 3,724.00 | 9,849.00 |
| TOTAL CURRENT ASSETS | 14,485.64 | 20,247.64 | 54,207.64 |
| NON-CURRENT ASSETS | | | |
| EQUIPMENT | 1,200.00 | 1,200.00 | 13,200.00 |
| ACCUM. DEPREC. -EQUIPMENT | 0. | 0. | ( 1,760.00) |
| TOT. NON-CURR. ASSETS | 1,200.00 | 1,200.00 | 11,440.00 |
| TOTAL ASSETS | 15,685.64 | 21,447.64 | 65,647.64 |
| CURRENT LIABILITIES | | | |
| CURRENT PAYMENTS ON LOANS | 0. | 0. | 576.00 |
| TOTAL CURRENT LIAB. | 0. | 0. | 576.00 |
| LONG-TERM LIABILITIES | | | |
| MORTGAGE OUTSTANDING | ( 133.50)( | 133.50) | 6,206.50 |
| TOT. LONG-TERM LIAB. | -133.50 | -133.50 | 6,206.50 |
| OWNERS EQUITIES | | | |
| COMMON STOCK | 10,000.00 | 10,000.00 | 30,000.00 |
| RETAINED EARNINGS | 5,819.14 | 11,581.14 | 28,865.14 |
| TOTAL OWNERS EQUIT. | 15,819.14 | 21,581.14 | 58,865.14 |
| TOTAL LIAB. & O. E. | 15,685.64 | 21,447.64 | 65,647.64 |

Acknowledgments

The following persons rendered significant help in the development of the concepts of "The Ideal Approach", with long years of inspiration and patient support to this work:

Dave Pava and Bob Miller,
BYTE INDUSTRIES, INC. - Hayward, CA

Jeff Boetticher
COMMODORE BUSINESS MACHINES
Sunnyvale, CA

Mike Lipschutz
BYTE SHOP of HAYWARD, CA

Boyd Wilson
BYTE SHOP of MOUNTAIN VIEW, CA

The authors wish to express their special gratitude for the help of these people. Without their guidance the results of the 9 man years spent on this project would not be the same.

About the Authors

Thomas P. Bun is President of Compumax, Inc. a Palo Alto, CA software house.The various prior responsibilities held by Mr. Bun in the computer industry include the position of senior systems analyst with Stanford Research Institute, applications designer with Rockwell International, systems manager with Light S.A., director of Reduto, and managment analyst with I.S.I.Corporation. He is a M.S. from Stanford University and an M.B.A. from the University of Santa Clara.

Paul J. Terrell directed the operations and closed the doors of a 3 million dollar turn-key computer system business in 1973.

Mr. Terrell started a sales company in 1974 and became completely familiar with the value of a business plan and accounting practices or the lack of them.

Mr. Terrell was a founder and president of Byte Shop Computer Stores, which had to address the problems associated with selling business computers over-the-counter to an audience of first time users.

# THE PERFORMING MUSICIAN AND THE PERSONAL COMPUTER

R. J. Higgins, R. K. Goodall, and R. Vedanayagam
Physics Department, University of Oregon
Eugene, OR 97403

The resources for this project were supported in part by grants from the National Science Foundation (Instructional Scientific Equipment Program), the Tektronix Foundation, and the College of Arts and Sciences of the University of Oregon.

ABSTRACT

This paper surveys recent developments in both computer music and electronic music. A number of recent advances are described which are familiar to synthesizer music performers but unfamiliar to personal computer users. A personal computer is a resource for music experiments, provided that several gaps are overcome. Solutions to two of these are described briefly: a dynamic keyboard, and software real time musical voice synthesis on an 8-bit microcomputer.

## TECHNIQUES OF COMPUTER MUSIC

Computer music began in earnest in the 60's when real time generation of sound became possible by hooking a D/A to a minicomputer. The input device was the ASCII terminal, and huge programs were developed to convert simple instructions (what note, what sound, how long, what envelope...) from the user into sound output. The process was indirect, and often not even on line. Although some composers found the potential fascinating [for example, John Cage. See also Ref 1], and significant work was done in discovering by simulation what made the sounds of familiar instruments what they were [Ref 2], computer music of this generation never attracted a large audience. Many listeners found it dry and inhuman at best, and there were no performers instruments where one could sit down to a keyboard and hear an immediate sound, and modify it in an interactive way.

The number of people able to do computer music has expanded greatly with the introduction of personal computers, freeing the field from its previous limitation to a few large expensive studios. Techniques and commercial products have developed for shaping the four basic parameters of a musical sound:

Pitch (what frequency is the note)

Timbre (harmonic content; like a flute or something harsher?)

Duration (percussive, like a drum, or continuous, like an organ?)

Envelope (initial attack, sustain while held, decay at end)

The techniques range between two limits, from making sounds in software only, to doing it all in hardware.

**Computer Music by Software** Simple software-only methods are the music boxes of computer music. The pitch is set by timing loops in a program, the timbre is a square wave set by the binary 10101010... string fed to the D/A, duration is set by another program loop, and the envelope is either on or off. Numerous articles describe this technique for various microprocessors [Ref. 3]. The technique introduces one to the ideas of real time sound generation, but rapidly loses its charm through limited capability. More satisfactory software solutions are described later.

**Computer Music by Hardware** This approach overcomes some limitations of software solutions by building special purpose hardware to take over most real time tasks. For example (Fig. 1.), an arbitrary waveform is stored in local memory, and strobed out by a local counter whose modulus (hence the pitch) is set by a number latched in from the microcomputer. An arbitrary envelope is also stored in local memory, clocked out at a speed set by another number, which sets the

duration. Commercial versions with S-100 bus compatibility have been available for several years (e.g. the SB-1 from SSM Corp.).

Either route is a useful demonstration of concept, but both have serious limitations. On the input side, there has to be a program which accepts musical information and instructs the computer how make the sound. This is not a real time keyboard, and 'compositions' are usually played back later. On the output side, these techniques tend to sound dull, being mostly monophonic, metronomic (the computer keeps time too well), and unmusical, with little richness of sound.

Computer music largely remains the domain of computer experts, and although opportunities for music education, composition, and analysis of the physics and psychology of sounds are being explored, the true performing instrument is rare.

## NEW DEVELOPMENTS IN ELECTRONIC MUSIC INSTRUMENTS

Classical musicians usually dislike what they've heard of electronic music (the grunt, squeak, and whistle school, or the imitative 'switched on Bach' school), and who can blame them. But they also ignore some genuinely creative sounds coming out of rock music (try the intro's to the Electric Light Orechestra's songs, for example). Much of this is electronic music, but it is performer's music, with the emotional content that comes from creating the sounds live. Rock musicians have discovered how to utilize the capability of commercial synthesizers fully, and have the dollars to spend to encourage the development of new ideas and new products. As a result, there is now a new generation of performing instruments, fully polyphonic, capable of synthesizing a wide variety of musically interesting sounds. Most of this new generation uses microcomputers extensively. Since most computer users are unaware of this creative ferment, some key developments will now be reviewed.

**Polyphony** Many instruments can now play several notes simultaneously, unlike the original Moog where creating an orchestra or even a piano meant multi-track dubbing (which limited 'live' performances). Polyphony is done by having a number of independent voices: the combination of voltage controlled oscillator (VCO), voltage controlled amplifier (VCA), and voltage controlled filter (VCF) which shape the sound in an analog instrument. The voices are dynamically assigned by a scanning keyboard, which uses digital hardware. Although the keyboard may have 60 keys, we only have ten fingers, and many current devices make do with less (4 or 5 is typical). A creative example of what can be done is the EMU keyboard, a microprocessor-controlled 16-channel keyboard which also can store in memory a sequence of note combinations input from the keyboard, for later playback as a repetitive 'rhythm line' underlying a live melody line. The keyboard is configured by a touch tone key pad, which can assign different sections to different sounds, transpose, define and play back sequences, etc. This is the Cadillac, costing $3000 for the keyboard alone, with no synthesizer! Many other less expensive but of course less flexible polyphonic keyboards are part of the newer generation of instruments [for a musician's review of this generation, see Ref. 4].

**Microcomputer Control** Patch cords are gone, and the configuration of voltages and resistances which define the parameters of a given voice can now be stored in memory. A given patch, when recalled, sets the parameters by D/A's or MOS switches. Many different patches can be recalled instantly, which greatly enhances live performances, since the sound qualtity can change to fit the music without a pause to adjust patch cords or slider pots. Most of these instruments still produce the sounds in an analog way, but the electronics for a basic voice (VCO, VCA, VCF) is now available in special IC's. A good example is the Sequential Circuits Prophet 5 [Ref 4]. It produces a musically rich sound by doubling its five voices, assigning two oscillators to any note. Slight variations between the two take away some of the 'electronic' sound of an unvarying oscillator, as with the multiple strings of a piano or the interplay of independent string

musicians.

**Dynamic Keyboards** A synthesizer keyboard is usually more like an organ than a piano. The key is either on or off, giving the performer no way to vary the loudness or the tone quality by how hard a key is struck. A few exceptions exist. The Yamaha CS-60 and CS-80 [Ref 4], for example, have extra travel at the bottom of the range, so that additional pressure on a key being played can alter loudness, tone quality (brilliance), or modulation (vibrato/tremulo). All three variables are adjustable in any combination, with front panel sliders. The dynamics mechanism is suprisingly simple but effective, consisting of a spring-loaded photoresister-light source combination to generate a control voltage. This synthesizer itself, however, is a strictly analog one, although polyphonic.

**Digital Sound Generation** This is just beginning, with affordable versions now limited to instruments based on the top octave synthesizer chip now the basis for most electronic organs. The voicing and envelope shaping of a typical syntheziser is added, usually by analog means. These instruments are inherently polyphonic. An example with only preset voices is the Krumar Orchestator. One of the synthesizers in the ARP Quadra [Ref 4] also works this way.

## NEW TECHNIQUES IN DIGITAL MUSIC DEVELOPMENT SYSTEMS

Truly digital sound generation offers in principle the same flexibility which programmable microcomputers brought to electronic instruments of all kinds. A number of new concepts are being applied.

**Additive Synthesis** Although analog synthesizers usually begin with the harmonic rich ramp waveform, any waveform can be stored when digital memory is used. Some instruments allow for additive synthesis, letting the user define the amplitude of each harmonic of the wave. An example is the New England Digital Synclavier, which gives complete control over the first 16 hamonic amplitudes. This instrument sets all parameters via a single control knob/numeric display panel, with simple indicator lights to tell which parameter is being adjusted. Although offering superior precision (3-digits) of control, the emphasis is on setting up patches at leisure, since the simultaneous but physically separate control of different functions is lost.

**FM or Phase Modulation Timbre** A slight frequency modulation produces the famliar tremolo. But if the modulation frequency is equal to or a harmonic of the frequency being modulated, a complex harmonic spectrum is heard [Ref. 5]. This is a very 'economical' technique, in the sense that a rich variety of sounds can be produced starting with only a sine wave. Since the frequency modulation amplitude can be adjusted continuously and in real time, timbre variation within the interval of a single note is possible, akin to 'natural' instruments. The tone quality of a trumpet, for example, varies geatly from the attack to the sustain, and a piano note begins harsh but ends as a sine wave, as the higher harmonics damp out more quickly. Phase modulation is done in a computer by varying the rate at which a stored wave is clocked out (Fig. 2). The Synclavier includes this feature, as does the about-to-be-introduced Crumar General Development System.

**Digital Filter Music** Here, the oscillators are digital versions of the 4 Op Amp analog simulation oscillator, which simulates the mathematics of a damped second order system. Done digitally, the corresponding difference equation is solved in real time. Oscillator parameters such as pitch and decay rate are set using real time multiplier IC's which set the coefficients in various feedback loops. With fast enough hardware, such an instrument is capable of rich polyphonic sounds. The oustanding example is the Bell Labs Digital Filter Synthesizer [Ref. 6], developed as an outgrowth of research in digital filters for real time signal processing in communications. The instrument has up to 32 oscillators, a keyboard which can be dynamically assigned for note inputs (not limited to the 12 tone scale) or the adjustment of parameters. For example, one demonstration uses the

keyboard to transpose the pitch of a computer-generated voice, generating 'barber shop' polyphony when several keys are pressed simultaneously.

All of the above techniques are so far available only in large expensive systems. Both the Synclavier and the Crumar Development System sell for about $25,000, five times the price of the most expensive electronic music synthesizer discussed earlier. These are, however, true development systems, being based around a very visible computer (not buried in the instrument as with performance-type synthesizers), usually with a terminal and floppy disk drive. Digital music is still very new, and the parameters of a less expensive instrument have yet to be worked out. The present breed of development systems, accessible to those unafraid of computers but formidable to musicians, could help to establish the parameters. However, should the non-musician computer expert be trusted to decide what is needed? How can interactive programs be developed to allow the non-programmer musician to experiment with a system's potential, closing the development feedback loop so that musically satisfactory solutions evolve?

## DIGITAL MUSIC EXPERIMENTS WITH PERSONAL COMPUTERS

We have begun experiments to look at some of these questions, set initially in the framework of a medium priced 'personal' computer system, a Cromemco System 3. The potential appeal differs from the digital music development systems above. We are aiming at the audience of personal computer owners which is much larger than the audience who can access the $25,000 development system. By widening the audience, some of the creative potential which has characterized recent developments in personal computers may apply itself to the exciting possibilities of digital performing instruments. A gap exists in the development of a instrument which first of all uses the resources of an existing personal computer to save costs and add flexibility, and secondly is a true performing instrument, defining the sound and accepting keyboard input and playing in real time. We assume from the outset that sound generation will be digital, and explore the limitations of present generation CPU's for this real time problem.

**The Dynamic Scanning Keyboard** It is relatively straightforward to make a polyphonic keyboard. Given that microprocessor (MPU) time will be largely devoted to generating sounds, the keyboard scanning has to be done locally, with the information 'which keys are now pressed' latched and strobed into the MPU at periodic intervals. Hardware solutions exist [Ref. 7], although a MPU dedicated to the keyboard is a prefereable solution, since it facilitates trying out new concepts with simple program changes.

A dynamic keyboard requires presenting not only which keys are pressed, but also how hard or how fast. Although force or velocity encoders are possible, the more general solution of presenting an array of instantaneous position information allows the dynamics algorithm to be defined in software. By varying the algorithm, one can allow the performer to exeriment with how it feels to go from a percussive instrument, where what matters is how hard you strike the keys, to a force-like alogorithm, where extra pressure on a key already pressed gives extra 'oomph' to the sound. We assume as a goal that each key has to generate this information separately, so that the full dynamics of the piano repertoire is accessible (for example, left hand notes slurred, while right hand notes are stacatto). While not limited to the characteristics of any existing instrument (if you want a piano, then buy a piano), we also recognize the importance of totally non-electronic aspects, and begin with a keyboard having the mass and some of the mechanical resistance of the piano action.

An example of an elegantly simple position sensing mechanism was developed by Alles [Ref. 8]. Each key carries a small plate, passing near a ribbon cable whose wires act as a second plate, forming a capacitor array. The wire closest to the plate has the strongest coupling and provides the basis for position

encoding. In the Alles scheme, pulse information is strobed down the ribbon cable with phase delay between the signals sent down each wire. The plate on the key is the receiver, and the phase of the signal is a measure of key position. We have determined [Ref. 9] that reversing the role of transmitter and receiver simplifies the detection electronics and enhances noise immunity (the coupling is only a few pF, so extraneous noise could be a problem).

Our circuit for a single encoder is shown schmatically in Fig. 3. The pulse fed to the key transmitter plate is generated by a high voltage transistor so that the voltage signal arriving at the weakly coupled second 'plate' approaches TTL levels. Capacitive coupling between wires on the ribbon cable is eliminated by grounding every other wire, resulting in strong localization of the received signal. Circuitry within the waveshaping block further shapes the received signal for easy adjustment of threshold levels. This is important in ensuring that one or two wires (not zero, and not more) generate an output pulse in spite of mechanical variation in spacing between the keys and the ribbon cable. No precision mechanical adjustment of individual keys is necessary.

The array of 16 outputs from the ribbon cable is then latched. If the transmitter is directly over one conductor, only that wire gives an output. The threshold of the waveshaping is adjusted so that when the transmitter is midway between wires, both give an output. This eliminates the 'no position' data condition. The resultant data ambiguity is resolved by a priority encoder, whose output is the binary number corresponding to the higher weighted conductor. The 'two pulses received' condition contains information which doubles the resolution, recovering what was lost by giving up every other conductor to ground. This condition is separately latched to form an additional bit.

**Software-only Digital Music Synthesis**

In addition to the idea of storing a waveform in a lookup table, digital techniques can readily generate the pitch, set the amplitude and timbre, amplitude and frequency modulate, and sum polyphonic voices before conversion to the final analog output. In addition to avoiding much expensive analog hardware, design complexity can be reduced and development flexibility enhanced if all of this is done in software. Such ideas have been summarized by Alles [Ref 10], and led to many of the features of the Crumar Digital Development System above. The Alles scheme uses dedicated special purpose hardware to overcome the speed and resolution limitations of the current generation of microcomputers. We have explored, instead, a software only implementation which demonstrates the techniques on a readily accessible personal computer system. With 8-bit resolution and limited speed of the Z80 used, the results are musically crude, but do indeed demonstrate the power of digital synthesis. A straightforward extension to the newer generation of 16 bit micro's will overcome much of the present limitations (for example, the pitch set to 8-bit accuracy is definitely out of tune) with a minimum of programming difficulty.

A series of six programs demonstrate on a personal computer the features of digital music synthesis [Ref. 11]. The programs, written in Z80 assembly language, were carefully minimized to meet the demands of real time sound generation. Even so, timing considerations require each of the features to be demonstrated in a separate program, and polyphony with even two voices is possible only by virtue of the Z80's double register set.

1. Digital Oscillator A sine wave or other waveform is stored in a 256 byte lookup table. The wave is scanned out at a rate which is varied by setting the size of the 'frequency control word' which generates an offset address. The even tempered scale is only approximately in tune because of this 8-bit limitation in setting the frequency, but double precision would slow the program so much that the full audio range could not be covered. The basic timing loop of the oscillator requires only 6 instructions.

2. Amplitude by Phase Cancellation Although output amplitude is usually set by a multiplying D/A or VCA, this too can

be done in software [Ref. 10]. Two table lookups are done, offset by an amplitude parameter PHI, and the results are subtracted. The amplitude is established by phase cancellation, since

$$\sin(A + PHI) - \sin(A) = 2 \sin(PHI) \sin(A)$$

This avoids a time consuming multiply operation.

3. Polyphony More than one voice can be synthesized 'simultaneously', and the voices added together while still represented in binary form, eliminating the need for an analog summation step. This is problem challenges the real time capability of a micro, but is facilitated by the double register set of the Z80. Each register set executes an independent oscillator subroutine as in (1), and the two are added in a third subroutine.

4. FM Timbre Generation As in (3) there are two independent oscillators, but now the second acts to modulate the phase of the first. The amplitude of the modulating waveform, which specifies the modulation index, is set by phase cancellation as in (2). If the index of modulation approaches 1, the output is rich in harmonic content, governed by the usual Bessel function expansion of FM.

5. Swept FM Similar to (5), this program demonstrates the potential of varying the timbre during the envelope of a single note. The modulation index is swept slowly , varying the sound from a pure sine wave to a harmonic rich overmodulated tone. Since both frequencies are still independent, all the possibilities of FM synthesis (bells, trumpets,...) can be demonstrated.

6. Amplitude Modulation To complete the bag of tricks in digital music synthesis, the two oscillators of (3) are now used for AM. The user specifies the modulation index (done by phase cancellation as in (2) and frequency of the modulating wave. Multiplication is again avoided by using this modulating signal to alter the amplitude of the tone by phase cancellation.

Although musically imperfect, these results demonstrate that even the present generation of 8-bit micros is capable of real time digital music synthesis. The outlook for completely software real time music synthesizers with the next generation of 16 bit micros is therefore promising.

REFERENCES


1. M. V. Matthews, The Technology of Computer Music, Boston, MA, MIT Press, 1969.

2. For example (violins), M. V. Matthews and J. Kohut, J. Acoust. Soc. Amer., **53**, 1620 (1973). For a review, see J. A. Moorer, Proc. IEEE **65**, 1108 (1977).

3. For a survey, see the Byte Book of Computer Music, Byte Publicatons, 1978.

4. N. Milano, Contemporary Keyboard, several issues in 1979.

5. J. M. Chowning, J. Audio Eng. Soc. **21**, 526, 1977.

6. For a review, see H. G. Alles, Proc. IEEE (in press).

7. Steven K. Roberts, Byte, Jan. 1979 p. 104.

8. H. G. Alles, J. Comp. Mus. 3, No. 3, p.28 (1978).

9. R. K. Goodall and R. J. Higgins, J. Comp. Mus. (to be published).

10. H. G. Alles, J. Comp. Mus. 1, No. 4, p. 14 (1977).

11. R. J. Higgins and R. Vedanayagam, J. Comp. Mus. (to be published).

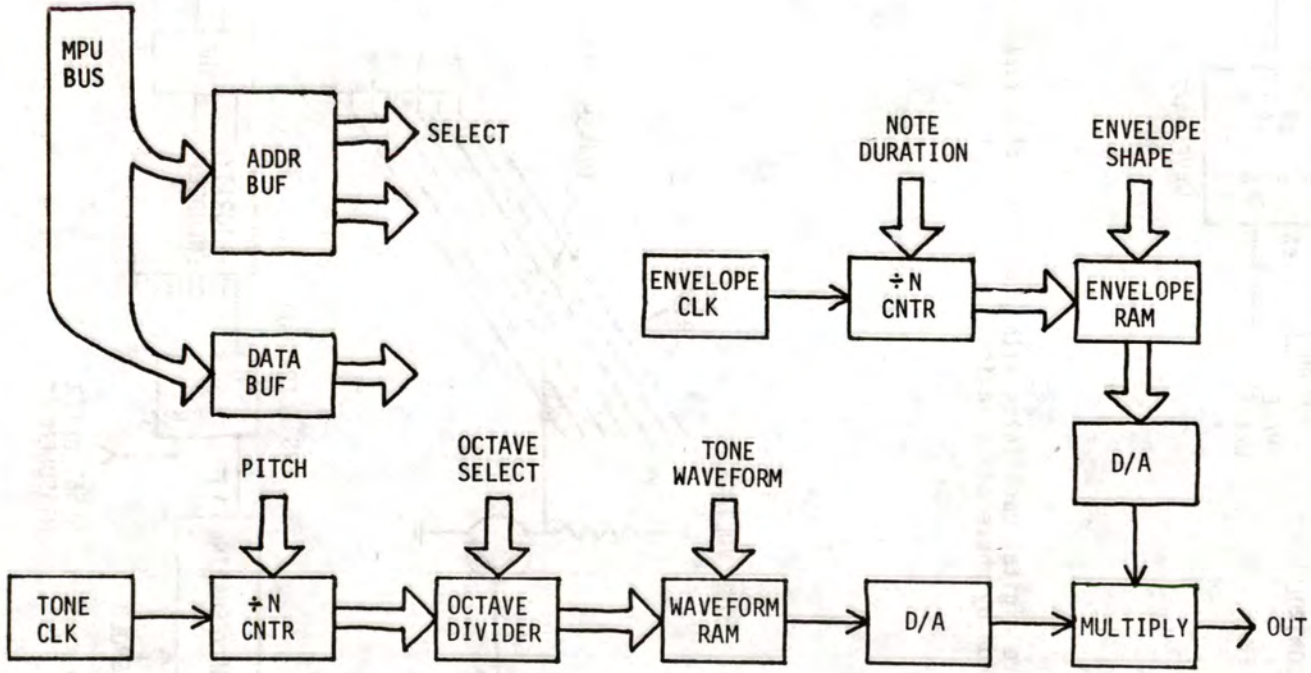



Fig. 1-- A hardware-intensive solution to real time digital music synthesis.

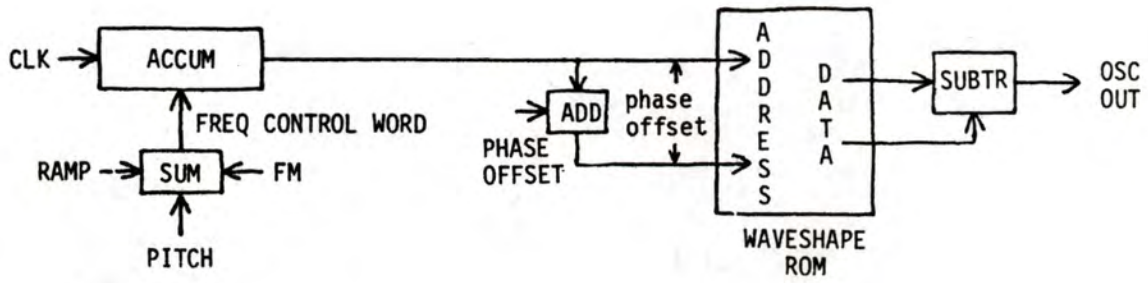



Fig. 2-- Software digital oscillator with phase modulated timbre and amplitude by phase cancellation.

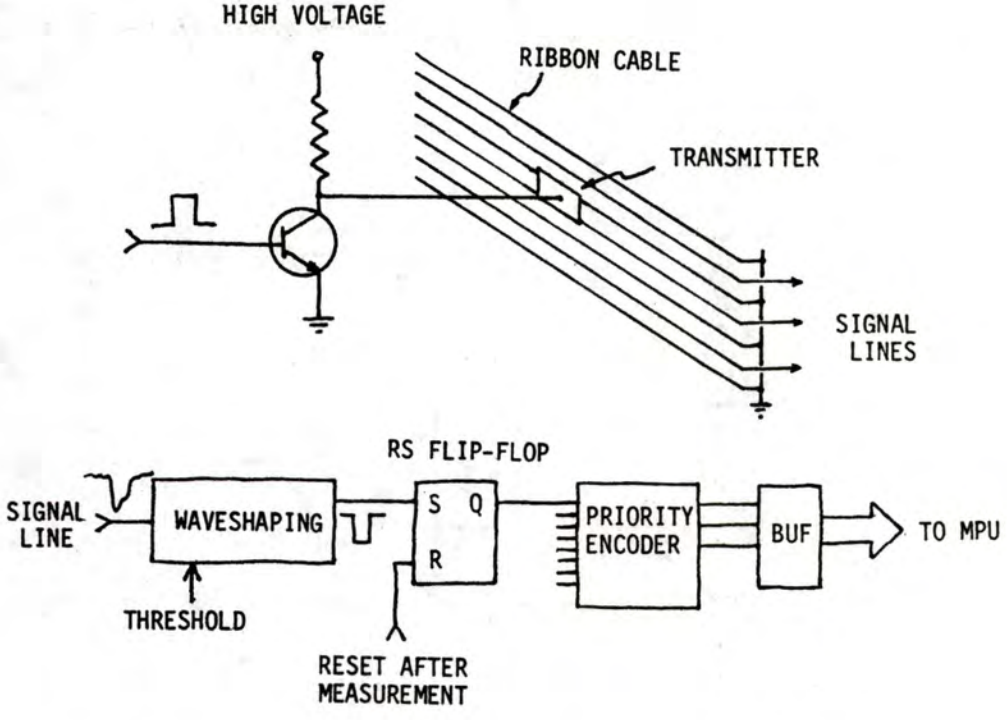



Fig. 3-- Position encoder circuit for a single key.

# TELECOMMUTING VIA THE PERSONAL COMPUTER

by
Jack M. Nilles, Director
Interdisciplinary Programs
University of Southern California
Los Angeles, California 90007

## Introduction

In 1973, an interdisciplinary research team at the University of Southern California began a study of the technological feasibility and societal impacts of a concept called "telecommuting." This coined word refers to the use of telecommunications and computer technologies to serve as a partial or total substitute for the daily trip to work. At the time the study was performed, personal computers did not exist. The technology that we were concerned with at the time was generally that of mini computers and larger systems. The subsequent introduction of the concept of distributed processing and of personal computer technology, as well as the appearance of several other trends, such as continued threats of major reductions in the availability of petroleum in the U.S., make it appropriate at this time to reexamine the issue of telecommuting in the light of contemporary conditions.

Several major factors and trends in contemporary society are acting to continually increase the desireability of telecommuting for several types of workers. In this paper I can only briefly summarize them, but the list of references at the end of the paper will give the interested reader more of the details. The fundamental issue concerns the relative advantages to the employer and the employee of telecommuting instead of the traditional way of getting to work. Unless there are clear advantages to both employer and employee, the general concept will not be successful. Secondly, if telecommuting becomes more wide spread than it is today, there are some broad-scale societal impacts which must also be considered.

In our original study, we took the position that telecommuting would not be a viable concept unless there were specific, measurable incentives to the employers of telecommuters to make or allow it to happen. Then, and only then, the concept would also have to be attractive to employees if it were to become wide spread. Given these basic "What does it take to convince first an employer and then an employee that telecommuting is a viable concept?"

## Background Trends

As a first step, we can rule out about half the contemporary U.S. labor force from participation in telecommuting. That is, our area of emphasis is on the other half of the labor force which constitutes what we call the information industry: those people whose primary job is to move, manipulate, and/or transform information in some way or other. Since half of the U.S. labor force is about 50 million people, in 1980, we are still talking about a fairly substantial number. Most members of information industries live and work in or near urban areas. The typical information worker commutes to work, as the sole passenger in a private automobile, a little more than nine miles each way (in Los Angeles this number is just over ten miles each way). The level of complexity of information work ranges from the humdrum routine to the most complicated imaginable mental effort. The price paid for the fruits of information work is generally proportional to the creativity required to perform the job. The capital investment in the average information worker is quite low compared with the capital investment in a manufacturing worker. The productivity of information jobs, as measured in manufacturing terms, has remained essentially constant over the last decade, while manufacturing activity has steadily increased at about a three percent annual rate. Salaries of information workers tend to increase roughly in step with inflation. The costs of transporation by private automobile are rising at a rate slightly faster than inflation. Housing prices in metropolitan areas have risen past the point where a private home is affordable by many young couples. The costs of computer technology have been decreasing at about a twenty-five percent annual rate. Individuals seeking jobs are more concerned with the "meaningfulness" of the prospective job than with any other factor. What does

all this mean?

Why Telecommute?

From the employer's point of view, it means the following things. It is increasingly difficult to find qualified and willing employees for entry-level information jobs, i.e., clerical and secretarial workers. At these levels and at higher, middle management levels of the organization, employees are looking for more diversity than routine in their jobs. Costs of information work continue to increase, while productivity does not. On the other hand, employees can be induced to accept lower rates of increase and fringe benefits in exchange for better working conditions. One means of providing "better" working conditions is to decrease the amount of time an individual employee spends in commuting to work. One way to reduce the commute to work is to decentralize the organization to some extent, setting up regional work centers located near the prime residential areas of the organization's employees (present or prospective). Decentralization is known to cause a variety of managerial headaches. Computer technologies are becoming much more attractive, particularly since, with the advent of effective computer-to-computer networking links, the effectiveness of operation of a computer becomes relatively insensitive to its physical location. Enter the low-cost microcomputer; a personal computer. The personal computer has the potential of being an extremely rich information tool for the information worker. With a relatively modest capital investment the productivity of an individual worker can be significantly increased, particularly for relatively routine information processing tasks. In 1974 we made fairly convincing cost-benefit analyses of the possiblities without considering the use of personal computers. With the advent of the modem-equipped personal computer, the basic economic factors became even more attractive to the employer.

The advantages to the employee are less easily defined in strictly economic terms. The primary economic advantage to the individual employee may simply be the reduction in out-of-pocket costs for commuting to work; if he/she only has to travel a distance of a mile or two to work instead of more than nine miles, on the average, the annual savings of transportation costs can be several hundred dollars. If, on the other hand, the employer compensates for this saving in the form of reduced salary and/or benefits (or their equivalent in reduced rate of salary or benefit increases) that saving can be a wash (even though the employee may not notice that it's happening). The more important advantage expressed by many prospective telecommuting employees is in the increased time it gives them for their non-work lives. The sudden decrease in, or absence of, the commute to work not only frees up an hour or so per day for the typical employee, but also reduces the tension associated with commuting, the inhalation of carbon monoxide and other noxious vapors and, if the option is available of walking or cycling to work, generally positively affects the employee's cardiovascular health.

Why Not Telecommute?

What are the disadvantages then? For the employer there is a major impediment to telecommuting: the requirement that a new organizational style must be embraced if telecommuting is to be effective. For example, let us suppose that a company decides to decentralize into four regional work centers in a metropolitan area. In order for the concept of telecommuting to act so as to realize the greatest benefits, an employee reports to work at the office of the company which is nearest to where he/she lives, regardless of the job performed by the employee.

For example, the accounting department of Company A, which formerly was located in a single, coherent group of offices in the company headquarters building, is now scattered among four locations, with some employees at each. The manager of the accounting department immediately wants to know how it is possible to check on the workers without some form of personal, face-to-face, supervision. The flip answer to that worry is simply that all it takes is the development of appropriate software for the computer communications network and the manager can instantly check on the progress of his employees' work, regardless of where they are located. In practice, unfortunately, such software does not generally exist. Thus, the company contemplating telecommuting also must contemplate a nontrivial software development project in order to make

the system work effectively.

Other worries also plague the employer. One is the increasing threat of unionization of his information workers. Current Federal regulations allow single sites of a corporation to be unionized even if the other sites are not. Thus, splitting the company up into several locations instead of a single one increases the probability of unionization. The employer also worries about employee productivity in the new situation. Generally this fear can be set to rest. Tests made during our original study generally demonstrated an increase in productivity on the part of employees working in a decentralized operation. Similar tests conducted elsewhere around the country have shown similar results.

The primary concern on the part of the employee has to do with the social aspects of work. There are two facets of this problem. First, for the aspiring manager, the worry is that location in a remote installation will decrease his/her opportunities for advancement, which are felt to be intimately related to the number of face-to-face contacts the aspirant has with upper levels of management. The impersonal personal computer is not felt to be a suitable surrogate for the "pressing of the flesh" involved in high level management conferences and casual meetings in the corridor. For this reason, telecommuting is felt to be only a partial solution for managment level personnel. In the typical management telecommuting scenario, the manager still frequently appears at corporate headquarters even though he/she may also spend at least an equal amount of time at one of the decentralized sites.

The second social disadvantage to the ultimate in telecommuting, working at home, is simply that the typical worker enjoys the social contacts of the office environment. Making the entire transition from working at a metropolitan center to working at home is generally felt to be too drastic a step because it severely restricts these sorts of social interchanges.

## Energy and Environmental Considerations

Telecommuting has direct implications on our uses of energy, particularly irreplaceable fossil fuels. At present levels of transportation usage for commuting, with the heavy dependence on the private automobile, the ratio between energy consumption for commuting using an automobile and telecommuting in which the commuter walks or bicyles to work is about 100 to 1. The annual energy saved by seven telecommuting employees would be the equivalent of the electrical energy required to run an average American home for a year. Replacement of about 12 percent of urban commuting by telecommuting in 1980 would eliminate the U.S. requirement for imported gasoline. This latter factor is probably the one most likely to sway some attitudes toward telecommuting in the future. Whether or not telecommuting can be viewed as completely desireable in all types of work situations, it may become a pragmatic necessity if U.S. economic activity is to continue at its present level under conditions where oil imports are severely restricted. Until we develop a far higher degree of energy self-sufficiency than we have at the moment, the telecommuting option should be considered if only for this reason alone.

The environment benefits as well. Since commuting and the rush hour ar synonymous, commuting automobiles tend to operate inefficiently. Start and stop driving

not only wastes gasoline but also increases the production of air pollutants, particularly carbon monoxide and unburned hydrocarbons. Threrfore, any reduction in commuting results in a more than proportionate reduction in air pollution.

## New Forms of Work

The advent of low-cost computer technology and particularly that of the personal computer allows us to think of many more types of work opportunities, via telecommuting, than were available in the past. Basically, our continued growth toward becoming an information-dominant society already has strong pressures toward the development of new kinds of information work. Furthermore, existing forms of work, existing types of jobs can be accomplished in new ways thorugh the use of telecommuting.

For example, telecommuting offers a new set of options for part-time workers. Many information jobs can be defined on a piece-work basis. Typica

of these are text editing tasks, consulting services, various forms of transactions services, and the like. With sophisticated telecommunications technologies there is no basic reason why the performer of these services has to be located at a particular place. The concept of the independent information worker becomes considerably more viable than has been the case in the past where it was required that the worker be in close physical proximity to the massive amounts of data, contained in file drawers, required for the work. The data are no longer in file drawers, but in electronic or optical form. The worker need not necessarily be physically colocated with the data base or, on the other hand, the data base need not be physically massive. Hence, the individual worker or group of workers can develop a specific information service which can be sold to a variety of buyers in the open market.

This opens job opportunities not only for the physically handicapped but also the worker who may only have a few hours per day to devote to work. For these situations, we have the more advanced form of telecommuting in which the worker is likely to be working from the home rather than from a regional work center. In practice, as the concepts of telecommuting develop, some combination of all these options will occur. Some companies will find it impossible to decentralize and will continue their operations in much the same way as they have in the past. Others, whose organizational structures are more flexible and more suited for this new way of life, will decentralize via telecommuting, particularly as personal computers become ubiquitous. The more entrepreneurial spirits in the country will use personal computers and telecommunications networks as major tools in developing their own, more independent ways of life. Given a personal computer, a modem, a telephone line and one's own individual information expertise, the future information worker may be freed from many of the physical restrictions that may have been a hindrance in the past.

Work hours, in a piece-work information economy, will be equally flexible because emphasis will be placed on the work produced, not the time spent in producing it. The skillful worker, aided by computer-based information tools, may use this advantage over less skilled workers to produce either more income or more leisure time. Of course, this aspect of the work situation is more likely to occur first at clerical and secretarial (text processing) levels than in management situations. This is because these more routine tasks are more easily quantified.

## Broad Scale Impacts

In contrast to the work at home situation, broad-scale adoption of regional-center-oriented telecommuting could lead to a much more rational form of planned urban/rural development. At the first level of effect, telecommuting would increase the importance of regional work/shopping/recreation centers in major metropolitan areas as the commuter load shifts to the regional centers from the monolithic central business districts of contemporary cities. Furthermore, the concept of telecommuting is worth investigating as a means of both redeveloping small towns in the United States and of providing an alternative to high urban housing costs for younger members of the workforce. Suburban and rural areas can become active total communities rather than the bedroom communities or ghost towns that many are now.

Thus we can visualize an image of the future in which the major metropolitan areas are transformed into clusters of regional business centers surrounded by residential areas and open space and interlinked by line-haul transit systems. In this image the automobile will no longer require one-third of urban land area to maintain its existence. The average information worker's use of the smaller, more efficient automobile will be largely for shopping and recreational trips rather than the commute to work. Yet, because of the power of personal computers, the work diversity and job mobility of the information worker will be at least as high as, and probably higher than, at present.

## Prognosis

This has been only a brief sketch of a possible future. Whether it will be a real future, and for how many people, rests on a number of entirely nontechnological factors. For a change, many of these factors can be under the control of the individual or, to emphasize the key point, articulate and forceful groups of individuals. The problems of management of

information work in a decentralized society have to be overcome. Sensible urban planning which considers all the technological and social options has to occur and be enforced. The technologies have to be adapted to human uses, not the other way around. Individuals have to decide on their own priorities; if the trend continues toward meaningfulness instead of gross consumption the chances of this occurring improve. Will telecommuting in some form happen ever? It's going on now for some people and the number of telecommuters is increasing. Will it be widespread? Not for a while, if ever; that depends on the peopleware just mentioned.

References


Gray, Paul; Nilles, Jack M.; and Lopez, David A. "Communications for Transportation: Implications for Traffic Engineering." Transportation Engineering 11 (1977): 19-24.

Hiltz, S. Roxanne and Turoff, Murray. The Network Nation: Human Communication Via Computer. New York: Addison-Wesley Publishing Company, Inc., 1978.

Martin, James. The Wired Society. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978.

Nilles, Jack M. "The Automated Office: Some Preliminary Observations on Changes in Work Style." In NTC'77 Conference Record, vol. 3, pp.45:4:1-4. New York: Institute of Electrical and Electronics Engineers, 1977.

_____. "Telecommunications and Urban Structure." In Proceedings of the IEEE 1977 International Communications Conference, pp. II:257-60. New York: Institute of Electrical and Electronic Engineers, 1977.

_____. "Telecommunications and Organizational Decentralization." 3rd rev. ed. In Mexicon '76, IV: 3-9. Mexico City: Institute of Electrical and Electronics Engineers, Mexico Section, 1976.

_____. "Talk is Cheaper." IEEE Spectrum 7 (1976): 90-3. This paper was adapted from a presentation to the Annual Meeting of the American Association for the Advancement of Science held in Boston, February 1976.

"INFORMATION w/cheese PLEASE"
THE EMERGING PERSONAL COMPUTER NATIONAL INFORMATION UTILITY NETWORK

by Ron Jacobson
Broadcast Communication Arts Department, San Francisco State University
Mailing address: 45 Westwood Drive, San Francisco, California 94112

## Introduction

As we are all aware, computers have entered the consumer market in a big way. Interface Age reports that home computer equipment sales are projected to be $1 billion in 1980 from a standing start in 1977. Our society is in transition, moving from a service to an information-oriented economy. Within sight is mass usage of a public information utility network, where data flows from host computers into the home computer terminal as freely and easily as water flows from the kitchen faucet. The possibilities of such a communications system are tremendous, the consequences staggering. What is at stake is a revolution of our sensibilities and the way in which we live our lives. This subject cannot begin to be thoroughly explored in the limited amount of space available here. Instead, I plan to pursue an historical and critical approach to the subject by reviewing the characteristics of some present systems operating in the world today and by trying to establish criteria for the systems of tomorrow. First, let's take a peek at what's happening today:

## Prestel

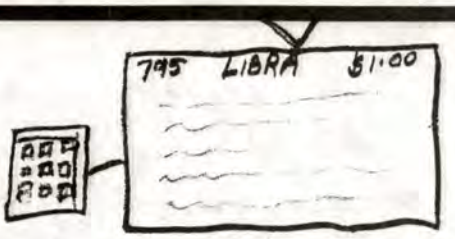
Prestel/Viewdata is the system designed and operated by the British Post Office. It is second only to the public library in the United Kingdom as a public service of facts, information, and knowledge. More than 150 governmental, non-profit, and commercial organizations provide information for the 180,000 pages of content currently stored. Pages of information may be requested from the home through a small calculator-like terminal connected to the telephone line. Data requesters are charged at the normal telephone rate in addition to a basic charge for the service itself. A commercial information provider may also charge a fee for pages of information.

The system uses ordinary telephone lines to connect the Prestel television set with a GEC 4082 processor. The basic Viewdata receiver is a television set with 1) an auto-dialler, which remembers numbers of computers it is likely to access and will dial them up at the press of a keypad button; 2) a modem, which converts audible tones transmitted over the telephone network back into a code recognized by the computer and terminal; 3) an identification number hardwired into the set so that the computer knows which terminal to bill for page access; 4) a character generator for construction of shape of each character that corresponds to codes transmitted to the terminal; and 5) a page store, which retains all the characters as they appear on store. There are a maximum 960 characters per frame (24 lines x 40 characters). Line 1 is reserved for system use, such as frame number and pricing information. Line 24 is reserved for other system messages. Graphics are built up as a mosaic of rectangles. The connection of the terminal to the telephone network is by jack plug and socket.

In the Prestel computer network, one GEC 4082 serves as the update machine, and all the others are dedicated to information retrieval. The GEC 4082 has 128Kb - 1Mb immediate storage capacity. The Prestel communication software is based substantially on CCITT standard X25.

There are shortcomings to the Prestel system which shall be listed later. The most popular and blatantly correct criticism is the

tremendous lack of intelligence in the home terminal. Less diplomatically, it's pretty stupid, in a relative technological sense.




Around the World

The British Post Office rushed to get Prestel into operation with the hope of setting a global standardization and benefitting monetarily from being "first." The German viewdata system, Bildschirmtext, is designed after Prestel and even uses British-purchased software. Holland and Hong Kong have also adapted the British system.

The French are developing a competitive system. Scheduled to be trial-tested outside Paris this year, Teletel is a viewdata system compatible with the French broadcast teletext system called Antiope. Teletel uses an electronic code to signify the beginning or end of a line of data. This is quite different from Prestel, which uses the time equivalent to one line sweep of the television screen to encode one line of data. Teletel also promises to be more user interactive than Prestel.

Bell Canada's viewdata system, Vista, is scheduled for commercial marketing in 1981. Meanwhile, there's Telidon, developed by the Canadian Department of Communications, and it bears a feature worth noting. Telidon can display curved lines rather than the more jagged graphics of Prestel. The whole viewer screen is like a dot matrix. Instead of a sequence of character codes constructing a mosaic, the Telidon computer transmits a set of geometric commands (line, arc, polygon, etc.) to describe the graphic representation. The result is a more realistic picture.

Japan had to develop its own system because of the complication of its written characters. Last year, the Captain system provided over 100,000 pages of information from 100 companies to over 1,000 subscribers in a Tokyo population sample. The five areas of service content on the system are news, public affairs, education, amusement, and "human living in general." It is planned for the Captain and broadcast television to be used compatibly in the future. For example, broadcast television would report a brief news story suitable for quick announcement, and the Captain would provide the detailed substance.

The Captain picture frame can contain a maximum 120 Japanese characters (lateral 15 characters x vertical 8 lines). Under investigation is the possibility of 480 small characters per frame (lateral 30 characters x vertical 16 lines). At the moment, large and small characters cannot coexist on one line. An extra feature of the Captain system is its ability to display color photographs by using a wide-band system instead of the ordinary telephone lines, which do not have enough capacity to transmit color pictures at acceptable speed.

The United States

According to the 1979 Nielson Report on Television:

98% of U.S. households own television sets.

Households, on the average, viewed 6 hours and 13 minutes of television a day during the 1977-78 season.

Television viewing hits it peak level at 8-10 p.m.

96% of the nation's television households have access to four or more stations. Only 38% can choose among 10 or more stations.

Cable television is present in 18% of U.S. television homes.

I threw these statistics in to show where we are today. Things are changing and some of these numbers are quite vulnerable to radical alteration in the near future. For instance, a recent Nielson study revealed how pay cable is threatening CBS, NBC, and ABC network programs by capturing large audiences. It is estimated that one out of every 15 homes now subscribes to pay cable service, and the predictions

are for the number to triple to 15 million pay cable homes within the next five years. ABC's own Video Enterprises Division will soon start developing properties for the pay cable and videocassette market. Cable networks are springing up left and right; many are offering specialized programming. Started last September, the Entertainment and Sports Programming Network transmits 24 hours a day via satellite to 625 cable systems. And, starting this June, Ted Turner initiates his 24 hour a day Cable News Network. I find a lot of these developments pretty exciting, but there's one major problem. All this stuff is still in a world of one-way communication.

Two-way communication is the top criterion for a public information utility network. The Qube system, a $10 million pay cable subsidiary of Warner Communications, Inc., began operation in Columbus, Ohio in 1977. It is an interactive system with 30 channels of programs, including one soft pornography channel (accessed only with a special key, which is presumably in the possession of the head of the household). Like many systems, the Qube viewer is charged for programs selected and billed monthly. "The Barber of Seville," for instance, shows for $2.50. The important difference with Qube is that the viewer can interact with a broadcaster via a computer interface. An example is a live program somewhat resembling "The Gong Show," where viewers score and gong the participants. On another program, a talk show host will interrupt an interview with a guest in order to conduct a live instant survey with the viewing audience to get their feelings on an issue. On the practical side, personal accounting and shopping can be done with the Qube system.

Two-way communication finds a powerful ally in the personal computer. You go out and buy a Radio Shack TRS 80, or an Apple, or a Commodore Pet, or an Atari, etc. You buy some software (Portia Isaacson predicts that next year the first software store will open - selling no hardware, just programs - like a record store. See reference #13). But there is more you can do. You can become part of an information utility network. For dirt cheap!

The Source

The Source is a low-cost computer timesharing information utility developed by Telecomputing Corporation of America, a subsidiary of Digital Broadcasting Corporation. The telcomputing network is available from 6:00 p.m. to 7:00 a.m. and all day on weekends and holidays for $2.75 per hour. The cost of usage at other times jumps dramatically to $15.00 per hour. There are no central processing unit charges. Disc storage is available at a low rate: $.033/block of 2,048 characters/day. There is a $100.00 network connection and registration charge. A 30 CPS Acoustic Coupler with Serial (RS-232C) Interface cost $245. All customer charges are transferred directly to a major credit card for monthly billing and payment.

I placed a telephone call to Jim Clark at Telecomputing Corporation of America's headquarters in McLean, Virginia to find out about the network configuration. Jim told me that the host computers are Prime 750s. "The Prime 750 system - priced between $180,000 and $300,000 - will support up to 63 simultaneous users, with a 32-million byte virtual address space per user, 8 million bytes of main memory, and maximum disk capacity of 2.4 billion bytes.(16)" When I spoke to Jim last year, he told me the company was using 10 Primes on the network and was adding one a month.

With The Source, users can write their own programs or choose from about 2,000 existing data bases and application packages. Programming capability is present in such languages as extended BASIC, FORTRAN IV, PASCAL, and Assembly. Data bases include United Press International"s worldwide news service, Dow Jones stock exchange information, Dartmouth College educational courses, the New York Times Consumer Data Bank, airline, hotel, and car reservations; games, and entertainment and hobby information. Users can develop new programs, offer them for inclusion in The Source library, and get paid royalties for every minute they are used by anyone else.

Electronic Mail & Message Systems, in its 2 July 1979 issue,

reports that "one of the most intriguing capabilities of The Surce is the ability to send electronic mail... The Mail Call utility allows a user to check his or her mailbox to see if a message is waiting, immediately after signing on. Messages are addressed and relayed via account numbers. The receiver of the message has the option of deleting it from their files, replying to the sender, filing it in storage, or forwarding the message plus a reply back to the original sender. Sending mail to another user is just as simple."

One more service on the system is worth noting. A Source subscriber can "chat" with another subscriber. People can set appointments for chatting using the electronic mail service. A subscriber types the word "chat" and his or her correspondent's account number on the screen. If the correspondent is not on the system, The Source will so indicate. If the correspondent is on, any typed message will appear on the screen. If this person would like a two-way conversation, he or she simply types "chat" on his or her terminal. If one does not want to be interrupted by "chatter" or is not in the mood for a crank call, one simply types the command "refuse chat" when signing on. Unlike Ma Bell's phone service, The Source is not capable of handling simultaneous messages from both parties. So, with The Source, one has to punctuate sentences and type an extra return when one party wants the other to respond. I've heard people call Ma Bell a cheap mother -----, but a California/New York Source chatting session of one hour length would cost the basic $2.75. Ma Bell would charge a hefty $16.50 for the same 8:00 p.m. weeknight call.

I want to take a moment here to stress the fact that I'm not advocating you go out and join The Source. I have been told there is a competitive system called Micronet which offers similar services. Of course, there are many time-sharing systems, like ARPANET and EDUNET, but they are not available as resources of information for your personal computer. I have no information as to the quality of performance of The Source, but that is not at question here. Again, I am using the system to show where we are today: at the ground level or in the embryonic stages, if you prefer, of thinking about, constructing, and implementing information utility networks for the general public and their personal computers.

Subscribers dial a local number to access The Source. Telecomputing Corporation of America uses two switching networks, Tymnet and Telenet, for packet switching services to help connect the some 250 cities The Source serves. For those unfamiliar with the term, a brief explanation of "packet switching" may be in order. Let's say you send in your request for the astrology program on The Source. After processing at the central processing unit, the program and your address will be sent out in a packet. Packets can contain up to 1024 bits or 128 characters each. Your packet is given an identification and a sequence number. It then passes through the network, with routing being selected by computers in the switching center. As the packet enters your loop, other computers will not pick it up because it's not addressed to them. It is addressed to you, and your computer will grab it for you. Your horoscope tells you that your personal finances aren't looking too healthy, but you already knew that and feel ill at being reminded. Oh, well.

Telenet and Tymnet present these features:

Store -and-forward service

Speed and code conversion

Transmission error detection

Virtual routing

Enabling customer to access common data files from remote terminals.

Accessibility to the computer network by dialing a local number from a customer remote location. (6)

A system like Telenet represents single-source network management and, by so doing, alleviates a major problem of network design. There are other advantages for using a packet switched network. There is an overall lower error rate because of internal de-

tection and correction. There is greater flexibility and adaptive routing to meet the traffic conditions of the system. Finally, packet switching networks are economical for other timesharing systems that operate on a national scale to use because transmission distance is not a factor in cost charges. Besides host to network interface cost, equipment, local telephone, and network port charges, there is a traffic charge based on the number of characters transmitted per hour or the number of packets.

Network criteria

Now that we have shallowly explored some systems, we can begin to ask ourselves what we like and dislike in what is present today and what features we would like to see in the emerging information utility network(s). Are we looking for one national network or a number of regional and local networks? I think there will be a call for intersecting networks ( like the airlines that have intersecting reservation networks). Should there be specialized personal computer networks? A medical information network is a definite possibility.

An important criterion in designing any network is user population. A potential problem for The Source is not being able to keep up with its growing number of subscribers. There must be capacity reserved for increases in traffic. To meet this requirement, there must be ample financial resources available. The capital of Telecomputing Corporation of America provides the economic base for The Source network. But what about a public information utility network?

I like the idea of Gene Youngblood's National Information Utility:

The National Information Utility "could integrate, synthesize, and transcend all the characteristics of both the switched telephone network and the mass distribution media. It could incorporate the program distribution, news publishing, library, telephone and postal services of the nation together with teaching, automatic process control operations, and professional and social services such as medical and legal aid, all in a single decentralized, user-controlled, special audience, perceptually adaptive, feedback communication system." (24)

We could pay for information just like we pay for heating our homes. (Last November, President Carter sent out special checks to people on Social Security and Welfare to help cover heating expenses for the winter. Someday, the Prez may be sending out checks to cover information costs).

Then we enter the important area of privacy. With Prestel and The Source, there is the option of password protection at the home terminal to guard against unauthorized access to the resources. Telenet provides an end-to-end encryption service to guard against unauthorized data in transit. But the major problem with packet switching and the X-25 protocol is that the address and the message of both the sender and the terminator are known. Privacy is a large concern of ours today, and many people fear we could lose much of what we have left with a personal computer network monitored by the state. Shades of 1984. The Privacy Protection Study Commission recently suggested that Congress create an independent Federal Privacy Board within our government to protect personal privacy and promote "fair informational practices."

How are we going to set up a National Information Utility? For network technology we look to the educational, military and commercial systems operating today. We also look to the rapid evolution of the general purpose digital computer and hopefully don't wind up with a Prestel-like system. For communication policy-making and setting up the Utility, we must turn to power-based existing institutions such as these:

I. Government
   A. Federal Communications Comm.
   B. Congressional committees
   C. Office of Telecommunications Policy in the Executive Office of the President.
   D. Office of Telecommunications in the Department of Commerce
   E. Office of Technology Assessment.
   F. Related agencies (FTC, Dept. of Justice, etc.)

II. Industrial
A. The National Association of Regulatory Utility Commissions.
B. Telecommunications Association of America.
C. Cable Television Information Center.
D. National Association of Broadcasters.
E. American Society for Information Science.
F. American Federation of Infomation Processing Societies.

III. Academic

IV. Citizen Organizations

One thing we want to have our say about is freedom of information. We must not have an information elite that decides what the general public will or will not have access to. Sure, there is a need for national security, but I'm not specifically referring to that. Information means power, and we don't want that just in the hands of a few.

You might be wondering why I titled this article "Information w/Cheese." Well, I was thinking about McDonalds ( "We do it all for you"), and Burger King ("Have it your way"). I guess I saw a correlation between fast food for the masses and fast information for the masses. One thing about McDonalds is that the food isn't very expensive. If you can't afford a Quarter Pounder with cheese, you might just order a regular Quarter Pounder. Those who can afford it might order their information with cheese - the whole works. You can specify the type of information you want. Like at Burger King, "special orders don't upset us." What we are talking about here is fair access - a democracy of information.

Other criteria in designing a personal computer network is reliability and availability. Reliability, of course, refers to performance without failure for a definite period of time or amount of usage. The availability of 24 hour emergency medical information on the system would seem mandatory.

One of the real possibilities that we haven't mentioned here is the development of a hybrid network of satellites, digital broadcasting , and coaxial cable or fiber optics. The personal computer can become a universal home terminal. (7)

I'd like to conclude with a little communication theory and philosophy. It has been proposed that knowledge and love are the two reasons for human communication. Knowledge is responsible for the content of communication while love determines the particular situation for the transfer of meaning. The bottom line is that the human being is the ultimate term of communication. Men and women, because of their unique physical existences and spiritual beings, use communication as a relationship. I believe the emerging information utility network must recognize us as people, or communication breakdowns will occur. I am saying that, in this Age of Information, the human being must be viewed as more than just a brain, a machine, a function.

We, too, share the responsibility of not letting our personal computers alienate us from one another. We are in a revolution. Power to the PEOPLE!

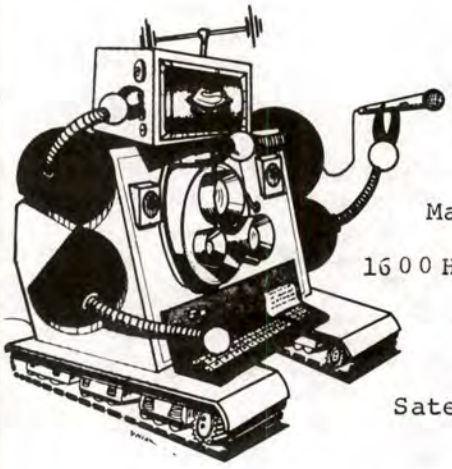
I jumped into the deep rivers
of chaos and confusion,
searching, turning over all
stones. I knew the truth
was swimming somewhere between
my fingers and I was changing.

Acknowledgement

I'd like to thank Professor Mark Cummings at San Francisco State University not only for the valuable material he lent to me to prepare this report but also for his unselfish encouragement and guidance. With his help, a Campus Community Computer Club has been formed with the purpose of installing personal computers in the San Francisco State Student Union. Anyone wishing to contribute to this exciting project aimed at increasing public awareness and computer appreciation should write to me at 45 Westwood Drive, San Francisco, 94112. Finally, I thank you for reading the article and welcome any thoughts, opinions, or additional information you may have. I think two-way communication is healthy, so get well soon.

Selected Bibliography

1. A.C. Nielson Company. 1979 Nielson Report on Television. Northbrook, Illinois: The Company, 1979.
2. Amara, R. "Toward Understanding the Social Impact of Computers." Report R-29. Menlo Park: Institute for the Future, 1974.
3. Becker, H. "Let's Put Information Networks into Perspective." Datamation, March 1978, pp. 80-86.
4. Bergland, R. "Comparing Network Architectures." Datamation, February 1978, pp. 79-85.
5. Collier, M. "Speeding Communications." Office Products News, August 1979, pp. 15-21.
6. Cummings, M. Competitive Analysis: Communication Carriers. Western Union, 1976 (in-house publication).
7. Cummings, M. "The Emerging Computer Revolution and Consequences for International Development." Third Asian-Pacific Television Conference, 6 June 1978.
8. Derfler, F. "The Ultimate Consumer Computer." Kilobaud Microcomputing, October 1979, pp. 94-96.
9. Doll, D. Data Communications: Facilities, Networks, and Systems Design. New York: John Wiley & Sons, 1978.
10. Edson, L. "Computers Come Home." The New York Times Magazine, 30 September 1979, p.84.
11. Isaack, J. "Standards for the Personal Computing Network." Computer, October 1978, pp. 60-63.
12. Issacson, P. "Food for the Home Computer." Datamation, November 1978, p. 188.
13. _____ "1979 - The Year of the Home Computer." Datamation, January 1979, p. 217.
14. Lesea, A. and Zaks, R. Microprocessor Interfacing Techniques. Berkeley: Sybex, 1978.
15. McGlynn, D. Distributed Processing and Data Communications. New York: John Wiley & Sons, 1978.
16. McLellan, V. "More Minis From Prime." Datamation, January 1979, pp. 55-57.
17. Minicucci, R. "Electronic Mail: The Facsimile Connection." Office Products News, July 1979, pp. 5-9.
18. NCR Technical Publications Department. NCR Data Communication Concepts. Indianapolis: Howard W. Sams and Co., 1978.
19. Prestel Users Guide and Directory - A Complete Guide to Information Providers and Services Available Through Prestel. July 1979.
20. Shrage, M. "McLean Firm First to Offer New Computer." The Washington Post (Business and Finance section), 16 December 1978.
21. Telecomputing Corporation of America. The Source System User's Guide. McLean, Virginia: The Company, 1979.
22. Tomita, T. "Introducing the Captain System." Pacific Telecommunications Conference Proceedings, 8 January 1979, pp. 4A-23.
23. Wilkinson, M. "Viewdata Systems: British Skills Lead the Way." Financial Times (London), 9 May 1979, pp. 19-26.
24. Youngblood, G. "The Mass Media and the Future of Desire." The Coevolution Quarterly, Winter 1977/78, pp. 7-19.



# THE ELECTRONIC SANDBOX

By
Mark Cummings, Asst. Prof.
BCA Dept. SFSU
1600 Holloway Ave., S.F., Ca. 94132
(415) 469-1787

Edited By
Georjean Frank, Mgr.
Satellite Broadcast Sales, W.U.

"The past is a dream, and the future is a mist. Great moments pass away. What amuses man is to be puzzled - not to know for a while the outcome. Men like to be unsatisfied." Mohammad Ali

Two-way CATV systems have been in existence in this country for at least fifteen years and yet, until recently, there have not been any strong two-way interactive service offerings. There have been satellites. First, government satellites and then commercial satellites. Western Union, RCA and AT&T operate satellite systems, but there are no two-way interactive programs. There are many new communication media, but they are generally inaccessible for general home or office use. Telephones, telex machines, acoustic couplers, and data access arrangements are most commonly used. What is missing is a universal home/office terminal device - a low cost terminal device that would be as prevalent as the telephone and would have the following capabilities:

- Accept alphabetic input and display
- Accept numeric input and display
- Color graphic input and display
- Motion video input and display
- Audio input and output
- Soft copy storage (i.e.,erasable storage such as tape)
- Hard copy output
- Access to computing power
- Two-way communication capability (providing audio/video and data communications)
- Ability to translate between audio, video and data

These currently available devices can be arranged to provide a universal home/office terminal except for one capability - hard copy. Hard copy output is important but it is not yet economically feasible. Existing printers are either expensive (approx. $3,000) or lack acceptable print quality. Matsushita has announced a facsimile system which they propose to use in conjuction with standard broadcast television signal. This device might provide a low-cost, hard copy output. Also, KDD announced recently that they have a new high speed facsimile device. there are also rumors of upcoming low cost electric typewriters and daisy wheel printers that can be machine driven.

An estimated cost for this equipment is depicted in Illustration 2. Prices are retail based on an average of the three most popular systems. Depending upon the equipment arrangement, the cost may vary by 10% one way or the other. If it is packaged by one manufacturer,

the price will go down and we can expect real cost declines. If the equipment doesn't come in one package, but several different pieces, this raises the price. But the bottom line is that for about $1,800 today, a universal terminal capability is available for homes and offices.

These devices, at first, will be programmed with recordlike functions, something that can be physically transported on an audio cassette tape, a betamax tape, and video discs. These are currently on the market and in use today (Illustration 3).

The potential power of this device, however, is not as a record player but rather as an interactive communication terminal. With the capability of satellite networking, and some form of a broadband two-way local distribution system, the possibility to do some fairly amazing things exists. Illustration 4 depicts the way satellite systems operate today. There is an uplink to the satellite and a return link using terrestrial telephone lines, utilizing standard video terminals and touchtone telephones and a typical studio complement to act as terminals for this system. For example, the Veteran's Administration currently operates a program in 33 veterans' hospitals in 17 western states using the satellite for in-service education of medical personnel. Students have a television set and a modified telephone receiver. Their responses are carried back through landline telephones to the originating center where their questions are answered by the speaker. However, a system which is fully two-way, as shown in Illustration 5, is the ideal. The telephone lines have been eliminated, the universal terminal has been added, and the satellite is now a two-way broadband link. Envisioned is a two-way link from the satellite earth station to the terminal via a cable television system. Cable is, of course, not the only broadband communications system that might be used. Use of fiber optics is possible, but cable is here today. Another potential local distribution technique is radio. Currently there are groups working with Fm SCA, Fm prime channel, TV Vertical blanking portion, and microwave techniques. Telephone lines are also available for local distribution but cost and bandwidth limitations are severe.

Potential Uses: With a system like this we have the potential capability to completely revolutionize the way we go about manufacturing and distributing symbolic information in our society. For example, a presentation by an instructor at one facility could be delivered to small groups in classrooms or special purpose communications rooms in other locales. What is far more interesting is what happens when large numbers of people are involved in a network. Instead of 20-30 people around a conference table or participating in a teleconference, multitudes can participate. They can remain where they are and information can be distributed by this combination of satellite long haul and two-way interactive cable local distribution system, in such a way that there is interaction similar to that which occurs in smaller groups.

One example of how this might be accomplished is shown in Illustration 6. The lecturer has prepared three to five minute lecture segments which are followed by multiple choice/true-false questions. Students would answer these questions with the digital portion of their universal terminals. Their answers would be analyzed by a computer which would flash a signal of general comprehension or non-comprehension to the lecturer. The lecturer would then either review the material just presented,

or go on to new material, depending upon the level of comprehension. Eventually, it would be possible to have videotaped segments selected by a computer. The instructor would know what areas students were apt to have trouble with, and pretaped review segments for those areas could be generated. The computer could switch between these pretaped segments based on the unique needs (comprehension level) of the particular audience tuned in.

Teaching of manual or aesthetic skills (eye, hand, mind cordination) is also possible within this system. In the instance of music instruction (Illustration7) the student, sitting in front of the universal terminal device, could be able to see the music in score form, hear the teacher's explanation and hear and watch his performance. On the other hand, the teacher could be able to hear the student's performance. The teacher could select an indivisual or groups of individuals at remote locations to listen to.

Other Potential Public/Private Usage Examples: Consumer organization is possible. A system could be developed whereby you dial a service bureau that enters test results into a data bank on specific consumer products. A charge would be applicable to access this data. An organization that tracks consumer complaints about bad treatment/service might also be established.

Political organization is possible. The potential exists for redistribution of power through change in access to information and through individual access to information services. What is the voting record of your Congressman? Do you know? Do you know how you can find out? If it's election time, community groups that keep records/tallies of these statistics can be called up, and an index prepared based on an averaging process. This average is also calculated for Congress as a whole, thus a comparison of your representative with Congress can be made. If it's not an election year, these indexes are not reported. However, there is no reason that this couldn't be available as a service which would be paid for as it is used.

Cashless society transactions - electronic funds transfer services.

Newspaper type services. It would be possible to subscribe to a news service and call up a picture on your screen of a newspaper page.

Electronic marketplaces for shopping - retail sales. A system whereby you would see products displayed on your terminal and you could order them.

Conventional entertainment programming.

Message recording services - telephone answering services.

Secretarial services.

Word processing services.

Electronic mail services.

Simple computational problems - home budgeting, ordering, stocking.

Heat, light and appliance monotoring and control.

Security services.

Daily calendar.

Acess to libraries - those libraries which exist today, or they could be specially developed libraries (i.e., consumer information).

Access to legal/medical advice. It would be possible to provide direct medical services over such a system, as has been done in Alaska and Canada.

A generation or two from now, people are going to look back at us and say, "you know, those guys were sitting down playing with all that stuff and having a grand old time, and they didn't know what they were doing at all, but somehow they got us

here." That's where I think we are now - children playing in an 'electronic sandbox'. We really don't know what we are going to do with all the 'sand', but the game is real important. It is of consequence for us, and our children, because this is how we learn and ascertain the capabilities of these devices.

I believe the introduction of this equipment and the link with communications can dramatically change society. What will these changes be, will we be able to control these changes, and what will the consequences be? One of the possible changes is the nature of access to information. Ten or fifteen years ago, access was impossible because those having information had top security clearances and weren't allowed to talk to others. Because it was all tightly controlled and done on big, expensive machines funded by the federal government, inquiries were prohibited unless you were a corporate executive or a military officer of a certain rank. If questions were asked, there was no one who was capable of answering them.

Now we do have the capability to talk amongst ourselves. Equipment affordable by individuals is available and knowledge is in the hands of a growing number of people who feel impelled to share it. This access to information can have significant consequences. Are we going to construct a system that further increases the amount of alienation that we have in our society, or will we construct a system that is going to promote socialization? How is that system going to effect the way that we look at ourselves, at our interactions with each other, at our world and the society we live in? Is it possible to make these relationships healthier, more fulfilling? I maintain that it is, but that prudence is necessary.

The Commerce Department, Office of Telecommunications, did a study (GPO #003-000-0058-2) which showed that in 1967 50% of the gross national product occurred in the information economy. In other words, 50% of all economic activity consists of the collection, manipulation, and distribution of information. So, the capabilities of this universal home/office terminal and associated interactive communication system can have an impact on general business as well as political and cultural activity.

With the universal home/office terminal device it will not be necessary to organize our lives around large, central offices in city cores, and commuting in high density traffic and smog. The power of such a system can free us from moving ourselves and physical objects when what we really are doing is moving information. But where will we be? One possible form that could evolve is individual people sitting with singular personal computers in private cubicles called apartments where 90% of their business, social and cultural interaction would occurr. Such an outcome would promote isolation and increasing alienation. A more humanizing outcome could be achieved through organizing around the concept of community communications centers. Illustration 8 is one possible configuration for such a center. There is a children's play area in the lower right-hand side. This play area is modeled after the Marin Computer Center. It would provide traditional indoor and outdoor play areas and equipment plus universal terminals configured for games, sport competition and, in conjunction with the community classroom, education. Above that, a high speed fax system and a high speed printer with a table in between for collating on. A teleconferencing room provides space for live meetings for

small groups of people or larger meetings via teleconferencing using video and audio facilities. A minicomputer controls this array of universal home/office terminals. There are two types of access: public booths which are available by the minute, hour, day; or private offices leased on a monthly basis. An electronic library is also included. One of these community centers per residential square block with terminal devices, organized education, economic activities, political activities might serve 250 people encouraging socialization by allowing to work together. All segments of a community could mingle freely while pursuing their daily activities.

I don't maintain that with community communications centers we won't have personal computers operating as universal terminals in our homes, or that we won't have some traditional offices. I think we will, but an argument can be made for high speed printers, high speed fax, teleconferencing, and other activities that can take advantage of economies of scale, combined with improved quality of life, to justify community work spaces. When this happens, I think we have made an important contribution to changing what I see as a trend toward growing alienation and hostility.

## Public Policy

Given the economics, the benefits and our 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? Will communications continue to be regulated? Will computing continue to be unregulated? These are questions which need to be addressed.

Currently, public policy in the communications and computer fields is expressed through governmental and bureacratic foundations with little or no public input. If policy decisions in these areas are going to shape the nature of our world and if we believe in the tenants of democracy, we must have informed public participation. By a process of natural selection, if you are reading this, you probably constitute a member of the "informed" public. Let me outline some of the public policy areas you might choose to participate in.

## Standards

The communications industry has evolved a set of communications and protocol standards. These are generally adequate and should be supported. Those most useful at this time appear to include RS232C, Bell 103, and CCITT X.25. Digital broadcasting is creating a need for additional communications standards. In this area, we suggest that the ANPA (American Newspaper Publishers Association) High Speed News Standard and the BBC's CEEFAX standards might provide good models.

On the terminal side, however, there is much more work to be done. Although Basic is almost universally available on micros, each vendor has their own non-compatible form. The graphics and audio generation languages tend to be the most idiosyncratic. Although there appears to be a trend toward standardizing around broadcast color television receivers as the output device, there are notable exceptions. In order for personal computers to interface with analog video systems a standard means of addressing frames of video information is needed. We suggest that the SMPTE (Society of Motion Picture and Television Engineers) Time Code might provide a good foundation.

## Privacy

In order to take full advantage of a universal terminal system an efficient addressing scheme is necessary for unattended operation. Packet switching protocols such as X.25 provide this capability. However, such approaches have one major drawback. In packet switched systems, information is broken up into units and assembled into packets by adding message headers. Assembled packets are routed through nodes and carried over high speed trunks allowing a vast number of users to share single trunks. The problem lies in the fact that all messages contain the address of both the calling and the called party. Even if data encryption is used to scramble the information in the packet it is possible to map the communications activity of all users. The possibility of monitoring who is talking to whom, even if we don't know what is said, creates the potential for invasion of privacy and a threat to personal freedom. Much attention has been focused on the problem created by monolithic personal information data bases. Similar efforts are needed in the area of communications protocols. One solution might be to devise packet switching protocols which contain only the address of the receiving terminal in the message header and allowing the sender to put his address in the data field (encrypted or non-encrypted) or leave it out altogether.

## Regulatory Barriers

There's one last thing that should be noted. The reason that many of these services haven't been offered earlier, is because of the regulatory and legislative bars to hybrid data processing and communications systems offerings. Righ now the communications act is being rewritten. There is proposed legislation on computer fraud and there is an opportunity as a result of these actions to remove the bar to one firm offering hybrid data processing and communications. Given the technology existing in the world, this effort should be supported.

## Conclusion

We are entering a period of dynamic change based on technological advances in micro electronics and communications technologies. The magnitude of these advances has to a large degree unvested the interests that have maintained the status quo, thus creating a plastic environment, an "Electronic Sandbox". This period will be relatively short lived. Soon new interests will become vested and the patterns built in "sand" will become locked in "concrete". Now is the time for creative play; for experimentation with new alternatives; and for choosing those alternatives which will foster a more humanizing, more socializing and more democratic society.

CURRENTLY AVAILABLE DEVICES THAT CAN BE ARRANGED TO PROVIDE THE CAPABILITIES OF A UNIVERSAL TERMINAL:

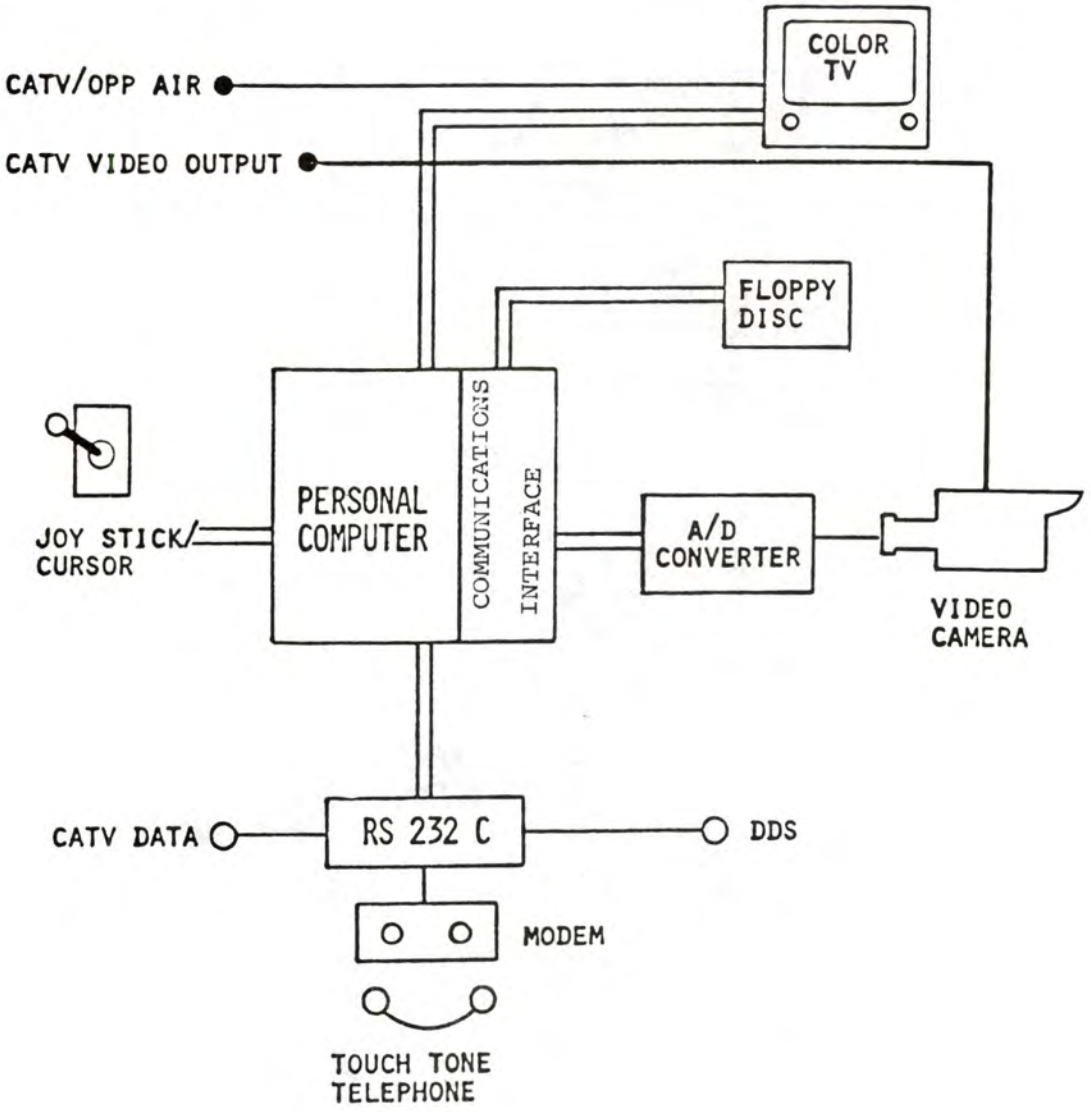



ILLUSTRATION 1

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.....$ | ? |

ILLUSTRATION 2

PUBLICATION IN SOFT COPY FORMATS OF:

-COMPUTER AIDED INSTRUCTION (CAI)

-GAMES

-CULTURAL EVENTS

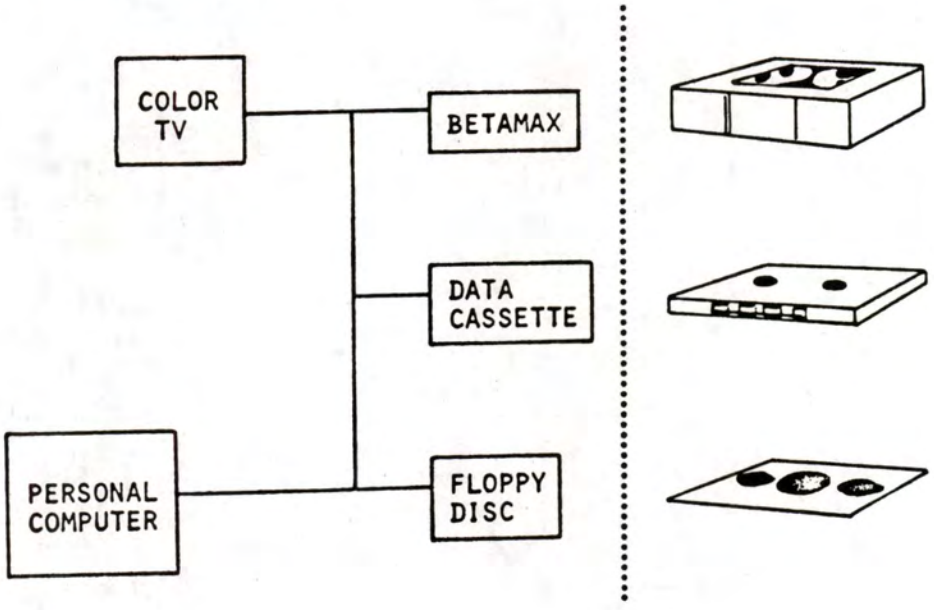



ILLUSTRATION 3

## BASIC SYSTEM

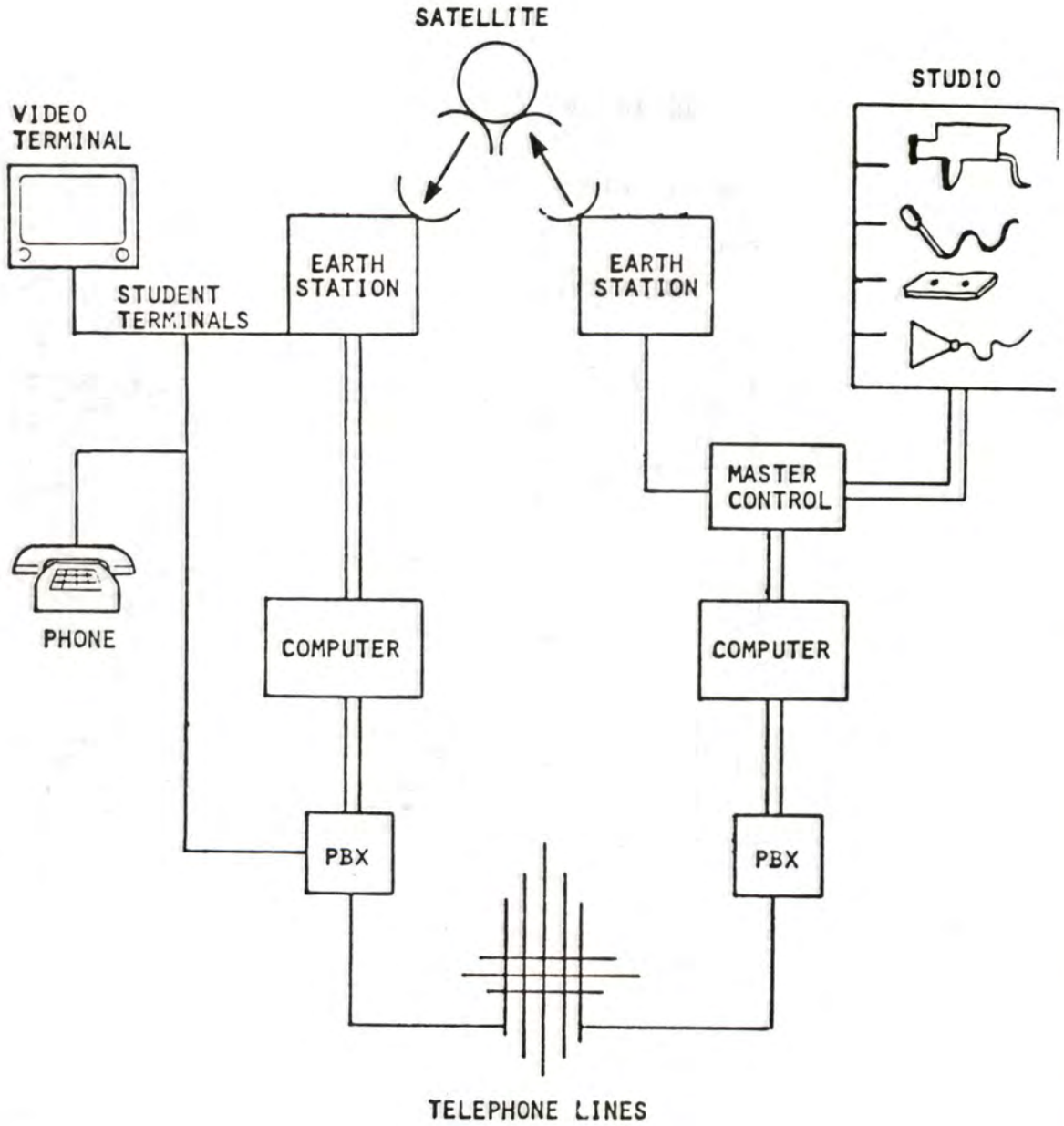



ILLUSTRATION 4

FULLY DEVELOPED SYSTEM

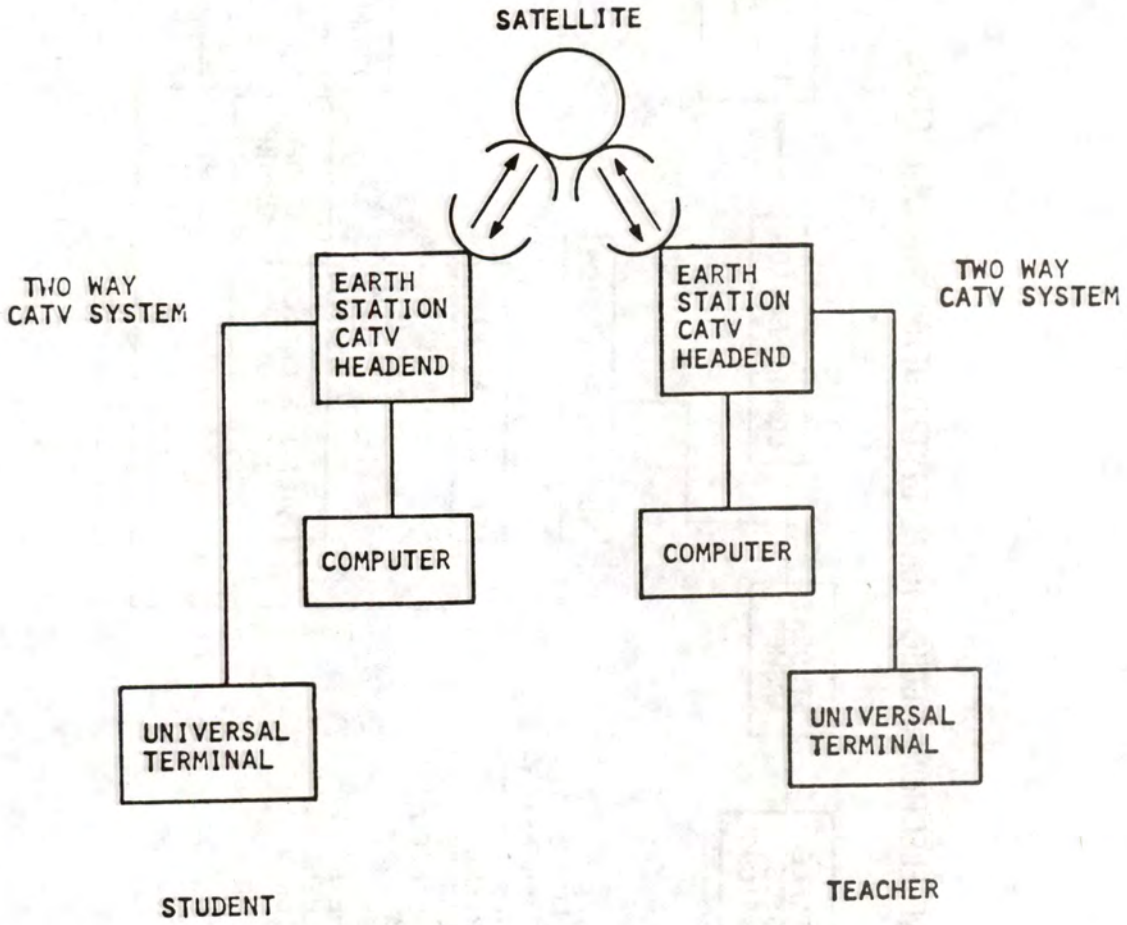
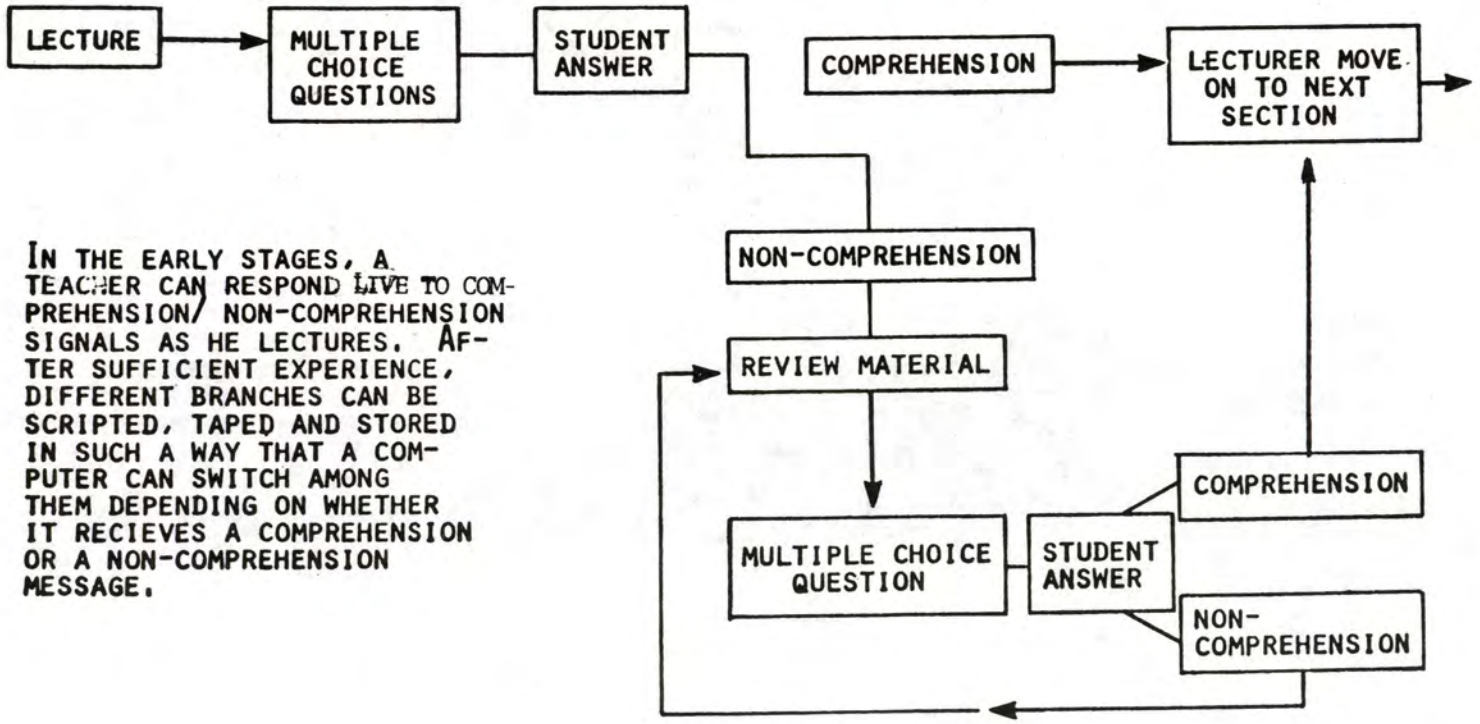



ILLUSTRATION 5

## INTERACTIVE ELECTRONIC EDUCATIONAL DISTRIBUTION SYSTEM LECTURE

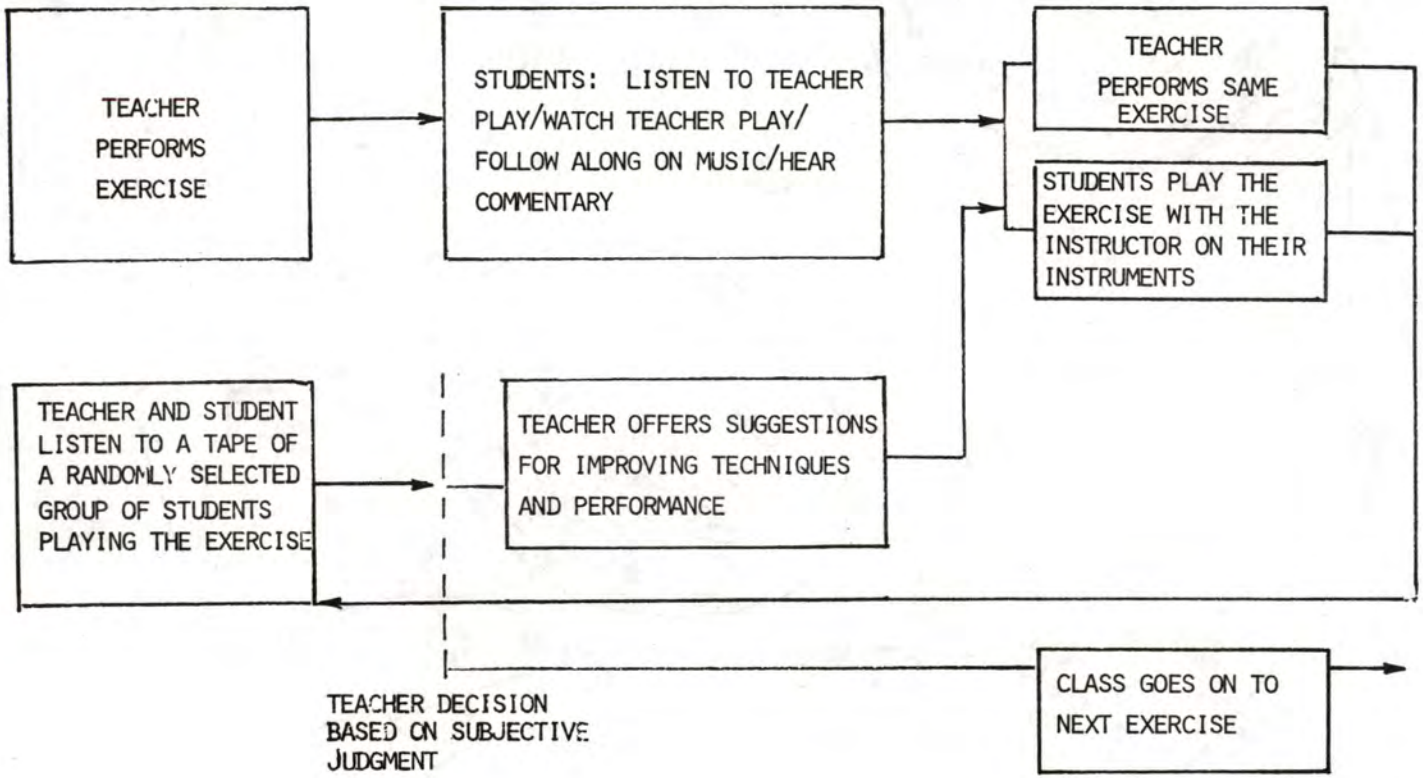



IN THE EARLY STAGES, A TEACHER CAN RESPOND LIVE TO COMPREHENSION/ NON-COMPREHENSION SIGNALS AS HE LECTURES. AFTER SUFFICIENT EXPERIENCE, DIFFERENT BRANCHES CAN BE SCRIPTED, TAPED AND STORED IN SUCH A WAY THAT A COMPUTER CAN SWITCH AMONG THEM DEPENDING ON WHETHER IT RECIEVES A COMPREHENSION OR A NON-COMPREHENSION MESSAGE.

ILLUSTRATION 6

ILLUSTRATION 7

## INTERACTIVE MUSIC PERFORMANCE INSTRUCTION



COMMUNITY COMMUNICATIONS CENTER

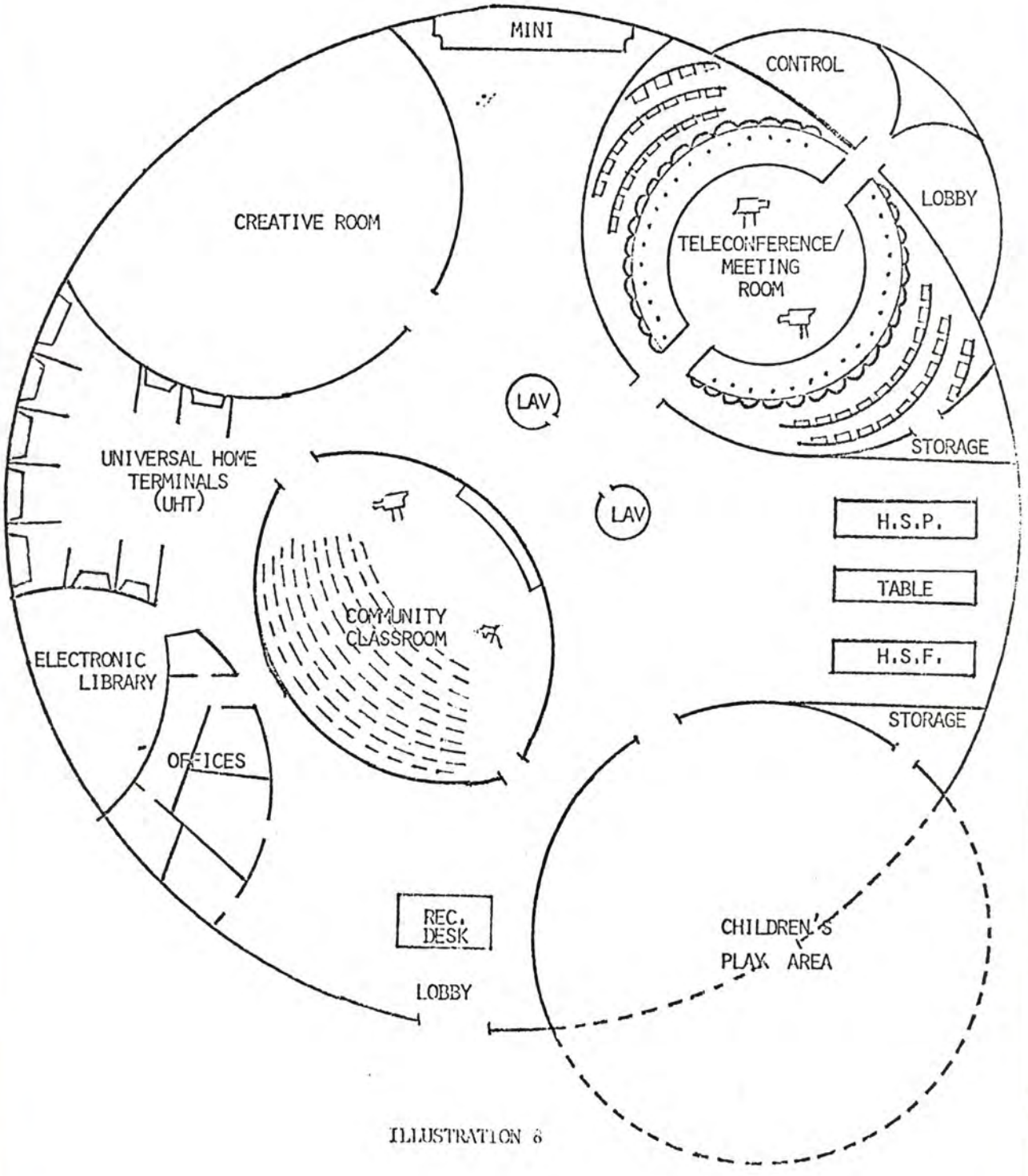



ILLUSTRATION 6

# ENERGY MANAGEMENT FOR THE HOME

# WITH THE HELION MICROMANAGER

Jack Park
President
Helion, Inc.
Box 445
Brownsville, Calif. 95919
(916)675-2478

## Introduction

At the 4th West Coast Computer Faire, I discussed many of the attributes of a home energy management computer and their relationship to energy conservation. At that time, my company, Helion, was in the process of delivering battery-powered field data acquisition systems for wind energy studies in the Mojave Desert being conducted by Atmospheric Research and Technology, Inc., of Sacramento, for the California Energy Commission. A photo of our Model 100 μ-Logger is included here. A logical outgrowth of the μ-Logger project is Helion's MicroManager (Trademark), a dedicated CMOS process control computer, developed specifically for energy conservation in the home. This paper describes the prototype version, two of which are installed at this writing.

### MicroManager

The MicroManager is a complete computer system dedicated to performing tasks related to home energy management and security. The prototype version has two functions:

- It performs home energy management according to a "home profile" provided by the dweller concerning week day schedules as well as week-end requirements.
- It performs a modest amount of home security control.

## Home Energy Management

The MicroManager serves as a home energy manager by automatically controlling energy consumption according to a user-input description of the dweller's daily requirements. That is, at initialization the MicroManager requests information such as desired temperatures throughout the day (along with acceptable variations) and time(s) of day for peak energy usage in the dweller's normal schedule. This provides a "home profile" which tailors the MicroManager to the user's individual requirements. Once initialized, the MicroManager automatically manages energy consumption in the home.

The impact of the MicroManager on daily life in the home depends on the way that the user configures the system. At one extreme, the only perceptible difference to the dweller is the reduction in energy consumption (as reflected in utility bills.) At the other extreme, the user may choose to adopt a new, energy-conscious lifestyle and may use the MicroManager to enforce miserly consumption. Of course, manual override of the system is allowed at any time. Also, whenever necessary, the user may reset the "home profile" to reflect changes in the daily schedule.

Energy management is effected via up to 15 control channels. These channels communicate with control points throughout the house, and are used to turn appliances such as heaters and refrigerators on and off. The MicroManager is programmed to control each appliance such that the appliance is functioning optimally when the need for it is great (according to the individual "home profile") and is functioning at energy-efficient levels otherwise.

At each control point, there is a small receiving device which plugs into or replaces existing electric outlets. Some receiving devices allow the current to be turned either ON or OFF. A special receiver is available for lamps which permits them to be dimmed (as well as turned ON or OFF.) Each such device

receives and responds to housewire FM encoded signals sent specifically to it from the MicroManager. The use of these receiving devices means that the MicroManager may be placed in any home, with no special rewiring being required. The prototype MicroManagers use the BSR X-10 system, which is available at Radio Shack, Sears, or by mail order. Eventually, an industry standard communication protocal may be developed that will ease product adaptability to various home energy systems.

Version 0 (prototype) software pre-dedicates each of the 15 control channels to specific tasks. As an example, one channel is dedicated to controlling an active solar heat system. Four channels are dedicated to "heating mode" control and two are dedicated to "cooling mode." Four channels may be used for "alarm clock" (time only) control functions. The remaining channels presently are dedicated to home security.

Any given channel may communicate with one or more control devices. Thus, for example, if one of the time-only channels is used to sound a wake-up alarm, it may also start the coffee pot and other appliances.

### Energy Conservation

As an example of how the MicroManager prototype is used for energy conservation, the hot water heater is turned off at night by its control channel. The heater is not burdened with maintenance of a high water temperature when such is not needed. First thing in the morning, however, water heater control is returned to its built-in thermostat.

The author's house is heated by solar and wood, so the usual night set-back thermostat channels are not used, though they are available. Planned, but not in use at this writing, are insulated window curtains which are opened and closed by subroutines that use time and temperature to make command decisions.

### Energy Management

The prototype is presently used to reduce peak electric demand by holding nonessential applicances off duri g user-defined "peak" home hours. These hours usually occur at breakfast and dinner during weekdays, and at each meal during weekends. During these hours, refriegerators and freezers (TV sets, electric hot water heaters, ect.) may be held off. The net effect is to reduce the peak electric power demand from the home. The various time-only channels are available for automatic control of alarms, electric blankets, coffee pots, sprinkler systems, and a host of other devices.

### Solar Heater Control

The diagram illustrates the mechanics of the hot water solar heater controlled by the MicroManager. It is an active system with, at present, a simple two-state (on-off) control. Continuously variable control can be added by changing the program and the control point. As a developmental tool, the user initializes the temperature differential used by the algorithm.

## Home Security Control

In the prototype version, the user may request that (during specified hours) lights throughout the home be turned on and off in a random pattern that suggests the occupant is at home, thereby discouraging a would-be intruder from attempting to enter. Also, the user can request that porch lights be turned on each evening at a specified time.

Although the amount of home security control is minimal in the initial version, the protoytpe MicroManager is designed such that additional home security control mechanisms can be integrated and tested at any time.

References

1. Farrand, F., "A Real-Time Operating System That Specializes In Home Energy Management", The Best Of The Computer Faires, Volume IV, pp. 132-138.

2. Park, J., "Overview of Energy Conservation Possibilities Using Home Computers", The Best of The Computer Faires, Volume IV, pp. 114-120.

Photo of Helion Model 100 Intelligent Field Data Acquisition System . MicroManager is a direct outgrowth of Model 100 technology.

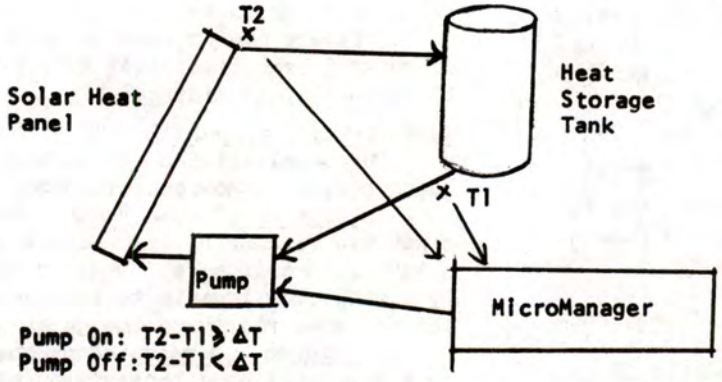



Diagram of the Solar Heat Control portion of Helion's Prototype MicroManager.

# MICROCOMPUTER-ASSISTED AMATEUR ASTRONOMY

Sidney Levin,M.D. , Astronomical Society of the Pacific
700 25th Ave., San Francisco, California 94121

Introduction:

The amateur astronomer's main activity at the telescope is to seek out and identify numerous celestial objects within the limitations of his optical (or radio) equipment & of existing "seeing" conditions. Objects are found by their celestial coordinates (right-ascension, equivalent to longitude, declination equivalent to latitude). These data are listed in many standard reference works, and of course designate an enormous number of stars, double stars, clusters, cluster galaxies, etcetera.

Dark adaptation of vision is a sine-quanon for most effective use of the telescope & and the necessity to illuminate and leaf through pages of hard copy is a tedious and time consuming chore. The microcomputer CRT set up on or adjacent to an observation deck, is an ideal solution for selected storage, rapid retrieval, and minimal degrading effect on the dark adapted eye.

In addition to location data, other useful programs include; 1. Retrieval of events-dates, and times (Jupiter satellite eclipses, planetary oppositions). 2. Translation of known celestial coordinates to navigational ones (altitude and azimuth). 3. Software creation of a sidereal clock, and 4. Educational programs to illustrate principles of astronomy and physics.

The author uses a TRS-80 model 1, level II computer with 32-K of RAM and single disc drive. Programs are in Basic and quite simple, though much of the sophisticated capability of the computer is utilized ( trigonometric functions,real-time clock, and graphic display). All of the programs were originally stored on cassette tape, the only disadvantage being slow loading.

Finding Coordinates of Celestial Objects:

1. Messier Catalog: lists data for the 109 "Nebulous objects" cataloged by the famous 18-th Century Comet Hunter. These are of greatest interest to the amateur because of suitability for small telescopes. Storage as a subscripted array allows display by catalog number, type of object(e.g."Globular Clusters") or as a list of M-objects visible during a season (e.g.winter) via a right-asension limits sort.

2. Constellation Catalog: displays data for the most interesting and suitable objects in a selected constellation, similiar to the monthly constellation presented in Astronomy Magazine.

3. Favorite List: A disc based file of 103 famous and favorite objects, each displayed as a record showing coordinates, basic astronomical data, and interesting commentary.

Many programs of this type could be tailored to the special interest of the astronomer (e.g.Double stars, variable stars, etc.).

Horizon Coordinates:

Converts right ascension and declination coordinates of stars and planets(avialable from astronomical publications or from the Nautical Almanac) to altitude and azimuth. This allows an observer to easily locate an object from compass direction and height in the sky. The program allows for input from Navigational data tables as well as astronomical ones. Rising & setting azimuths are also displayed.

Retrieval of Event-dates and Times:

Displays dates and/or predicted times of events, such as Jupiter satellite eclipses for the year, Heliocentric longitudes (hence opposition dates) of the Planets, Meteor showers, Variable star Maxima/Minima.

Sidereal Clock:

Sidereal time is based on the time interval for a complete rotation of the earth referenced to star posistions, roughly 4-minutes shorter than the sun-referenced day. Many telescopes have a clock setting circle which utilizes hour-angle distances from sidereal time to find objects.((Sidereal time-right ascesion)=Hour Angle))

Software design takes advantage of the computers' real time clock by making corrections to display sidereal time.

Educational Programs:

The possibilities for instruction of young people or novice astronomers are legion.

1."Find M-31" is a graphic demonstration naked-eye finding of our nearest galaxy-neighbor in Andromeda. M-31 is demonstrated by moving from Pegasus to Andromena, and from the Key star Mirach to the galaxy.

2. Taurus Supernova shows how to calculate from published photographs of the Crab nebula, the probable year of the Supernova (1054 A.D.) and from published spectroscopic Doppler shifts, the rough distance of the nebula.

3. Galileo's Great Experiment simulates the action of gravity on bodies in "free fall" (the heuristic origins of the "relativity" of motion)

and the proportionality between mass and gravatational attractive force.

5. Einstein-Lorentz Transformation illustrates a simplistic derivation of the relativistic constant ( $\sqrt{1 - V2/C2}$), and illustrates time dilation, Mass-inertia increase, and length contraction with increasing velocities.

Future Trends:

Output devices (encoding of telescope shafts) to allow "push - button" setting of the amateur telescope on the desired object are just now appearing, some commercially available. Programs for correcting errors in equatorial telescope alignment are available for calculators. Computers allow construction of simpler, sturdier, and less expensive (alt-azimuth) mounts as compared to the traditional equatorial mount (as in recently constructed great telescopes).

References:


1. Mullaney, J. and McCall, W. The Finest Deep-Sky Objects. Sky Pub. Corp. 1972.

2. Burnham, R.; Celestial Software Part 4: How To Calculate Horizon Coordinates. Astronomy Magazine, 1978. pp 50-51.

3. Brown, Sam. All About Telescopes: Edmund Scientific Co. 1976, pp 28 - 29, Vol.6, No.12, December 1978 pp 378-382.

4. Gingerich, Owen, Laboratory Exercise in Astronomy; The Crab Nebula. Sky and Telescope Magazine Vol.34, No.5 November, 1977, pp 378-382.

5. The Nautical Almanac; U.S. Government Printing Office, published annually, Washington, D.C.

6. Percy, John R. Editor; Observer's Handbook. Royal Astrnomical Society of Canada/University of Toronto Press, published annually.

# WRITING AND NEGOTIATING THE VENDOR'S SOFTWARE LICENSE CONTRACT: LET'S MAKE A DEAL

Joseph R. Igelmund
Attorney at Law
605 Market Street, Suite 1425
San Francisco, California 94105
(415) 392-2811



Introduction. The writing and negotiation of his license contract are serious matters for the software vendor. If handled thoughtfully, they can enhance his business prospects tremendously and rival the best prescription in helping him sleep well at night. If handled otherwise, they may cause one heck of a headache and even loss of the vendor's business.

Today I am going to discuss when a license contract may be appropriate for a software vendor; considerations in developing the contract; the vendor's decision to negotiate concessions in the contract demanded by a user, and some ideas to facilitate the negotiation process.

Two caveats are in order. First, there is a lot of material to cover in the brief time that I have. An overview obviously is required. It is my hope that this overview of closely related areas will be valuable particularly to those vendors in the audience considering the subject for the first time. Second, the vendor's license contract, should he choose to use one, is not going to remove the vendor's problems forever more. He may still be sued by an unhappy user. Worse yet, he may be hit for a big judgment by an unhappy user. However, the license contract, if well written and utilized appropriately, will eliminate many disagreements and help assure that the vendor is positioned in the best possible light when they do arise.

When a license contract may be appropriate. Let me first confirm what I mean by a "license contract." A contract is a legally binding agreement between two parties, under which one gives something in exchange for something from the other. For the software vendor, this contract is usually a license contract, under which he licenses another to use his software in exchange for a license fee. Certain contracts cannot be enforced unless in writing. The perpetual license agreement, which is the habit in the software business, must be in writing to be enforceable. The writing may be as simple as an invoice, or it may be a multitude of pages, as long as it includes the legally essential terms and is signed by the party who is being held accountable.

My subject today is not an invoice. Rather, when I say "license contract" or "contract", I mean a document to be signed by vendor and user which describes the full nature of their transaction and covers such topics of concern to the vendor as the type of license conveyed, payment schedule, proprietary nature of the software and obligation of the user to protect its proprietary nature, confidentiality of the idea, and servicing if appropriate.

Since repeated writing of a license contract can be a time consuming and expensive proposition, this contract normally becomes the vendor's "form" or "standard" contract. He has devoted much thought to its construction and content, and so it will be the basis for all transactions in which the vendor desires to have a license contract.

The purpose of the license contract is to protect the vendor's rights and limit his obligations. In order to determine when a license contract may be appropriate, the vendor should ask two questions. What may I lose if there is no contract? Also, what may the user lose, and therefore want from me, if there is no contract? These are, of course, realistic questions because periodically in any vendor's business, deals will go bad. It may be because the user is not operating the software correctly, did not listen to the vendor, or is acting irresponsibly. It may be that the software is not doing what the vendor knows it can do. Whatever the cause, deals will go bad. When that happens, someone may suffer significant dollar loss and want to be made whole.

The risk of significant dollar loss by vendor and user is common

especially where the vendor writes small business and commercial applications. My remarks today therefore have particular relevance for that class of vendor. For him, the contract and what the parties commit to under the contract are even more important.

Let me give an example to illustrate each of these questions. What may the vendor lose? Assume a misunderstanding about the type of license conveyed. Vendor thinks that he has conveyed a nonexclusive license to user. User thinks that he has received an exclusive license. Vendor will lose substantial revenue if he cannot license the software to others. If the license gives user an edge over his competition, you can bet that he will try to protect that exclusivity. The result is a rough and tumble dispute that also may cost vendor substantial legal fees and costs, whether he wins, loses or settles.

What may the user lose and therefore seek from the vendor? Assume that user installs vendor's software to process employee salary payments. A design deficiency in the software makes it incompatible with user's new terminals. Vendor believed that they would be compatible and represented same to user. While vendor works on the design problem, user processes salary payments manually (with substantial overtime expense), but is unable to get some checks out on time. Several employees complain to the labor board and, in view of user's previous labor problems in other areas, proceedings against him result. User demands restitution by vendor for damages and expenses. Their agreement does not limit the kind of damages for which vendor is liable, nor the amount. Again, the result is a dispute that may cost vendor substantial damages, legal fees and costs.

The vendor's reputation also may be impacted by a contract. The user has certain expectations of the vendor. When those expectations are not met, no matter who is to blame (and the user is not going to blame himself), the vendor's reputation suffers. This is especially important to a vendor concentrating on a limited market, such as accounting, insurance or medical. Users communicate their experiences, and comment critical of a vendor can have serious, long term implications for him. If a contract can help avoid disagreements and loss of reputation, it is something to consider.

Finally, the vendor's contract can further his business. The well written contract shows that the vendor knows what he is doing. I can look at a vendor's contract and assess its character quickly. Many users, particularly sophisticated users with the big bucks, can do the same from a business point of view. In a competitive environment, this professional image may help the vendor close a deal.

Some of you may disagree. After all, the contract is a potential roadblock to making a deal because the user must agree to its terms. Furthermore, users love to moan and groan when they have to review the vendor's contract. Having been Senior Counsel for a major Bay Area user and in charge of their DP contracting, I suggest that vendors not take that moaning and groaning too seriously. Most users fully understand that, where the vendor and user each know what they are doing, the professionally prepared, well written vendor contract is the way the game is played.

Let's summarize as to when a license contract may be appropriate. The vendor should ask two questions. First, what may I lose if there is no contract? Second, what may the user lose, and then want from me, if there is no contract? In addition, the vendor should examine his vulnerability to loss of reputation and consider whether a license contract would benefit his image with users.

Developing the optimum license contract. When the vendor decides to develop a license contract, he must determine the kind of contract it will be.

The vendor wants a contract that suits him and his business. There are a couple of characteristics that the optimum license contract will have. It will be as brief as possible under the circumstances. The vendor does not want a ten page contract if two pages will do, nor does he want two pages if his business suggests more comprehensive coverage. The longer the contract, the more opportunity for the user to be upset with terms.

The contract should be readable. I do not mean legible. I mean understandable. The more readable it is, the better the user can digest it and the more willing he is to sign it. Generally worded provisions offering broad protection to the vendor should not be rejected, of course. However, clever, cryptic phraseology should be avoided. A number of users undoubtedly will object and, in this consumer-

oriented society, the courts may not enforce such trickery anyway.

These are characteristics of the optimum license contract. What provisions should be included? The vendor basically has three choices. He can give a little, give a lot or give something in between. I have seen all three. For example, there is what I call the "quivering contract". The user is told that he will get some software for his money but, after reading the contract, he no longer is sure. Only the most naive user can sign the quivering contract without developing the most severe quivering of the limbs. This raises a practical problem for the vendor's accountant because the user's signature often is illegible.

The other extreme is what I call the "I'm OK, you're OK contract". Paragraph one recites that vendor is licensing software to user for a specified fee. The software in question often is not specified. Paragraph two recites that vendor and user are nice guys. Their signatures follow.

Most vendors' contracts obviously will fall somewhere between the "quivering contract" and "I'm OK, you're OK contract." To determine more precisely where, I suggest a two step process. First, reduce to writing a summary draft that reflects the most protective contract a vendor could hope for, keeping in mind the desirable characteristics of brevity and readability. Second, measure the vendor's strength in the marketplace, and develop a full, amended draft that reflects such strength. Let's review each step in detail.

The summary draft includes only concept, not the verbiage that is part of the final contract. It focuses the vendor's thinking on what is right for his situation. The license contract will be the basis of his legal relationship with each of his users. The further that the vendor departs from the most protective contract possible, the greater his risk. The summary draft, as a reference point, will help him comprehend clearly the risks he undertakes in the final contract.

The second step, measurement of the vendor's strength in the marketplace and appropriate amendments to the draft, is equally important. It will determine his risk in marketing software and viability of his contract as a marketing aid. If the vendor has a unique product and is assured of a ready marketplace for some time, he may have greater freedom to load his contract as he sees fit. If, however, vendor competition is tight in his marketplace, he may feel constrained to adopt a contract more attractive to the user.

I should emphasize here that a competitive marketplace does not preclude the vendor's aggressively developing his contract. He can be aggressive while opening the door to discussion of terms with the user when necessary. The vendor does not have to open the door fully, but just enough to get that user on his customer list.

How does one measure the vendor's strength in the marketplace? Here are six relevant considerations. First, how attractive is the vendor's product in comparison to his competition? The more attractive the product, the tougher the vendor's contract may be.

Second, what is the vendor's reputation in the marketplace in comparison to his competition? The greater his reputation, the tougher the vendor's contract may be.

Third, how sophisticated are the vendor's prospective users? Users will range from highly knowledgeable to less knowledgeable, but how sophisticated is the mainstream? The more sophisticated, the more careful the vendor's contract must be.

Fourth, what do the competitors' contracts say? Depending on the competitors' experience in the marketplace, their contracts may manifest what users generally will accept. I would not recommend pirating competitors' contracts, however. They are becoming increasingly sensitive about that sort of thing, and a "cut and paste" job is not dependable protection for the liabilities at stake.

Fifth, what fringe benefits or "throwaways" can the vendor offer prospective users that will cost little, or not obligate the vendor, but make the contract more palatable? Several of these can make the contract more effective.

Finally, the vendor should not disregard the power inherent in his position. Many users will sign the vendor's contract no matter how tough it is. They will sign it because they want the vendor's software and he has put that contract in front of them. Even the most sophisticated users usually accept the disadvantage of negotiating from the vendor's contract, rather than their own.

To summarize developing the optimum license contract: The contract should be reasonably brief and also readable. The contents may be determined by a two step process. First, ascertain the most protective provisions a vendor can hope for. Second, measure the vendor's strength in the marketplace and make appropriate amendments.

Negotiating the license contract. Now that the vendor has a license contract, the object is to get it signed. If the contract is well written and prepared as discussed, most users will sign it, no or few questions asked.

Some users, however, will not. Something in the contract bothers them and they will not sign until it is resolved. As to them, the vendor must make a decision. Do I attempt to negotiate a compromise, or do I not? The following observations may help to answer this question and, if the vendor chooses to negotiate, facilitate the process. For those vendors in the audience with substantial negotiating experience, my apologies if some of these observations seem elementary.

The advantage of negotiating is that the vendor preserves his chances of making a deal. There are, on the other hand, two basic disadvantages of negotiating. First, the vendor will lose some protection. His rights will have been reduced or his obligations increased under the contract with that user. This may affect his relationship with more than that user. Again, users tend to share their experiences in dealing with vendors. Other, prospective users may find out about these concessions and make similar demands of the vendor.

Furthermore, whether or not the negotiation results in a deal, it is going to cost the vendor time and possibly legal fees. Unless the negotiation covers only straight business issues, the vendor probably is going to want a lawyer involved at least for redrafting purposes. The contract is a carefully crafted device to protect the vendor. If it is going to be monkeyed with, it should be monkeyed with carefully. By straight business issues, I mean those within the peculiar knowledge of the vendor, such as price, payment terms and delivery of what when and where.

The vendor, like any businessperson, wants to limit his legal fees. There are ways to accomplish this in a negotiation. The vendor should use in the negotiation the lawyer who wrote his license contract. That lawyer already is familiar with the contract and knows the purpose of its terms. A new lawyer may do a fine job but, at least the first time, will cost more. When the vendor is looking for a lawyer to prepare his license contract, he should ascertain that the lawyer chosen will be available for negotiation assistance in the future.

Before the vendor brings his lawyer in, he should also make sure that the user knows what he wants. The vendor might suggest that the user put his proposal in writing "so that I can take it to my lawyer." Allowing the user to create the writing reduces the vendor's bargaining position, but the saving in legal fees probably is worth it to the vendor. If the user has house counsel, the vendor should make sure that user and house counsel are in agreement on what they want before bringing in his lawyer. Some house counsel are used in an overkill mode by management and can run up the vendor's legal bill easily.

Most importantly, the vendor should communicate fully to his lawyer all negotiating limitations in effect. For example, how important is this user to the vendor? If the vendor wants his lawyer to assist in negotiating, should the lawyer call it quits if negotiations cannot be concluded swiftly? How swiftly is "swiftly?" The lawyer then will be able to function most efficiently in representing the vendor's interests.

For the vendor who decides a user is worth the negotiation, here are some final thoughts that may help smooth the way.

The good guy/bad guy strategy will help move almost any negotiation. The vendor is the good guy and his lawyer is the bad guy. If the vendor cannot concede on terms, it is because his lawyer won't let him. Why the lawyer has such power over his client, the vendor, is not always clear. The beauty of this approach is twofold. The lawyer does not have to appear, indeed even exist, for the vendor to use it. In addition, if the negotiations become stalled, the vendor, since he is the good guy, is in position to mediate a satisfactory compromise between his lawyer and the user. Incidentally, users have been known to play the same game.

Second, the vendor must know that he can concede protection without

necessarily cutting his throat. Remember that this license contract reflects what he believes the marketplace will accept. What the marketplace will accept and what the vendor can live with are, of course, two different things. Concessions that are tolerable to the vendor will depend on the issue and the deal.

Third, the vendor must know that there always is a middle ground. Contract negotiation issues, including legal, are no different from any other business issue in this respect, and the vendor should allow no one to tell him otherwise. The vendor may not be willing to take the middle ground, or the user may not be willing. However, the middle ground is there if they are willing. It may take some ingenuity to uncover it. It may require focusing on the ideas or concerns behind the user's position, rather than his words. It is there.

Let's summarize the discussion on negotiating the license contract. The advantage of negotiating is that it allows the vendor still to make a deal. The disadvantages are reduction of vendor protection, and vendor time and legal expense. To save on legal fees, the vendor should consider using the same lawyer for drafting and negotiation, make sure the user knows what he wants before bringing the lawyer in, and communicate fully to the lawyer all negotiating limitations in effect. In conducting the negotiation, the vendor should consider the good guy/bad guy strategy, know that negotiation will not necessarily be fatal to his protection and that there always is a middle ground to agree on.

Conclusion. Writing his license contract, and negotiating it, require a lot of thought by the software vendor. It is worth it.

You may have gotten the idea from my comments that the well written license contract can both provide significant protection to the vendor and serve as a marketing aid. If you have gotten that idea, I have explained myself well. Handled the right way, the vendor's license contract can help him make the deal he wants. That's what it is all about, isn't it?

Thank you, Mr. Chairman.

# THE SOFTWARE JUNGLE: LEGAL PITFALLS

By
Raymond Karch, Attorney At Law
Opamp Building, Suite A
1033 N. Sycamore Avenue
Hollywood, California 90038
Telephone: (213) 464-4358

## Abstract

This article deals with some of the legal problems that are commonly encountered by software developers, distributers, and purchasers. Alternative methods of software protection are seen in the context of copyrights, patents, and trade secrets. Potential liability from software use is explored along with means to limit it.

## Introduction

In Greek mythology there is a detailed account of the god, Prometheus, giving the gift of fire to mankind. It was soon after that someone received the first burn. In a similar manner, the computer has proven its utility to humanity in numerous ways. The complex legal problems associated with computer use are now becoming more apparent in both the civil and criminal courts.

The high demand for applications software has resulted in a small army of independent developers and distributers. The computer industry has quickly reached the multi-million dollar stage. A single program can cost anywhere from a few dollars to a hundred thousand dollars depending on its complexity. It comes as no surprise that piracy of software is fast becoming a major concern to software developers who may have invested much time and effort in their creations. The question is not whether any protection exists for software, but which alternative offers the best protection.

As programs with real-life applications proliferate, it is conceivable that lawsuits will be filed against the original software developer for damages or injuries arising from the use of their programs. Unsophisticated consumers should be aware of the simple means they can employ to protect their investment in software when dealing with an independent vendor.

These are only a few of the problems facing the computer industry today. The field of computer law is still in its infancy. This article should be considered merely expository. It is by no means intended to act as an all-inclusive guide or definitive treatise on the areas covered. Readers are urged to seek the advice and counsel of an attorney regarding specific legal problems.

## Software Protection

Applications software comes within the realm of intellectual property. Statutory protection for computer programs is available through either copyright or patent. A third alternative is non-disclosure and retention as a trade secret under common law protection of the state. The historical authority for both copyrights and patents is found in the United States Constitution.[1] The protection offered software as a trade secret differs from state to state. (In one state, unauthorized duplication of software by an employee resulted in

a conviction for larceny.)
The new copyright law provides a fairly easy and inexpensive way to protect software from infringement.[2] The proceedure consists of publishing the material with the required copyright notice, filing the proper forms, and remitting the nominal fee.
Initially there was a great deal of confusion over whether the combination of symbols, words, and instructions contained in a program consituted proper subject matter for copyright. Hearings conducted by the government finally settled the issue with a policy statement expressing a legislative intent that computer software and data bases be registered by the Copyright Office in the same manner as literary works. The fact that the program may only be in a machine-readable form is of no consequence as long as the proceedures for obtaining the copyright are otherwise followed. As a consequence of the Copyright Office's policy of favoring registration of software, the number of programs being copyrighted has increased dramatically in recent years.
The most attractive feature of the revised copyright act is the extended term of protection for new works. Under the prior law a copyright was valid for a maximum of 28 years. The new law extends copyright protection for the life of the author plus fifty years. (The residual term is subject to disposition by inheritance.) The obvious disadvantage of securing a copyright is the public exposure. The idea or basis of a program is not protected by the copyright, but only the author's expression of that idea. Nor will a copyright protect the internal logic, methods, or processes employed in the program. Thus, while making an exact copy of a program constitutes a copyright infringement, a programmer is free to develop a similar program based on the same idea. (The above analysis is applicable to unauthorized duplication. The purchaser of a book or a cassette containing software is granted a non-exclusive license by virtue of the sale to utilize the programs contained therein without constituting an infringement. However, a purchaser's duplication and sale of the book or cassette to others would be a copyright infringement.)
On a practical level, the adding of superficial changes to a program can make the already difficult task of detecting a copyright infringement virtually impossible.
Patents offer yet another possible statutory method of protecting applications software.[3] When compared to copyright registration, a patent application is an arduous, expensive, and uncertain proceedure. The application requires that the item to be patented be described in technically worded 'claims'. Additionally, a costly patent search is required.
While the Patent and Trademark Office readily grants patent protection for hardware devices and inventions, applications software has generally been excluded. One reason for this is the Patent and Trademark Office's conclusion that such software represents 'mental output' rather than 'physical devices' or 'processes'. Despite this, there have been a number of programs that have been granted patent protection after lengthy court battles involving the Patent and Trademark Office and the applicants. Most of the cases deciding in favor of granting patent protection revolve around software which

the court characterized as part of an industrial or manufacturing process. The case law in this area often appears contradictory and is an uncertain guide as to whether a particular program can be patented. Due to the overlap of subject matter jurisdiction, it appears that some software would qualify for both patent and copyright protection. This, too, is an unsettled area in the law.

The term of protection offered by a patent is only 17 years. The public disclosure problem involved in copyright applies with equal force to patents. Once a program is exposed, the circumvention of the patent claim becomes likely through reverse engineering.

Once a software developer discovers an unauthorized use or duplication of a copyrighted or patented program, an action for infringement can be brought to recover lost profits. In some instances, punitive damages have been awarded when the infringement has been blatent and flagrant. Injunctive relief is also available to prevent further infringement.

Proof of the infringement may be difficult if the software is in machine-readable form. This may present an evidentiary problem before trial. Loss of profits must also be proved as a measure of damages. Clearly, copyrights and patents do not prevent unauthorized duplication, but they do provide a deterrent and offer a remedy at law.

The area of law providing the greatest protection for certain applications software is that of trade secrets. Obviously, this doctrine is not applicable to mass distribution of software recorded on cassettes or published in books. However, a software developer who leases a large, customized system package will find it much easier to prevent imitation and piracy if only a few users have access to the source code.

The proceedure for establishing a program as a trade secret primarily involves a well-drafted contract outlining security measures. The source code should never be published or publicly disseminated. The actual program tapes or disks should be marked with a notice to the effect that the contents are not for public view and should probably be numbered. The lease agreement should contain clauses that require non-disclosure of any documentation, data, source code, etc. To add an incentive, a liquidated damages clause should be included. (A liquidated damages clause specifies a dollar amount that the leasee agrees to pay should any unauthorized duplication occur.)

Unlike copyrights and patents, trade secrets have no specific term of protection. A trade secret is protected as long as it is kept secret. This is not to imply that a legal action to recover lost profits can not be instituted once an unauthorized duplication is discovered.

To aid in the determination of which buyer 'leaked' the source code, a different and unique identification number, program flaw, or 'bug' can be added to each cassette or disk to act as a tracer. (As computer technology becomes even more advanced, it may be possible to prevent discovery of a source code or duplication of a tape or disk by some sort of internal scrambler.) Presently, use of machine language makes it far more difficult for infringers to decipher source codes.

While even an accidental duplication of copyrighted or patented material will result in an infringement, no similar

protection is available for trade secrets. While the law of unfair competition will prohibit the luring away of key employees with access to trade secrets, it is quite permissible to attempt by honest and independent means to discover or duplicate the trade secrets of another.

Limiting Potential Liability

An independent developer of applications software should realize that the use of a program might result in legal liability. Either the buyer, or an unrelated third party, might receive injury or suffer some loss as a result of computer use. Example #1: A billing system that disintegrates due to a program flaw and erases a company's accounts receivable. Example #2: A bridge designed by a software program which collapses and injuries some motorists.

For this reason, a vendor's contract of sale or lease should include specific disclaimers for any and all damages stemming from the use, or misuse, of the software. It would be prudent to include a 'hold harmless' clause whereby the purchaser agrees not to hold the vendor liable for any damages arising from the use of the software.

While these avenues offer some protection as between the vendor and purchaser of the software as portrayed in example #1 above, it would not insulate the vendor in the second example since there is no contractural relationship between the vendor and the third party claimant motorists. Therefore, a vendor would be advised to include an indemnity clause which would require the purchaser to reimburse the vendor in the event of a lawsuit based on the use of the software. This might include a provision for attorney's fees as well.

In addition to contractural limitations on liability, a vendor of real life applications software should obtain adequate liability insurance in the event that the court finds these contract terms void or invalid. Yet another layer of insulation from liability is possible with incorporation.

While none of these approaches offers an guarantees that one will escape liability, they do minimize the potential risk involved.

Purchase Considerations

Any purchaser of applications software desires that the program operate up to certain standards. All too often these standards are never reduced to writing. For this reason, a purchaser should make sure that the program fills the needs of the business by outlining what is expected of the program and conducting a trial or benchmark to assure compliance with his requirements. This should be done before the full payment is remitted to the vendor. A reputable vendor should be more than willing to offer a written guarantee that the software is operational and will perform as measured against the benchmark. (The legal significance of the benchmark is that it may determine whether a breach of contract has occured.)

Concurrent with this guarantee, a vendor should also warrantee that the software violates no patents or copyrights and that the software does belong to the vendor.

When dealing with an independent software developer, it is essential that full and complete documentation be obtained. The reason for this is that if some-

thing should happen to the vendor it would be extremely expensive and disruptive to have another programmer attempt to maintain the program working from scratch. Unless the purchaser has the resources to do so, ongoing program maintenance and upgrades should be a featured part of any contract or lease for applications software.

Summary

The law offers three distinct ways to protect applications software: copyright, patent, and trade secret. Trade secrets offer the greatest protection but are not suitable to mass-distributed software. Copyright registration is a good alternative due to the extended term of protection. Patents are the most difficult and costly proceedure and have the shortest term of protection. Both patents and copyrights have the disadvantage of public disclosure.
Contract clauses can help to minimize the potential liability of a software developer.
Software purchasers should seek guarantees that the software is functional and that the vendor holds good title. The specific requirements should be reduced to writing and a benchmark run. Buyers should demand complete documentation to protect their investment.

Footnotes

1. United States Consitution, Article I, Section 8, Clause 8
2. Title 17 United States Code, Section 100, et seq.
3. Title 35 United States Code, Section 100, et seq.

References

Melville B. Nimmer, Nimmer on Copyright, Mathew Bender & Co., 1979

Robert P. Bigelow & Susan H. Nycum, Your Computer and the Law, Prentice-Hall, Inc., 1975

Dick H. Brandon & Sidney Segelstein, Data Processing Contracts, Van Nostrand Reinhold Co., 1976

Walter E. Hurst, Copyright, 7Arts Press, Inc., 1977

Dr. B.J. Korites, Software Publishing, Kern Publications, 1979

---

# MODULAR AND STRUCTURED PROGRAMMING ON SMALL SYSTEMS (INCLUDING 6809 ASSEMBLY LANGUAGE)

Terry F. Ritter
Motorola, M2880
3501 Ed Bluestein Blvd.
Austin, TX 78721

## Introduction.

The goal of programming is to write a sequence of instructions to implement an algorithm to produce the desired result. The goal of modern programming is to be able to write more than four lines of code at a time and still have that code work properly. Many programmers have begun to follow a body of techniques that can be of significant help in the creation of reliable programs. This paper will define these techniques, show why they are helpful, extend them for use in a small computer environment, and show how they can be applied at the assembly-language level on the 6809.

This paper presents a systematic new way of programming at the assembly-language level. Using the described techniques, modular, structured programs can be written without using a high-level-language. This is possible because the required structured-programming facilities have been identified, and each is available directly on the target machine (the MC6809). The assembly-language use of these facilities is not automatic, as it would be in a structured high-level language, but the facilities are available for use, and such use becomes convenient and obvious. The programming innovation consists of restricting the code to the use of only the structured facilities. Although the 6809 is upward compatible from the 6800, 6800-like programs can only perpetuate those non-modular features which demanded the creation of the 6809 in the first place. Structured techniques were so complex on all early microprocessors that they reduced --rather than improved--the probability of writing a correct program, and drastically reduced performance, as well.

## Reliable Programming

It strains credibility to say that writing programs should be so difficult as to need a disciplined approach to the craft. Yet this is indeed so, and some thought on the topic will show that a myriad of details must be remembered during the actual process of programming. If writing new or different instructions can change the operation of other code segments as a side-effect, then the number of details which must be remembered increases combinatorially; the problem quickly gets out of hand.

Modern programming practices seek to generate systems that are easily decomposed into seperate, readable, understandable, testable elements with no side-effects on each other. This limits the programming design problem to the single module in design, plus well-defined interfaces to other "lower-level" modules.

Structured Programming. The first attempts at limiting combinatorial complexity centered around using a few well-defined control structures as opposed to just "any old thing." (Conditional repetition is the very heart of programming; most looping errors occur in the first pass or near the last expected pass through the loop.

Having just a few looping structures means that these can be well-learned, and their associated error-possibilities better anticipated). In addition, "Spaghetti Code" was singled out to be totally eliminated. (Spaghetti Code is usually produced when the programmer has inadequite editing tools, and rather than inserting new code in-line, he jumps to new code in a non-sequential location, which then jumps back into the old code. The resulting code has large numbers of GOTOs (Jumps) which make the code exceedingly hard to follow, and thus hard to analyze for correctness). Spaghetti Code is obviously combinatorially complex.

The battle plan (aimed at bringing structured programming into wide use) was to design languages which did not even <u>have</u> facilities which could create combinitorially complex programs. The principal problem was the GOTO, since it could be used to create any awkward control structure, as well as spaghetti code; so it was suggested that the GOTO be eliminated in favor of a few well-defined control structures. Unfortunately, the defined control structures were not quite sufficient; occasionally a GOTO was still required (and thus still exists in Pascal). However, the situation had changed from languages which required GOTOs for <u>all</u> control, to languages allowing a GOTO in specific instances. This GOTO controversy clouded the whole issue: Structured Programming involves <u>using</u> "correct" control structures; the fact that others are available is just beside the point.

Additional goals of a structured programming language were to allow programs to be understood based upon their straightforeward appearance. (For example, the "scope" of variables in Pascal is based upon the appearance of the program before it is executed, instead of the dynamic character of invocations during execution). Unfortunately, this goal is not fully met even in Pascal (due to the IF/THEN syntax).

<u>Modular Programming</u>. Other attempts to limit combinitorial complexity resulted in writing the code to do a well-defined task <u>as a subroutine</u>.

In addition, the subroutine must not have <u>any</u> un-obvious side-effects to other computation; the operation of the module must be able to be fully-understood by looking at this module (plus its input data) alone. Perhaps the principal implication of this requirement (to language design) was to require the availablility of "local" as well as "global" variables. (A local variable is used for temporary data storage only while in a module; the facility for local storage must be arranged so that any write to a local does not affect the later processing of other modules). (A global variable is used for data storage which is available to all modules. Unfortunately, this means that a module-in-error may damage the value of a global, thus causing later program failure by modules <u>not</u> in error, and far-removed from the error-causing trouble-maker). Local storage implies the ability to use exactly the same variable-names (or line numbers!) in different modules; there can be no modular-programming environment at all without local variables. And some global storage facilities (the <u>only</u> storage in most microcomputer languages) are still required even in a modular-programming environment.

These "Structured" and "Modular" techniques do not eliminate programming as a complex design activity: the design of big systems is a big job. But that job is more manageable when using good techniques.

<u>Re-entrancy</u>, <u>and Recursion</u>. Many real-world programs will eventually involve execution in an interrupt-driven environment. If the interrupt-handlers are at all complex, they might well be written to use routines in the I/O service environment. If so, they might well call the same routine which has just been interrupted. To protect the present programs against certain obsolescence, all programs should be written to be re-entrant.

(A re-entrant routine allocates different local variable storage upon each entry. Thus a later entry does not destroy the processing associated with an earlier entry).

The same scheme which was implemented to allow re-entrancy will probably also allow recursion. (A recursive routine calls itself). One might write a recursive routine to simplify the solution of certain types of problems, especially those which have a data structure whose elements may themselves be a structure. For example, a parenthesized equation represents a case where the expression in parenthesis may be considered to be a value which is operated on by the rest of the equation. A programmer might well choose to write an expression-evaluator which, when it encountered parenthesis, would call itself, passing the parenthesized-expression (which might also contain parenthesized expressions) in the call, and receive back the returned value of the expression within the parenthesis.

## The Small-Computer Environment

The application-opportunities for microprocessors consist of those things for which large computers are too large, or too expensive. (For the most part, microprocessor users perceive that the worth of the processor exceeds its cost--and that they can recover this "additional" value, simply by properly applying the unit). Many people have home computers today (who did not have them ten years ago) as a result of a technology which allows computers to be mass-produced at very low cost. And the very lowest cost way to store a single program for mass-production execution is in Read-Only-Memory (ROM).

There are advantages and disadvantages to ROM's, of course. But because of that little detail about their low cost, many programs written in microprocessor assembly language do end up in ROM. Unfortunately, most of these ROM's are only applicable in a single system; this tends both to limit applications for the ROM, as well as forcing the system, too early, into a rigid hardware configuration.

Additional factors in the design of programs for microprocessors include position-independence, local storage allocation from the stack, and indexed-indirect operations for I/O.

## 6809 Modular Facilities

Making a Module. It is necessary, but not sufficient, that a module be a subroutine. This means that, upon entry, the absolute address which is the return to the calling routine is on the top of the stack (0,S). (The term "0,S" is used in the address-field of an indexed-mode instruction used by the 6809 assembler. "0,S" means: "temporarily add the value 0 with the value in the S-register and use the result as the effective memory address"). Now, to isolate those registers which will be modified inside the module (essentially requiring a form of local storage), we will push those registers on the stack (with one instruction, e.g., PSHS Y,X,B,A). After the body of the module is executed, we will collect the saved registers, and perform a subroutine return in "one fell swoop," (e.g. PULS A,B,X,Y,PC).

Parameters. Parameters may be passed to (and from) assembly-language modules either in registers or on the stack. If the registers provide sufficient storage for parameter passage, fine; but there is no point in saving a register upon module entry if you intend to use it to return a value (to the caller). If parameters are to be passed on the stack, they are placed there before calling the lower-level module. The called module is then written to use local storage inside the stack as needed (e.g., ADDA offset,S ). Notice that the required offset consists of the number of bytes pushed (upon entry), plus two for the stacked return address, plus the data offset at the time of the call; this value is easily calculated by drawing a "stack-picture" diagram representing module entry,

and assigning convenient mnemonics to these offsets. Returned parameters replace those sent to the routine. If more parameters are to be returned than would normally be sent, space for their return is allocated by the calling routine before the actual call (if four additonal bytes are to be returned, the caller would execute LEAS -4,S to acquire the additional storage).

Local Storage. Local storage space is acquired from the stack for use while the present routine is executing (or waiting on a lower-level execution); the storage is then returned prior to exit. The act of Pushing registers which will be used in later calculation essentially saves those registers in temporary local storage. Additional local storage can easily be acquired from the stack (for example, executing LEAS -2048,S acquires a buffer area running from 0,S to 2047,S inclusive). Any byte in this area may be accessed directly by any instruction which has an indexed addressing mode (this includes 57 instructions). At the end of the routine, the area acquired for local storage is released (e.g., LEAS 2048,S) prior to the final Pull.

Global Storage. The area required for global storage is also most-effectively acquired from the stack, probably by the highest-level routine in the standard package. Although this is local storage to the highest-level routine, it becomes "global" by positioning a register to point at this storage, then establishing the convention that all modules will pass that same pointer value when calling lower-level modules. In practice, it is convenient to leave the stack-mark register unchanged in all modules, especially since global accesses will be common. The highest-level routine in the standard package would execute the following sequence upon entry (to initialize the global area):

```
PSHS  U       higher-level mark
TFR   S,U     new stack mark
LEAS  -17,U   global storage
```

Note that the U-register now defines 17-bytes of locally-allocated (permanent) globals (which are -1,U through -17,U) as well as other external globals (2,U and above) which have been passed on the stack by the routine which called the standard package. Any global may be accessed by any module using exactly the same offset value at any level (e.g.; ROL RAT,U ; where RAT EQU -11 has been defined). Furthermore, the values stacked prior to invoking the standard package may include pointers to data or I/O peripherals. Any indexed operation may be performed indexed-indirect through those pointers, which means that the standard package need know nothing about the actual hardware configuration, except that (upon entry) the pointer to an I/O register has been placed at a given location on the stack.

System-Independence.

It is best to develop software which does not restrict the further development of the target hardware. Using facilities availiable at the machine level, programs can be written which can be placed in ROM, and that ROM can be positioned anywhere in the memory address space, and still execute correctly. (Facilities allowing position-independence include relative branches for simple or conditional control-transfers within modules, relative long-branch-to-subroutine to call other modules, and program-counter-relative (PCR) indexed addressing for access to tables within the program).

Program-Counter-Relative addressing on the 6809 is a form of indexed addressing that uses the program-counter as the base register for a constant-offset indexing operation. However, the 6809 assembler treats the PCR address field differently from that used in other indexed instructions. In PCR addressing, the assembly-time program-counter value is subtracted from the (constant) value of the PCR offset; the resulting distance is the value placed into the machine-language object code. This machine-instruction contains the distance to the desired symbol; during execution, the processor

adds the value of the run-time PC to that distance to get a position-independent absolute address.

The PCR-indexed form can be used to point at any location relative to the program regardless of position in memory. The PCR form of indexed addressing allows access to tables within the program-space in a position-independent manner. The wide range of low-overhead facilities for obtaining Position-Independent-Code (PIC) means that there is little excuse for writing position-dependent programs on the 6809.

Given a program which is completely position-independent, some absolute locations are usually required, particularly for I/O. If the locations of I/O devices are placed on the stack (as globals) by a small set-up routine before the standard package is invoked, all internal modules can do their I/O through that pointer (e.g., STA [ ACIAD,U ] ), thus allowing the hardware to be easily changed, if desired. Only a single small and obvious set-up routine need be re-written for each different hardware configuration.

Loops. Clearly, the usual structured loops (i.e., REPEAT...UNTIL, WHILE...DO, FOR..., etc.) are available in assembly language in exacatly the same way a high-level-language compiler would translate the construct for execution on the target machine. Using a FOR...NEXT loop (a very error-free construct) as an example, it is possible to Push the loop count, increment value, and termination value on the stack as variables local to that loop. On each pass through the loop, the working register is saved, the loop count picked up, the increment added in, and the result compared to the termination value. Based on this comparison the loop counter might be updated, the working register recovered and the loop resumed, or the working register recovered and the loop-variables de-allocated. While apparently more complex than one would like, reasonable macros could make the source form for loops trivial, even in assembly-language; such macros might reduce errors resulting from the use of multiple instructions simply to implement a standard control-structure.

## Programming Development

Although a programming language can offer the programmer features which may be used in a structured, modular way; the language is only part of the development story. The value of the ability to program, test, execute, and trace modules as they are developed in an interactive environment cannot be over-estimated. Perhaps the only language to implement a modular development environment has been APL (although BASIC09, when released, also enjoys this same advantage).

It is important to be able to save and recover a whole group of modules which is the environment (workspace) under development. It is important to be able to save and recover individual modules from previously-saved workspaces, and to change their names. It should be made easy to enter input-parameters to the module from the keyboard, execute the module, then observe the values returned by it. Since parameters to and from the routine will frequently be on the stack, it should be possible to easily place values on the stack to be sent to the called routine, and returned values easily examined. The programmer should be able to easily modify the code inside a module, and selectively trace execution of one module while others (when they are called) execute at full speed. And the trace should display both the instruction executed (in the source language), as well as any variables changed by that instruction (and not the myriad variables which do not change). Since global values are an inherent requirement for complex programs, it should be possible to selectively display any write-access to particular global variables, or, upon such access, to stop and await additional commands from the programmer.

The above properties, while completely outside a language definition, strike at the very heart of software development. Nor are they necessarily limited to high-level language implementations; it is entirely feasible to design an interactive, modular, assembly-language programming system. Such a system would have an incredible effect on the ease of programming in assembly-language.

Conclusion.

The aims and practice of modern structured programming have been elucidated. Specific methods have been detailed for writing structured, modular, and hardware-independent systems in 6809 assembly language. The result of using these techniques on a day-to-day basis is so beneficial as to produce a true revolution in assembly-language programming.

## 6809 BACKGROUND

by Ritter, et. al.


"Standard-Product ROM's (SPR's)," Proceedings of the IEEE Workshop on Microprocessor Firmware, (presented March, 1979; to be published).

"Programming the 6809," Volume 1, Number 9 (Nov/Dec 1979), pp. 34-37.

MC6809 Preliminary Programming Manual, First Edition (1979).

"Upward Compatibility: More Power--Less Pain," The Best of the Computer Faires, Volume 4: Conference Proceedings of the 4th West Coast Computer Faire (May, 1979), pp. 351-358.

"M6809 is Silicon," BYTE, Volume 4, Number 5 (May, 1979), pp. 30-31.

"A Microprocessor for the Revolution: The 6809, Part 3: Final Thoughts," BYTE, Volume 4, Number 3 (March, 1979), pp. 46-52.

"A Microprocessor for the Revolution: the 6809, Part 2: Instruction Set Dead Ends, Old Trails, and Apologies," BYTE, Volume 4, Number 2 (February, 1979), pp. 32-42.

"A Microprocessor for the Revolution: the 6809, Part 1: Design Philosophy," BYTE, Volume 4, Number 1 (January, 1979), pp. 14-42 (Cover Article).

"Resident Memory Test Systems: With an Example for the 6800 (sic)," Dr. Dobb's Journal of Computer Calisthenics and Orthodontia, Number 25, Volume 3, Issue 5 (May, 1978), pp. 28-33.

"Compatibility Cures Growing Pains of Microcomputer Family," Electronics, Volume 51, Number 3 (February 2, 1978), pp. 95-103 (Cover Article).

The MC6809 Microprocessor (Motorola NMOS Microcomputer Design Specification) 6/28/77, 9/30/77, 2/28/78, 8/04/78; 224 p.


other authors:


OSBORNE/McGraw-Hill. An Introduction to Microcomputers, Volume 2: Some Real Microprocessors (Update 5, July 1979), pp. 9-175 through 9-211; pp. 9-D1 through 9-D39.

Gooze, Mitch. "How a 16-bit Microprocessor Makes it in an 8-bit World," Electronics, Volume 52, Number 20 (September 27, 1979), pp. 122-125.

Peatman, John B. Microcomputer-Based Design (McGraw-Hill, 1977, 1978). Appendix A7: Motorola 6809, pp. 533-551.

MC6809 Advanced Information datasheet (Motorola Semiconductor Products, Inc., 1978).

# STRUCTURED FLOWCHARTS — A HYBRID APPROACH TO PROGRAM DESIGN

Gregg Williams
BYTE Magazine
70 Main St.
Peterborough, N.H. 03458

## Abstract

Flowcharts lead to poor program design. Structured pseudocode (or, equivalently, a Pascal-like language) does not have the visual appeal of a graphic design technique. The notation described here, which results in a what will be called a structured flowchart, attempts to combine the best features of flowcharts and structured pseudocode. When read from top to bottom, the structured flowchart describes the task to be performed; when read from left to right, it describes successive decompositions of a task into its corresponding subtasks. In this manner, the technique allows top-down structured design.

## Introduction

No rationale need be given for structured programming techniques, nor for the concept of top-down structured design (repeated division of a task into its subtasks until each subtask is easily codable in a given programming language). The notation described here grew out of my dissatisfaction with existing design methods (this includes flowcharts, structured pseudocode, Nassi-Schneiderman charts, and Warnier-Orr diagrams).

The closest existing solution for me was structured pseudocode — that is, an informally written Pascal-like "program" that uses terse English phrases describing the actions to be performed. But even that had one problem: once the program was written, the overall program flow was obscured by the details.

The notation described here, which we will call structured flowcharts, evolved from an idea presented by O. Ferstl in the Programming Languages subgroup magazine (SIGPLAN) of the ACM [1]. It has evolved over many months of daily use, and it is responsible for the existence of a 35-page COBOL program I wrote that was later expanded to 75 pages without any loss of program clarity or maintainability.

Of course, this method (or any other secondary method) is of little use when the target language is Pascal or any Pascal-like language: there, the design effort produces the computer program directly. But there are still many people who must write in an unstructured language — the average hobbyist, in BASIC; the business programmer, in COBOL; other programmers, in PL/I, FORTRAN, and other languages. This paper assumes that you will be able to translate the constructs described here into the target language. For a treatment of the translation process to BASIC, refer to article [2] mentioned in the references.

## The Basic Constructs

According to the tenets of structured programming, any program can be expressed as a combination of four basic building blocks. These are sequence, while...do, if...then...else, and decompostion. The first three constructs are described in conventional and structured flowchart form in figures 1 through 3.

The sequence construct shows us one of the main differences between the conventional and the structured flowchart. While the program flow runs through the boxes in a conventional flowchart (as in figure 1a), in a structured flowchart, control runs down a central spine, with the boxes executed along the way being drawn to the right of and connected to the spine.

The while...do is a tradeoff. While the conventional flowchart cannot directly express this kind of loop but must use an if diamond and an externally defined loop (figure 2a), the structured flowchart must introduce a new symbol, a hexagon. (Actually, the hexagon denotes one of several kinds of loop structures; the word while makes this a while...do loop.) The box connected to the hexagon below and to the right is performed as long as the condition listed in the hexagon is true. The condition is performed first (denoted by the position of the hexagon being spatially above the box being performed); this allows the possibility of the body of the loop being performed zero times if the condition is initially false.

The if...then...else construct is fairly straightforward in the conventional flowchart (figure 3a). In the structured flowchart, the boxes to be performed are to the right of the central spine, but the decision diamond between them allows only one of the two boxes to be performed, based on the value of the condition inside the diamond. If the else side of the construct is not needed, one box can be eliminated with the understanding that, if the conditions for the remaining box are not met, no action is performed and control continues with the next construct on the spine.

The fourth and pivotal construct of this programming notation can be best stated as a rule: any box representing a task can be broken into multiple boxes representing the necessary subtasks. The subtasks may be rectangular boxes representing simple tasks; or they may be any other valid structured flowchart construct (if...then...else, while...do, etc.). They are written top to bottom in the order that they are to be performed, with the top subtask horizontally level with the parent task. A vertical spine is drawn between the parent task and its subtasks and connects the two by a horizontal line from parent task to spine and by horizontal lines from the spine to each subtask.

The example of figure 4 illustrates the above construct. Task X is composed of five subtasks performed in numeric sequence. Tasks 1, 2, and 5 are simple subtasks. Subtask 3 is an if...then...else construct that allows either subtask 3a or subtask 3b to be performed. Subtask 4 is performed as long as the condition within the hexagon (B>Y) is true. Of course, any subtask box may itself be divided into its component subtasks.

Since any box can be broken into its component subtasks, we can now see how this notation can be used to design a program. The boxes in the level immediately to the right of the leftmost spine give the overall design of the program; then, as both algorithmic and implementation details are decided, any given box can be expanded to the right. As a result, the user can scan one or several of the leftmost columns of boxes for an overview of varying depths of the program design; or he or she can study the implementation of any major or minor subtask by concentrating on only the boxes and control structures growing to the right of the given subtask.

## An Example

The following short example will illustrate the process of developing a program using structured flowcharts. Suppose we are given an array of N numbers, V(1), V(2),..., V(N) and have to find the index value MAXINDEX such that the largest value in the V array is MAXV = V (MAXINDEX). The entire structured flowchart for this problem is given in figure 5.

Cover the right two-thirds of the flowchart so that only the subtasks numbered 1, 2, and 3 are visible. This is what the "first pass" of the flowcharting effort should look like. Subtask 1 is the initialization of the problem. Subtask 2 is the determination of MAXINDEX and MAXV. Subtask 3 is the printing of these two values. Since the task in subtask 3 is simple enough to be accomplished directly in the target language (for example, BASIC), it does not need to be subdivided.

Subtasks 1 and 2 are developed concurrently. Subtask 2 is basically a loop that examines V(1), V(2),..., V(N) in turn, keeping the appropriate values for MAXV and MAXINDEX for the I elements encountered so far. For this, the values of MAXV, MAXINDEX, and INDEX must be set (as is done in subtasks 1.1, 1.2, and 1.3).

The main work done for each element is done as subtask 2.1.2: if the current V element being looked at (CURRV) is greater than the maximum V element so far, MAXV and MAXINDEX are set to the current array and index values, respectively. Note that these are shown as subtasks 2.1.2.1 and 2.1.2.2, and that they are performed only when the relationship given in the diamond of 2.1.2 is true.

Once the structured flowchart has reached the level of detail shown in figure 5, most of the design considerations have been thought of and worked out; it is then a simple task to translate the program into BASIC or any other general-purpose computer language. The benefits are more pronounced when used with a larger program. If a structured flowchart is subdivided to the right until each box represents a task that can be directly coded in the target language, you will have already caught most of the "oops, I forgot to..." insertions and changes that most programmers think of after they have started coding the program.

## Other Control Structures

Other control structures can be devised for the convenience of the programmer. For example, boxes 1.3, 2.1, and 2.1.3 in figure 5 implement a control structure available in most programming languages — a do-loop that varies INDEX from 1 to N. An example of the notation I have devised for this is given in figure 6a; the body of the loop is performed according to the parameters given in the hexagon.

Another well-known control structure is the repeat...until loop, shown in figure 6b. Note that the body of the loop stems from a diagonal line that leads up to the first subtask; this indicates that the body is performed before the condition is tested, with the loop repeating only if the condition is false.

Other constructs come to mind: a case structure, an unconditional goto, and two controlled gotos — the restart (restart the immediately-containing loop) and the exit (go to the first task after the immediately-containing loop). Although I have used these constructs for quite some time, they are not presented here because I am not satisfied with the notations for them I have developed so far. In any case, you should add to and modify these constructs to fit your needs.

## Conclusions

I have found this structured flowchart helpful in designing programs. It is obviously intended for highly unstructured languages, with its utility decreasing as the structure of the target language increases.

The notation is, at the moment, informal and should stay that way. It should be extended and modified in whatever way seems useful to the user; if the structured flowchart is to be read by another person, however, all the structures used should be defined in terms of their equivalents in unstructured (conventional) flowcharts.

If the final structured flowchart is to be redrawn, do so with clarity in mind. Place only those boxes that help explain the overall design with the main flowchart; leave the implementation details to subordinate flowcharts.

I hope you will find this notation useful. I would appreciate hearing your suggestions, criticism, and comments.

## References


1. Ferstl, O., "Flowcharting by Stepwise Refinement," SIGPLAN Notices, volume 13, number 1, January, 1978. (ACM SIGPLAN, 1133 Avenue of the Americas, New York NY 10036; $3.00)

2. Williams, G., "Applied Structured Programming," Program Design, Blaise Liffick, ed.; BYTE Books, 1978. (BYTE Books, 70 Main St., Peterborough NH 03458; $6.00)


(figures follow on next 4 pages)

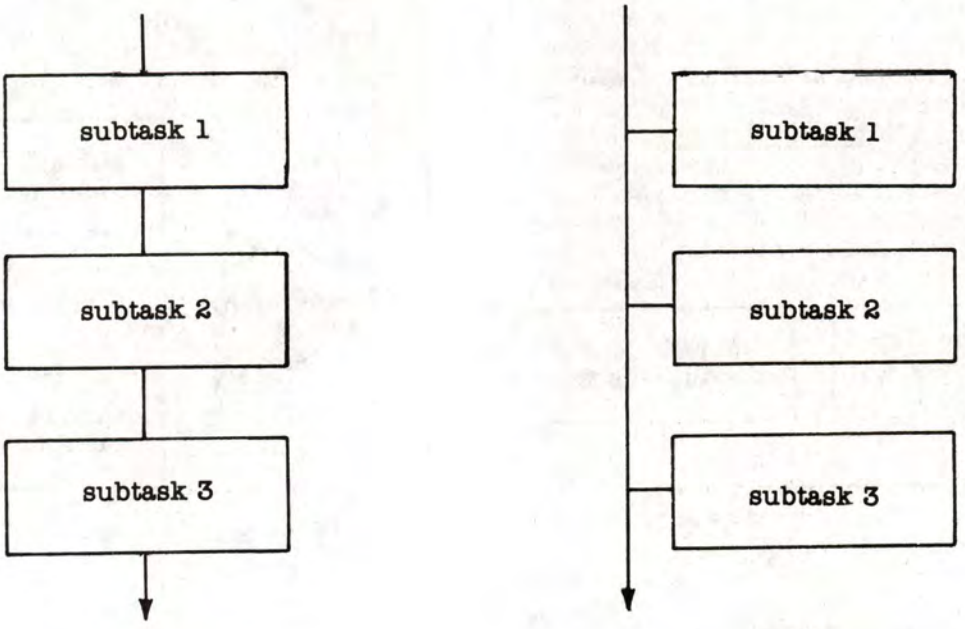



Figure 1: Sequence as a control structure. Figure 1a shows how a linear sequence of subtasks is drawn using conventional flowchart notation. Figure 1b shows the equivalent sequence using structured flowchart notation.

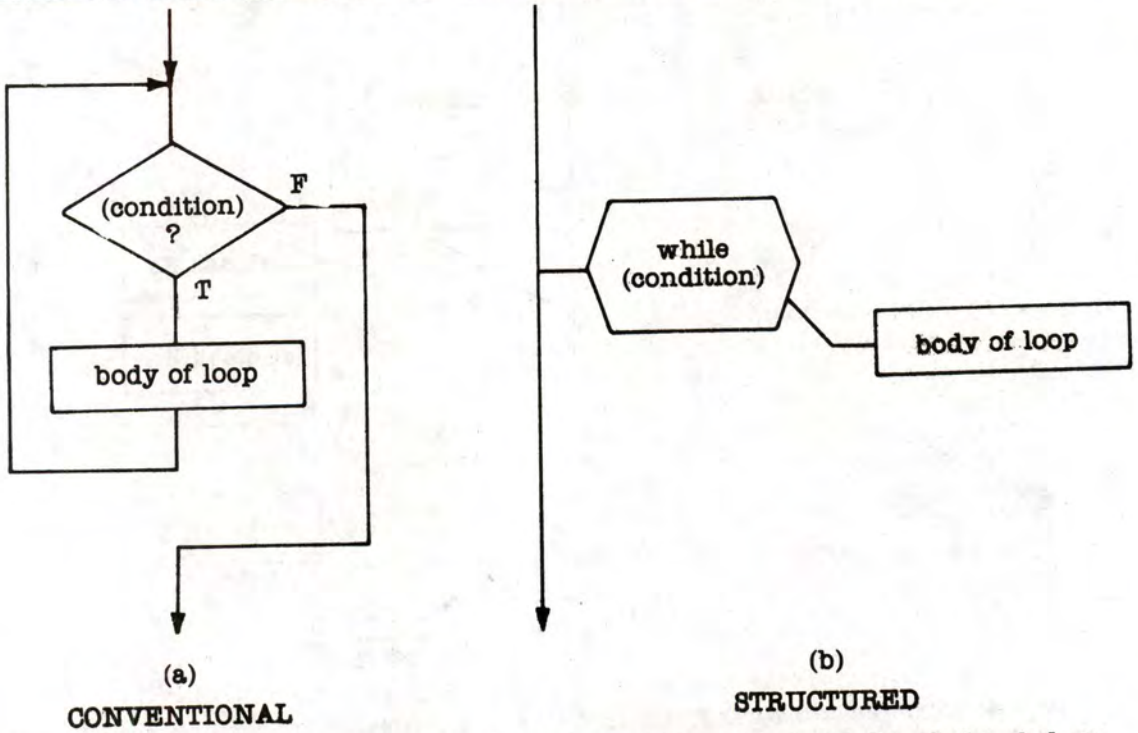
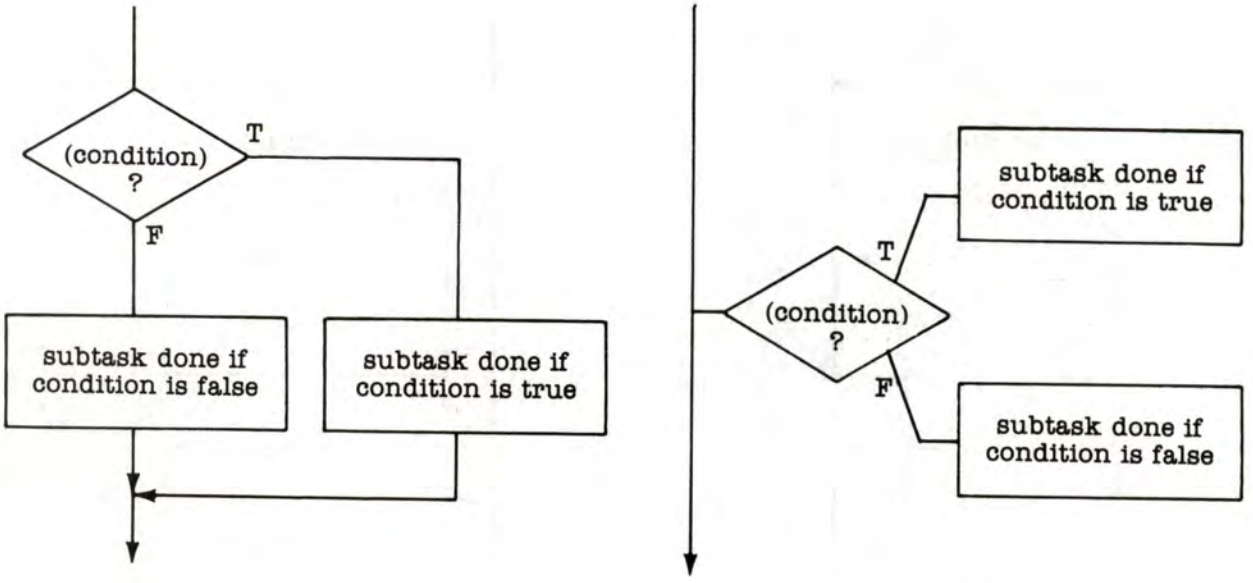



Figure 2: The while...do loop as a control structure. Figure 2a shows the while...do loop using conventional flowchart notation. Figure 2b shows the equivalent loop using structured flowchart notation. The diagonal line leading down indicates that the condition (in the hexagon) is performed before the body of the loop.




(a)
CONVENTIONAL

(b)
STRUCTURED

Figure 3: The if...then...else construct as a control structure. Figure 3a shows the conventional notation for this construct, while figure 3b shows the structured flowchart equivalent. Note that it is the placement of the letters T and F (for true and false) that determine the conditions under which a given subtask is performed.

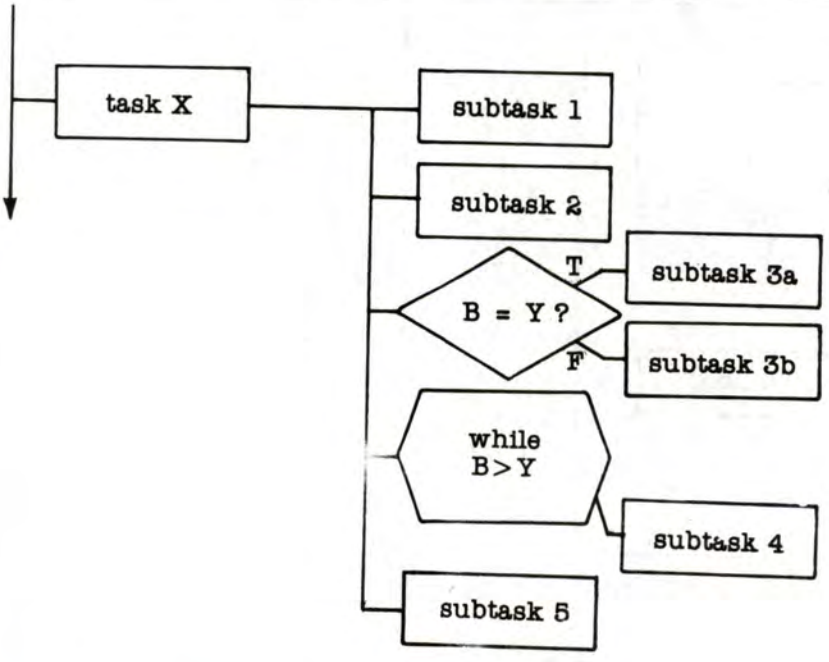



Figure 4: Example of the subdivision of a task. A central rule of structured flowcharts is that any box can be broken into multiple boxes that represent the necessary subtasks. Here, task X is broken into five subtasks executed in top-to-bottom order. Subtasks 1, 2, and 5 are simple subtasks. Subtask 3 is an if...then...else construct. Subtask 4 is a while...do loop.

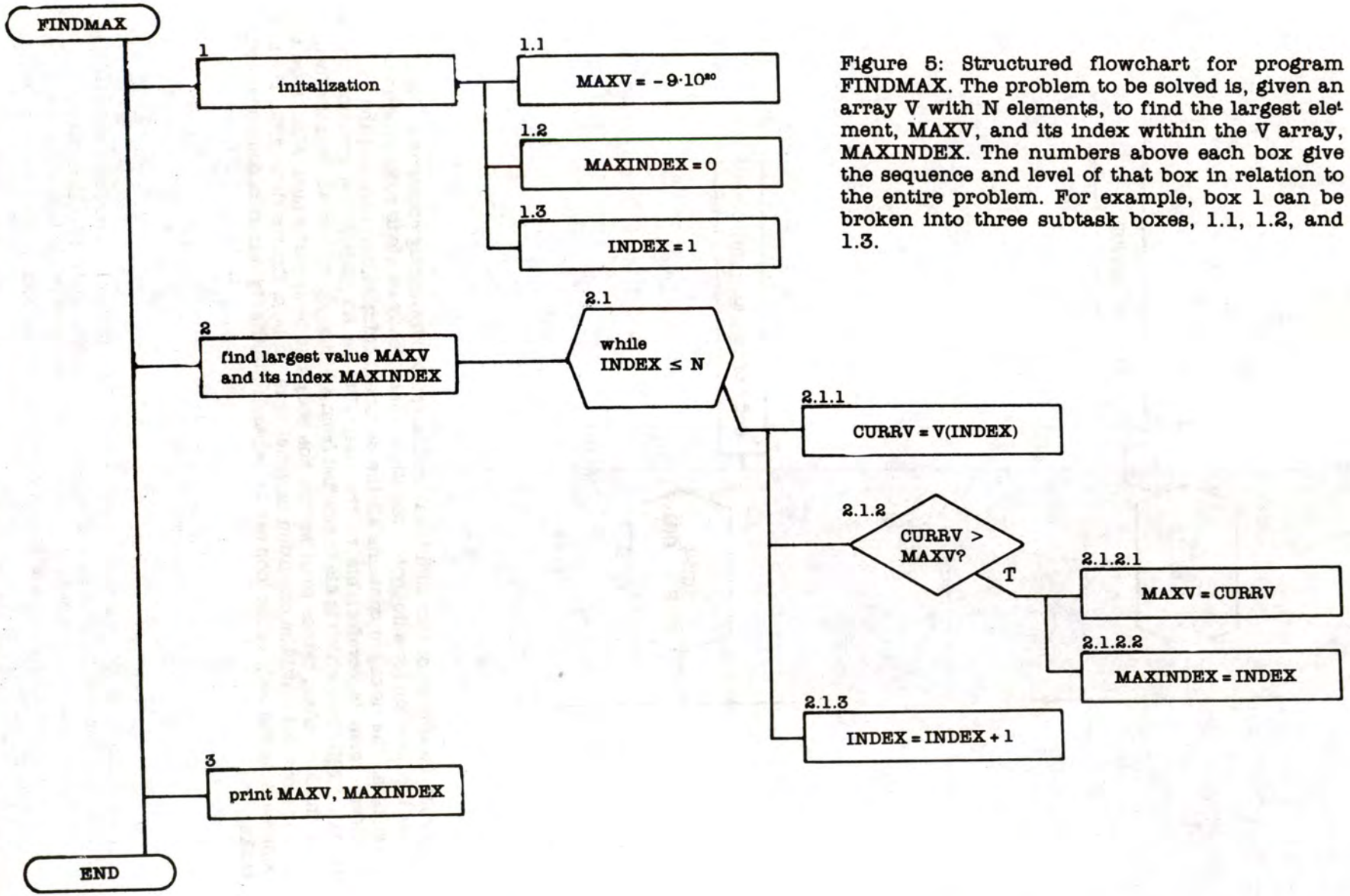
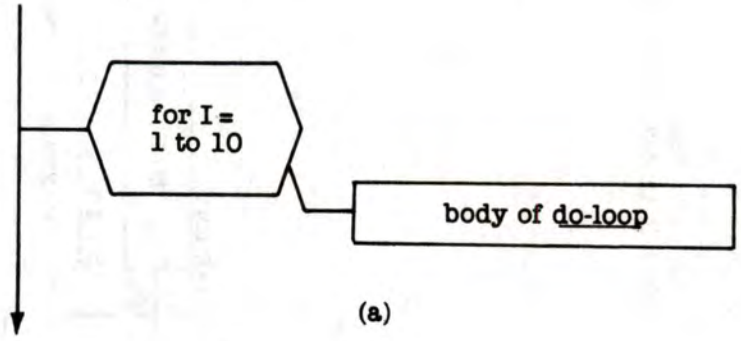
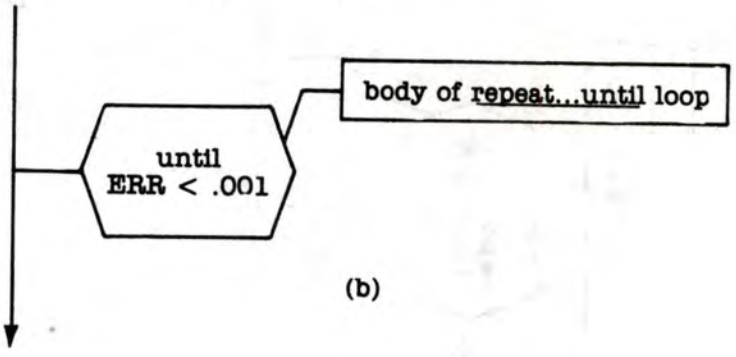



Figure 5: Structured flowchart for program FINDMAX. The problem to be solved is, given an array V with N elements, to find the largest element, MAXV, and its index within the V array, MAXINDEX. The numbers above each box give the sequence and level of that box in relation to the entire problem. For example, box 1 can be broken into three subtask boxes, 1.1, 1.2, and 1.3.




(a)




(b)

Figure 6: Examples of two additional structured programming constructs. Figure 6a shows the notation for a do-loop, a construct that already exists in most programming languages. The hexagon contains all the pertinent information defining the loop, in whatever form is comfortable to the user. Figure 6b shows the notation for a repeat...until loop, which is distinguished from a while...do by the fact that the former tests the loop after, rather than before, the body has been performed. Also, here the loop is executed until the condition becomes true. In both figures 6a and 6b, the box representing the body of the loop can be expanded to the right into its component subtasks.

# A CASE STUDY IN UNSTRUCTURED SOFTWARE

Howard R. Hollander
Specialist Software Engineer
Boeing Aerospace Co.
Little Mountain Test Annex
Hill AFB, UT 84056

There has been much talk within the software industry concerning structured software. The benefits are well known and many software groups are implementing structured software to various degrees. The following paragrapns detail a nonfictional account of an unstructured software project. The names of all parties concerned have been deleted. This was done to protect the innocent and to save the guilty from embarrassment.

Several years ago, while with a different company, I was given an assignment to work on the design of a real-time communications simulator for a major customer. The original plan was for my company to design, code, and debug the job. After debugging was completed, several people would relocate to the customer site for integrating the hardware and software. Documentation was important as changes would be required as time went by. Already there was talk of the first major revision, although it probably wouldn't be required until after the initial job was completed. The customer had asked for a structured software system, as previous experience with other contractors had shown them the difficulty in modifying unstructured code. We decided to use Hierarchy plus Input-Process-Output (HIPO) (1) for generating our design document. This would allow for easy update of documentation at the end of the integration phase. The many other benefits (e.g. functional breakdown for modular coding) convinced us that HIPO was the best method for implementation.

Our first milestone, the preliminary design review, was hailed a success. The system was big, but the design team spent many hours reviewing the system requirements so that no detail would slip through the crack. The structured design document allowed us to easily insert or delete design as required. The many published theories of structured software had made believers out of all of us.

As we proceeded with our detailed design in preparation for the critical design review, the customer took the first step toward unstructuring the system. It was decided that we were to stop the design, as the customer wished to make a major change in the requirements. We were to switch from a single task to a multitask system. At this point, most of the high level design was useless.

When the design group got down to business again, the customer delivered the second blow. The customer decided to use it's "organic capability" (later found to mean two employees, as well as four other workers, each from a different contractor company) to take over the design and then to implement the system. Our field support was reduced to one person hired at the customer site in addition to myself. Our company was to complete the section of design we were working on before I was to move out to the site.

A major module of our design was based on work done on a similar project several years earlier. The team members in the field had asked me for information on this portion of the design. As a gesture of kindness, I sent the source listings of the old version to the customer. It should be noted that both projects used the same computer manufacturer, although different models in the product line were used (the earlier project used a sixteen bit machine. The current project was using a 32 bit minicomputer). I assumed the source listing gift would be used to shed some light on the inspiration behind the design. The listing was of no value to the customer, as a number of software standards agreed to by all parties were not considered in the older system. The source code from the original system was not structured and used several structured no-no's such as address and instruction modification.

When I arrived at the customer site, I got one of the biggest shocks of my life. I had hand carried the still unfinished design document to the site, yet I found several people already coding. They had taken the source listing I had sent out, and were translating code line by line to accomodate the different computer. It was claimed that the listing was close enough to start with. Management was getting restless because there had been no visible progress (i.e. lines of code) on the project until that time.

A little background material concerning unstructured software is called for now. It is important to code without any restrictions, such as design, to hinder your progress. A line of code is the only metric available to project management. A design document is non-executable on a computer, hence they are of no value to

anyone. Lines of code should be grouped into modules of kilolines. Comments are strictly forbidden.

In order to truly unstructure the software the design document was thrown away (except for all the portions that were my responsibility. I still was a believer in structured software). With the design no longer available to slow down the progress, coding began in earnest. This method is known as real-time design in the parlance of unstructured software.

With something tangible (i.e. code) to show management, progress can be noted on the master schedule (i.e. a management tool that means nothing to any concerned party, but required by persons removed from the project). When a module is completed, one step forward is noted. At the next progress review, that step and another should be removed, as it is generally found that the completed module is not doing half of the functions it should be. Although the other half of the code performs everything it should, all the interfaces to the module are incorrect, and the software is rendered useless. The two steps backward to each step forward ratio represents progress in the middle of the project. Earlier on there are a greater number of steps backward for each one forward. The ratio decreases as time goes on, until eventually the code, debug and integration is completed, albeit at least one calendar year late (seven man years in this project. Luckily, it wasn't too large a project).

The program description document presented major problems. Several team members felt documentation was beneath their abilities. Several other team members found a way to use the original document. In order to conserve paper, they used the back side of the original document for the draft copy of the new document.

The customer approved an update to the system shortly after completion of the original effort. The team effort to implement the change was Herculean. Although I was no longer associated with the project, I was told that some team members grumbled under their breath about not structuring the code. To paraphrase a well known expression, just wait until next project.

Reference

(1) HIPO - A Design Aid and Documentation Technique, International Business Machines Corporation, GC20-1851-1, 1974.

# ANIMAL - An Animation Language used in Creating Animated Scenes in Color on a Personal Computer

by
Jim Blum
President of COMAGRAPH
(Computer Automated Graphics
and Text Processing)
768 Inwood Drive
Campbell, California 95008

ANIMAL (ANIMAtion Language) currently runs on a CROMEMCO SYSTEM III microcomputer system (see Figure 1), using a Summagraphics "BITPAD" digitizer, a portable SONI color TV, and the CROMEMCO "DAZZLER" bitmapped dot graphics board (64 X 64 resolution with 1 of 16 colors plus black, or 128 X 128 with 16 colors).

With the exception of the I/O drivers for the DAZZLER and BITPAD devices written in Z80 assembly language, the entire program is written in UCSD PASCAL, and runs under the UCSD PASCAL system. Therefore, ANIMAL can easily be adapted to other systems.

ANIMAL provides commands for creating animated scenes; running them in real time, and for saving and retrieving them from diskette. A scene consists of one or many individual frames which are "run" sequentially to create the animation. The entire scene is stored in memory using PASCAL dynamic variables structured as linked records. The entire scene must be resident in memory before it is run for several reasons: One, the design of the DAZZLER board uses main memory as the refresh memory, requiring memory accesses to go through the main bus. (There is not enough bus bandwidth left for simultaneous diskette access.) Two, even if this were possible, the diskette access time plus the CPU setup time would be too slow (even if overlapped, interrupt-driven, double-buffered I/O were implemented). Fortunately, the refresh memory is only 2K bytes. In addition, records in memory contain a fourth (one quadrant) of the frame. Therefore, only changed quadrants need be saved in memory.

In order to achieve the required speed for achieving real-time animation, entire quadrants are paged into the refresh memory using the Z80 move memory instruction, LDIR. Due to the limited CPU power plus the use of interpretive PASCAL, the display refresh frame buffer itself is used to hold the representations of the objects displayed on the screen (instead of using external object record definitions). Objects on the screen are pointed to by outlining a temporary rectangular "window" with the BITPAD stylus.

In addition to the pixel data in the quadrant, the records in memory contain the quadrant number, and a delay time (in milliseconds) to allow the current frame to remain on the screen for the given time. Two display buffers are used to overlap display of the current quadrant with access of the next quadrant, and to simplify moving objects around on the screen.

## Input of Commands and Data

Full command names are displayed on the CRT for each level of a command hierarchy. Except for some special commands, commands are entered with the first character of the command. Special commands are displayed with angle brackets around the command name - i.e. <EXIT>. The BITPAD (which has a resolution of 2200 X 2200) is used as the input device for entering all commands and data. The lower portion of the BITPAD has a menu (pasted on) containing the alphabet, numbers, color names, and the special commands spelled out in full. Any one of the displayed commands may be selected by pressing the stylus inside the box surrounding the first character of the command (or the box containing the

name of a special command - see Figure 2). When a command is executed, further data (if required) is requested by a prompt on the CRT. This input data is normally the selection of a point (or points) on the BITPAD corresponding to a point on the display (TV). Since the resolution of the BITPAD is higher than the display, the points are mapped (divided down) to the resolution of the display. When it is desired to return to a higher level of the hierarchy, the special command (EXIT) in the lower left-hand corner is selected.

## Description of Commands

The program is initiated by executing a file called ANIMAL. A greeting is displayed on the CRT followed by a display of the top-most level of the command hierarchy. The commands at this level are:

Draw Fred Animate Changemode

<STOP PROGRAM>

Draw ("D" box) provides a set of commands for drawing objects on the screen in the current frame.

Fred ("F" box) stands for frame editor, and provides a set of commands to alter the current frame.

Animate ("A" box) provides a set of commands to run the current scene at real-time, "walk" the scene one quadrant at a time, or to save and retrieve scenes from diskette.

Changemode ("C" box) provides a set of commands to select high resolution (128 X 128 with one color plus black) or low resolution (64 X 64 with 16 colors). In addition, black and white may be selected with high resolution, and 16 grey scales may be selected with low resolution.

<STOP PROGRAM> (box located in bottom right corner) is used to terminate the program.

## Draw Subcommands

When draw is selected, the desired color (or grey scale) is prompted and selected with one of the 16 boxes containing the color names, followed by the subcommand names displayed as follows:

Dot Line Area Curve Paint

Triarea <EXIT>

After selection of one of the subcommands, a cursor will appear on the display and will follow the stylus around whenever it is held near any point corresponding to the display area.

Dot ("D" box) prompts for one point. Selection of the point is made by pressing down the stylus on the BITPAD when the cursor is located at the desired point on the display. One pixel (picture element) will then appear at the desired location in the selected color.

Line ("L" box) prompts for the two end points of the desired line. The two points are selected in the same manner as in the Dot subcommand.

Area ("A" box) prompts for one point and fills an area with the color selected. The surrounding area fills up with the new color until boundaries on all sides are reached. A boundary is a line or point containing a different color from the original color at the point selected.

Curve ("C" box) draws an arc or a complete circle by prompting for three points. The first point is the center of the arc or circle to be drawn. The second and third points are the two end points of the arc. If the second and third points are selected at the same location, a complete circle is drawn.

Paint ("P" box) is a continuous drawing mode (has no prompts). Dots (pixels) are placed on the screen where and whenever the stylus is held down and moved across the BITPAD. This gives the effect of a paint brush being moved across the screen. This mode will continue until a point outside of the display area is selected. The paint mode provides the creator or artist with maximum self-expression, and is used when none of the other draw subcommands can create the desired shape or form. "Painting" takes a little practice to get used to, but once learned becomes a very powerful tool. Besides animation, "painting" may also, and in fact, has been used by an artist to create very impressive modern art (see last page).

Triarea ("T" box) prompts for three points and draws a triangle

defined by the three points and fills it with the selected color.

<EXIT> ("EXIT" box) returns to the main command level.

Fred Subcommands

When Fred the frame editor is selected the following subcommands are displayed:

Fill Erase Move Rotate <EXIT>

Fill ("F" box) prompts for a color and a window, and fills the window with the selected color.

Erase ("E" box) displays the next lower sublevel of commands as follows:

Screen Window

If screen ("S" box) is selected, the entire screen is erased. If window "W" box) is selected, then the corners of the window are prompted, and the area inside the window is erased. (Erasing can also be done by "painting" an area with the erase color--first color box on left).

Move ("M" box) prompts for a window and destination point, and then moves the pixels in the window to the destination. (Only one destination point is required and becomes the new location of the corner entered first.) This subcommand takes a while to execute and displays the message "Translating..." while the window is being moved.

Rotate ("R" box) prompts for the window, number of degrees to rotate, and the point of rotation. This subcommand also takes awhile and displays the "Translating..." message.

(Note, the time it takes to move or rotate a window is directly proportional to the size of the window. A scale subcommand to make objects bigger or smaller might be added later, but is non-trivial to implement on a raster scan display such as a TV.)

<EXIT> ("EXIT" box) returns to the main command level.

Animate Subcommands

The animate command displays the following subcommands:

Walk Run Beginning Save Purge

Fill Dump <EXIT>

Walk ("W" box) retrieves the next sequential record from memory and displays it in the proper quadrant. This subcommand provides a means of "walking" through the frames for inspection and/or editing.

Run ("R" box) runs the frames (quadrants) at full speed and thus animates the scene.

Beginning ("B" box) resets the next record (quadrant) to be displayed back to the beginning of the scene for rerunning or rewalking. The beginning quadrant is not displayed in order to allow new frames to be inserted at the beginning.

Save ("S" box) saves one of the quadrants displayed on the screen at the end of the current scene (or as the only quadrant if none existed prior) in memory. A delay time in milliseconds (1 thousandth of a second) is then prompted. The delay time is entered using the number boxes followed by the enter ("ENTER") key. Remember this delay will occur prior to accessing this quadrant on subsequent walks or runs.

Purge ("P" box) throws away the entire scene in memory and releases all of the memory used back to the system.

Fill ("F" box) loads a scene from a diskette at the end of any existing scene in memory. Multiple scenes may be strung together to form one long scene using this subcommand. The name of the scene corresponds to a diskette file name, and is selected by entering the characters of the file name using the alphabet and number boxes, followed by the enter key ("ENTER").

Dump ("D" box) dumps the entire scene to a file on the diskette. The filename is entered in the same manner as in the fill subcommand.

<EXIT> ("EXIT" box) returns to the main command level.

Both the PASCAL and ASSEMBLY LANGUAGE source code for ANIMAL are available on diskette from the author at the address shown.

What's on the Horizon?

CROMEMCO has come out with a new SUPER DAZZLER board having much higher resolution (up to 756 X 484 pixels using 48K of memory), and has solved the problem of simultaneous refresh and disk I/O by using a two-port memory. One port is used for refresh and the other for CPU access (through the main bus). In order to use the maximum resolution of the SUPER DAZZLER (which requires 48K of memory) scenes would have to be paged in from a fast external device (such as a hard disk) instead of using main memory. In order to be able to display one frame while retrieving the next; and to simplify the moving of objects around the screen, an additional 48K of display memory would be required. The 96K of memory plus a hard disk would double the cost of the system (from $10,000 to $20,000).

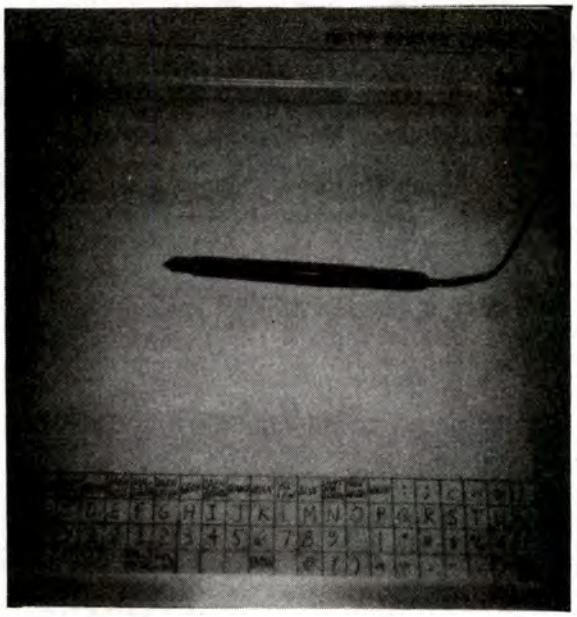


Figure 2
"BITPAD" Digitizer

In the next few years we might see CCD's (charge couple devices) used as the refresh memory, and bubble memory as the external storage device. This would reduce the cost down to a reasonable level. In addition, faster microprocessors (such as the ZILOG Z8000 or the MOTOROLA MC68000) will be readily available to reduce the calculation time required for animation.

Figure 1
COMAGRAPH COMPUTER SYSTEM

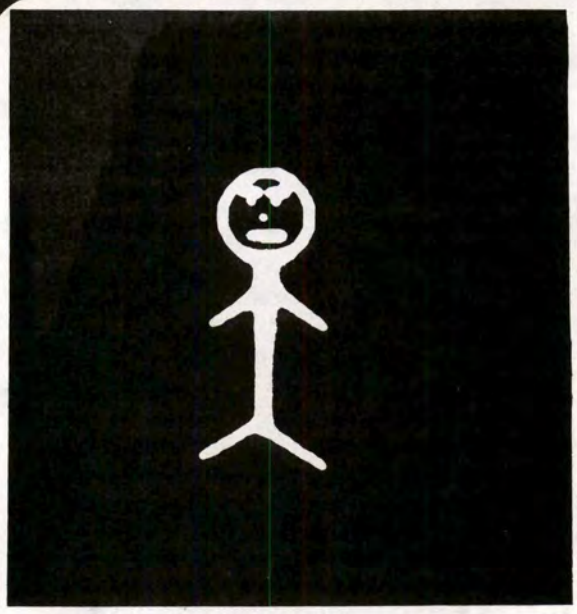


Figure 3

Figure 4

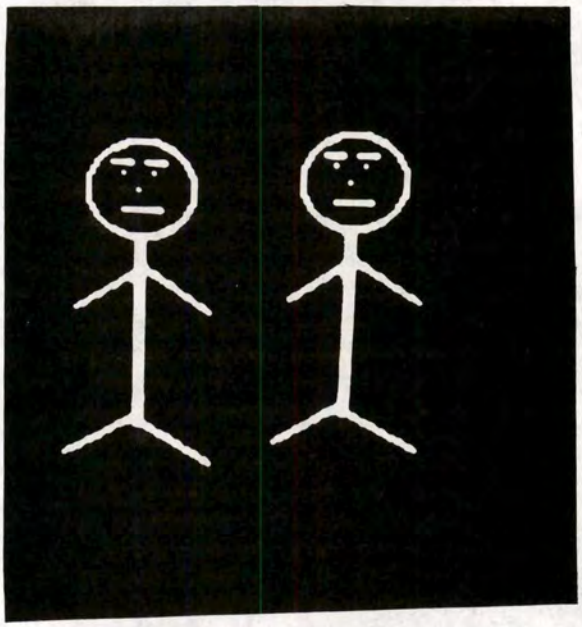


Figure 5

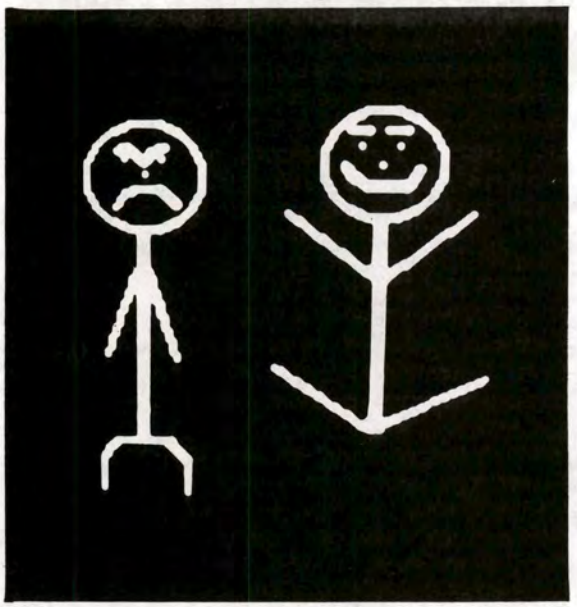
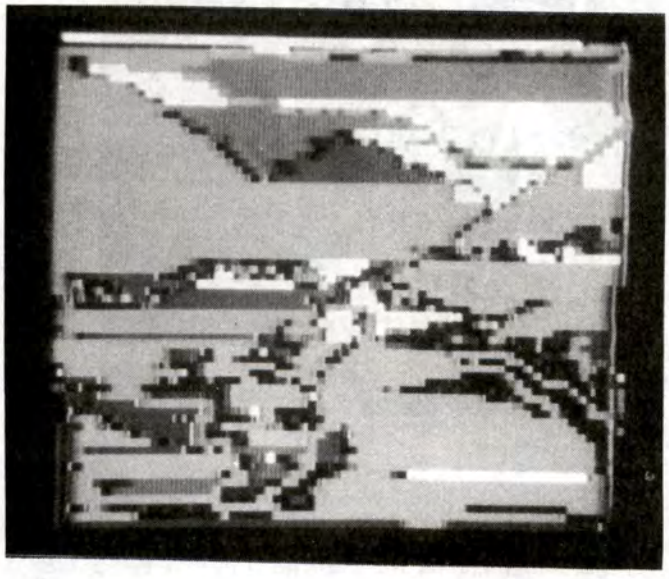


Figure 6

Figure 3 was generated by using the Draw Arc, Draw Line, and Draw Dot subcommands. Figure 4 was generated using the Fred Rotate subcommand. Figure 5 was generated by first using the Fred Move Window subcommand to move the figure to the right side of the screen; followed by the Animate Save quadrants subcommand to save quadrants 2 and 4; followed by another "move" to the left side of the screen; followed by an Animate Beginning and two Animate Walk subcommands to retrieve the two saved quadrants. Figure 6 was generated using the Fred Erase window and Draw Paint subcommands. A simple animated scene to make the arms and legs move with facial expression changes was generated (not shown) by alternating between "saving" and "retrieving" the frames in figure 5 and 6 enough times, and then animating the scene using the Animate Run subcommand.



These "paintings" were created using the Draw Paint subcommand.

# NPS MICRO-COBOL

LT Mark S. Moranville
Naval Postgraduate School
Monterey, CA 93940
PH 408-646-2449

## Abstract

A compiler-interpreter for a subset of ANSI COBOL has been implemented on an 8080 or Z80 under the CP/M operating system. The implementation provides all nucleus level constructs and file options from ANSI COBOL. The language was implemented through a compiler and run-time package which can be executed in 20K bytes of main memory. A program consisting of 8K bytes of intermediate code can be supported on this size machine. The programs that make up the compiler and run-time package require 50K bytes of disk storage.

## Background

The NPS MICRO-COBOL compiler/interpreter is the result of thesis research at the Naval Postgraduate School that commenced in 1976 to demonstrate that it was feasible to implement a COBOL compiler on a microcomputer.

The original design was based on HYPO-COBOL which is a Department of the Navy approved subset of COBOL, designed to place minimal requirements on a system for compiler support. The definition of HYPO-COBOL was prompted by the need for a small scale package that could exist in a microcomputer environment. The problem with using one of the existing COBOL level specifications was that the level structure was oriented toward batch environments on systems of various sizes, permitting COBOL implementations of various degrees of sophistication. Many of the features supported by even the lowest levels of COBOL are not applicable to an interactive, single user system such as CP/M. Additionally, the COBOL language being highly verbose makes high demands on systems with regard to parsing time and storage space MICRO-COBOL was designed to reduce the size of the compiler and to eliminate functional redundancy provided by multiple options in the language statements. It should be noted that MICRO-COBOL is a proper subset of ANSI COBOL.

## Goals and Objectives

The original objective of the NPS MICRO-COBOL project was to demonstrate that it was feasible to implement a COBOL compiler on a microcomputer. This goal was reached and a second objective of producing a production quality COBOL compiler which could be used to teach COBOL programming on microcomputers was established. Since the compiler was to be used in a teaching environment minimizing compile time was considered more important than minimizing execution time. In addition since many CP/M systems do not support 64K of main memory minimizing the size of the compiler and interpreter was considered important.

## Language Features

NPS MICRO-COBOL is a subset of ANSI COBOL. The following paragraphs describe the most important features of the language.

## Syntax

A complete syntactic specification of the language is contained in the appendix. The major deviations from ANSI COBOL are: (1) In general only the short form of reserved words are supported, for example: PICTURE is not a legal reserved word. PIC is the only legal form for this construct. (2) The input to the compiler does not need to conform to ANSI COBOL format. Free form input will be accepted as the default condition with the exception that any line with a '*' in column one is treated as a comment line. If desired, sequence numbers can be entered in the first six positions of each line. However, a toggle needs to be set to cause the compiler to ignore the sequence numbers. If the sequence number toggle is set then all lines in the program must have sequence numbers or six blanks in columns one to six.

## Data

NPS MICRO-COBOL supports the standard COBOL PIC clause entries (i.e. data characters: A, X, 9 operational characters: S V and insertion characters: Ø, ', B, ., $, +, -, Z, *). It also supports standard numeric data definitions up to eighteen digits in DISPLAY format. All numbers are stored in DISPLAY format and converted to packed BCD format only for arithmetic operations. This simplifies the conversion of numbers by the compiler. Although the syntax for the SYNC clause is supported no actions are taken by the compiler since numbers in display format are always aligned on even word boundaries in 8-bit machines. One dimensional tables are supported at the elementary level.

## Input-Output

NPS MICRO-COBOL supports I/O to the console and disk files.

Console I/O is accomplished by using the ACCEPT verb to read data from the console and the DISPLAY verb to write data to the console. Entire records or elementary data items can be read or written using these verbs.

Disk I/O is accomplished by using the READ verb to read data from a disk file and the WRITE verb to write data to a disk file. Sequential and relative (random access) files are supported. All CP/M files created using the editor can be accessed either sequentially or randomly except variable length record files which can only be accessed sequentially. The physical file structure for random access or sequential files does not change under CP/M however, the types of reads and writes allowed by a MICRO-COBOL program depends on what type of organization and access method has been selected. The implementor name specified in the select clause follows standard CP/M file name format including drive specification, however, the printer can not be specified. If output is intended for the printer it is first written to a disk file and then sent to the printer using the PIP command. Files that are written using the ADVANCING clause are print files and can not be read. Files must be closed if they have been written, however, the standard COBOL requirement to close an input file prior to the end of processing does not exist.

## Implementation

### Compiler

The compiler is a one pass compiler that scans and parses MICRO-COBOL source programs, and generates intermediate code for the MICRO-COBOL interpreter. The scanner is similar to other scanner implementations. The parser is an LALR(1) table driven finite state automation. The compiler reads the source program from a disk file extracts needed information for the symbol table and writes intermediate code to a disk file. The compiler consists of two major modules: PART ONE and PART TWO.

### PART ONE

PART ONE of the compiler performs three major functions. First it establishes the interface between the compiler, the source file, the intermediate code file, and the module which reads and passes control to PART TWO of the compiler. Second, it scans and parses the source program from the beginning up to the PROCEDURE DIVISION. Third, it generates output consisting of the symbol table (saved in main memory) and any data initialization intermediate code.

### PART TWO

PART TWO of the compiler scans and parses the source program starting with the word PROCEDURE through the end of the program. PART TWO overlays PART ONE of the compiler. This is possible because of the nature of the COBOL language. All variables in COBOL are declared prior to the beginning of the procedure division. Therefore the symbol table can be built by PART ONE and used by PART TWO. PART TWO generates intermediate code for all executable instructions in the PROCEDURE DIVISION of the program.

### Interpreter

The interpreter actually represents a pseudo-machine which interprets and executes the MICRO-COBOL intermediate code. The pseudo-machine contains a program counter, three thirty-six digit registers for arithmetic operations, a subscript stack used to compute subscript locations, and a set of flags which are used to pass branching information from one instruction to another.

The machines instruction format consists of multiple parameter operations which contain all the information required to perform one complete action by the language. For example: the COBOL statement

MOVE A TO B

might generate the following intermediate code

MOV <ADDRESS OF B><ADDRESS OF A><length of A>
<length of B>

The interpreter consists of two modules BUILD and INTERP. The BUILD program is loaded first and reads in the intermediate code, initializes all memory locations requiring initialization, and resolves all unresolved address references. When these actions are completed the BUILD routine copies the INTERP programs into memory and transfers control to it.

### Memory Organization

The memory of the pseudo-machine is divided into three major areas: 1.) the data area is established by the DATA DIVISION statements of the source program, 2.) the constants area which is established by both the DATA and PROCEDURE DIVISIONS of the source program, and 3.) the code area which is established by the PROCEDURE DIVISION.

The data area is the lowest area in the pseudo-machine. This area contains the storage for identifiers declared in the DATA DIVISION. Additionally, the data area contains the File Control Block (FCB) and the buffer space (128 bytes) for all files declared in the source program.

Immediately following the data area is the code area. This contiguous area of storage contains all executable code generated. The constants area is located in high memory of the pseudo-machine. This area contains all edit field masks as well as all numeric and non-

numeric literals. Figure 1 illustrates the memory organization of the pseudo-machine.

## Conclusions

This project has resulted in the construction of an applications oriented COBOL compiler for microcomputers with 20K bytes of memory. The compiler compiles at an average speed in excess of 400 lines per minute on a Z-80 CP/M system. In comparison to full ANSI COBOL compilers, it compiles faster and uses less memory and still supports most of the constructs that support structured COBOL programming.

## Acknowledgements

NPS MICRO-COBOL is an outgrowth of on going thesis research at the Naval Postgraduate School. The original design was completed by A. S. Craig in 1977. Further development was accomplished by P. R. Mylet and John Pierce in 1978 and J. Farlee and M. L. Rice in 1979. Further development was accomplished as part of course work by Doug Losket, Hal Powell, Dennis Jackson, Doug Stowers, Robert Hartell, Ed Rynne, A. O. Cravets, and J. S. McRoberts.

Professor Gary Kildall served as thesis advisor for the first two phases and the author served as thesis advisor on the last thesis.

## List of References

1. Craig, A. A., MICRO-COBOL An Implementation of Navy Standard HYPO-COBOL for a Microprocessor based Computer System, Master's Thesis, Naval Postgraduate School, Monterey, California, 1977.

2. Mylet, P. R., MICRO-COBOL A Subset of Navy Standard HYPO-COBOL for Microcomputers, Master's Thesis, Naval Postgraduate School, Monterey, California, 1978.

3. Farlee, J. and Rice, M. L., NPS MICRO-COBOL An Implementation of Navy Standard HYPO-COBOL for a Microprocessor Based Computer System, Master's Thesis, Naval Postgraduate School, Monterey, California, 1979.

4. Department of the Navy HYPO-COBOL Language Specifications Automatic Data Processing Equipment Selection Office, Washington, D. C. 20376.

PSEUDO-MACHINE ORGANIZATION

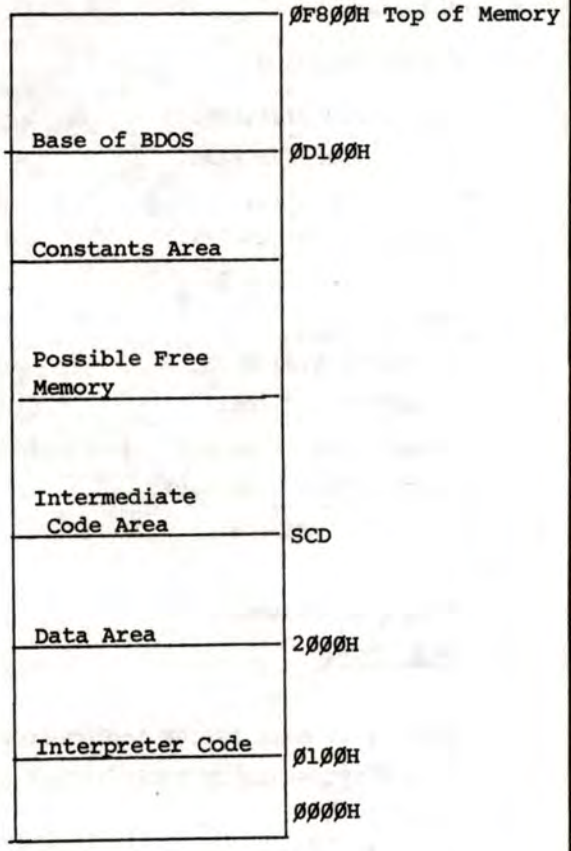



Figure 1

## NPS MICRO-COBOL SYNTAX

This section contains the syntactic specification of each element in NPS MICRO-COBOL. The following conventions are used in explaining the formats: Elements enclosed in broken braces <> are themselves complete entities and are described elsewhere. Elements enclosed in braces {} are choices, one of which has to be used. Elements enclosed in brackets [] are optional. Elipses "..." indicate that the preceding item may be optionally repeated. All elements in capital letters are reserved words and must be spelled exactly. User defined names appear in lower case. These names are restricted to 15 characters in length and the first character must be an alpha character.

### IDENTIFICATION DIVISION

```
IDENTIFICATION DIVISION.
PROGRAM-ID. program-name.
[AUTHOR. <comment>.]
[DATE WRITTEN. <comment>.]
[SECURITY. <comment>.]
```

### ENVIRONMENT DIVISION

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. <comment>  [DEBUGGING MODE].
OBJECT-COMPUTER. <comment>.
[INPUT-OUTPUT SECTION.
FILE-CONTROL.
   <file-control-entry>  . . .
```

### FILE CONTROL ENTRY

```
1.
   SELECT file-name ASSIGN implementor-name
   [ORGANIZATION SEQUENTIAL] [ACCESS SEQUENTIAL].
2.
   SELECT file-name ASSIGN implementor-name
   ORGANIZATION RELATIVE [ACCESS {SEQUENTIAL [RELATIVE date-name]}].
                                 {RANDOM RELATIVE data-name      }
```

### DATA DIVISION

```
DATA DIVISION.
[FILE SECTION.
[FD file-name
   [BLOCK integer-1 RECORDS]
   [RECORD [integer-2 TO] integer-3]
   [LABEL RECORDS {STANDARD}]
                  {OMITTED }
[<record-description entry>] ...] ...
[WORKING-STORAGE SECTION.
[<record-description entry>] ... ]
```

```
DATA DESCRIPTION ENTRY
    level-number {data-name}
                 {FILLER   }
    [REDEFINES data-name]
    [PIC character-string]
    [USAGE {COMP}   ]
           {DISPLAY}
    [SIGN {LEADING} [SEPARATE]]
          {TRAILING}
    [OCCURS integer]
    [SYNC [LEFT ]]
           RIGHT
    [VALUE literal].
PROCEDURE DIVISION
    1.
       PROCEDURE DIVISION
       section-name SECTION.
       [paragraph-name. <sentence> [<sentence> ... ] ... ] ...
    2.
       PROCEDURE DIVISION
       paragraph-name. <sentence> [<sentence>  ...] ...
VERBS
    ACCEPT identifier
    ADD {identifier} [{identifier-1} ] TO identifier-2
        {literal   } {literal-1   }
        [ROUNDED] [SIZE ERROR <imperative statement>]
    CLOSE file-name
    DELETE file-name [INVALID<imperative statement>]
    DISPLAY {identifier} [{identifier-1}]
            {literal   } {literal-1   }
    DIVIDE  {identifier} INTO identifier-1
            {literal   }
            [ROUNDED] [SIZE ERROR<imperative statement>]
    EXIT
    GO procedure-name
    GO procedure-name [procedure-name-1] ...
       procedure-name-20
       DEPENDING identifier
    IF <condition> {<imperative statement>...}
                   {NEXT SENTENCE}
       [ELSE {<imperative statement>}
             {NEXT SENTENCE}
    MOVE {identifier} TO identifier-1
         {literal}
    MULTIPLY {identifier} BY identifier-1
             {literal   }
             [ROUNDED] [SIZE ERROR <imperative-statement>
```

```
OPEN {INPUT} file-name
     {OUTPUT}
     {I-O   }

PERFORM procedure-name [THRU procedure-name-1]

PERFORM procedure-name [THRU procedure-name-1]
       {identifier} TIMES
       {integer   }

PERFORM procedure-name [THRU procedure-name] UNTIL
        <condition>

READ file-name INVALID <imperative-statement>

READ file-name END <imperative-statement>

REWRITE record-name [INVALID imperative-statement]

STOP {RUN}
     {literal}

SUBTRACT {identifier} {identifier-1}  FROM identifier-2
         {literal   } {literal-1   }
         [ROUNDED] [SIZE ERROR <imperative statement>]

WRITE record-name [{BEFORE} ADVANCING {INTEGER}]
                  {AFTER }           {PAGE   }

WRITE record-name INVALID <imperative-statement>
```

CONDITIONS

```
RELATION

 {identifier}{[NOT] GREATER} {identifier-1}
 {literal   }{[NOT] LESS   } {literal-1   }
             {[NOT] EQUAL  }

CLASS

  identifier [NOT] {NUMERIC}
                   {ALPHABETIC}
```

IMPERATIVE-STATEMENT

An imperative statement is one that has no conditional action (i.e. ADD A TO B).

# AN INTRODUCTION TO THE WONDERS OF PASCAL

James Gagné, M.D.
President
DATAMED RESEARCH
1433 Roscomare Road
Los Angeles, CA 90024
(213) 472-8825

This is an outline of my talk; for more detail see my article in the recent Kilobaud Microcomputing.

## I. Structured Programming

A. Concept of clear, controlled program flow

B. Recognized control structures

1. Sequential statements
2. FUNCTIONS, PROCEDURES (call by value, call by reference)
3. IF...THEN
4. IF...THEN...ELSE
5. CASE...OF...
   a: ...
   b: ...
   :
   :
   n: ...
6. REPEAT...UNTIL
7. WHILE...DO
8. FOR...TO/DOWNTO...DO

C. Top-down programming

## II. Fundamental Concepts

A. Statement

B. Block

C. Single-pass compiler compatability

## III. Data Types

A. Integer

B. Real

C. Character

D. Arrays

E. Records; case variant records

F. Pointers

G. Files

## IV. Input and Output

A. Predeclared files INPUT, OUTPUT

B. READ, READLN

C. WRITE, WRITELN

D. Simple inputs; structured outputs

E. EOLN, EOF

F. GET, PUT

## V. Standard Functions

A. ABS (X)

B. SQR (X)

C. SIN (X)

D. COS (X)

E. EXP (X)

F. LN (X)

G. SQRT (X)

H. ARCTAN (X)

I. ODD (X)

J. TRUNC (X)

K. ROUND (X)

L. ORD (X)

M. CHR (X)

N. SUCC (X)

O. PRED (X)

VI. A Word About Dynamic Variables

VII. A Brief Description of UCSD Pascal

# A NEW, MINIMAL-COST SOFTWARE CLUB

# FOR USERS OF UCSD PASCAL

James Gagné, MD
President
DATAMED RESEARCH
1433 Roscomare Road
Los Angeles, CA 90024
(213) 472-8825

## Abstract

Datamed Research is announcing a new service for users of Pascal, particularly UCSD Pascal (trademark, the Regents of the University of California). We are forming a users group, very similar to the popular CP/M Users Group, and will distribute donated Pascal software on 8-inch floppy disks in both UCSD and CP/M disk formats, at a very low cost.

This paper briefly reviews the features of UCSD Pascal, discusses the existing Pascal Users Group and why this Datamed Research service is required, and describes the philosophy and logistics of this new software sharing group.

## Introduction

The UCSD Pascal language system is one of the most sophisticated microcomputer software systems available today. Because of the ease with which one can write and maintain high quality programs of most types, from systems software to business applications to games, it is becoming increasingly popular. Already a number of other Pascal implementations have appeared for microprocessors, though none so complete. The UCSD system promises to be the vanguard of an enormous interest in Pascal in the coming decade.

UCSD Pascal compiles its programs to P-code, designed for a hypothetical 16-bit stack machine that must be emulated in software on most microprocessors. Even though this P-code interpretation system runs three times slower than assembly language, it is much faster than any other interpretive language available for micros, and has the additional advantage that P-code occupies approximately one-third the space of native machine language. (Now, however, at least two manufacturers have produced computers that execute P-code directly, as their machine language, offering extremely rapid execution.) In addition, once the P-code interpreter has been installed, programs written in UCSD Pascal may be run on any microprocessor without modification. Even the disk formats are the same, except for the minifloppies used for the Apple, North Star, or TRS-80. So disk software may be freely shared among users of such diverse machines as a PDP-11 or an 8080.

## The Pascal Users Group

It would seem natural for a large users group to arise to share software. To date, however, only the original Pascal Users Group ("PUG") serves this function. The PUG sprang into existence in 1976 to serve as a forum for the then few hundred Pascal programmers. Primarily, they support the standard language based on the Jensen and Wirth Pascal User Manual and Report, discuss the horrors certain programmers would like to language to make it fit their needs, and report on available Pascal implementations and programmer opportunities. Only secondarily does the PUG disseminate software (based on Jensen and Wirth Pascal), although since 1978 the PUG has published several superb "software tools", such as automatic formatters to print programs clearly and a large and extremely sophisticated text formatter. They can be reached at the University Computer Center, 227 Experimental Engineering Building, 208 Southeast Union Street, University of Minnesota, Minneapolis, Minnesota 55455. A subscription is now $6 per academic year. I suggest you start your collection from the 1977-1978 year. (Back issues are available at the same annual rate.)

The PUG newsletter presents several difficulties to the user of UCSD Pascal who wishes to share software. First, relatively little software is published, albeit that which does appear is of high quality. Second, the programs must be adapted for UCSD Pascal. It turns out that this task is trivial, and concerns primarily disk i/o. Third, the programs are a little slower because the nonstandard UCSD functions like strings aren't used; strings run more quickly on a UCSD system than the character-by-character approach needed by standard Pascal. The overwhelming problem, however, is that

the Pascal Users Group publication, the "Pascal News," appears only on PAPER (yecch!). And even if they did use machine-readable media, most members run large computers that talk to each other via tape. So you have to type the software into the machine on your own. I can assure you, having done it, that this is no mean task.

A UCSD Pascal Users Group on machine-readable media

In the interests of promoting the more widespread use of Pascal and building up a sizable program base, Datamed Research is announcing the formation of a UCSD Pascal users' group. It will take a form very similar to the highly respected CP/M Users Group: all offerings will be on 8-inch, single density, IBM-compatible soft-sectored floppies, offered virtually at cost ($10 per disk). Software will be donated by interested users. Software donors will receive a free disk volume of their choice in acknowledgement of their donation. For software to be accepted for distribution it MUST (a) work (no known bugs) and (b) come with at least adequate documentation on the disk. Further, with rare exceptions it must be supplied in source code to allow other users to adapt it to their systems.

There is one exception to the requirement for source code. If you were developing a sophisticated program that you hoped to sell, but needed assistance in discovering remaining bugs and system incompatibilities, you might donate the interim P-code to the users group. Then, users who first discovered a particular problem or gave other feedback you considered valuable would receive an incentive such as a discount on the completed program. In the meantime, users would benefit from a program that worked most of the time.

Potential sources of Pascal software abound; by no means must you donate only original work. There is a mountain of public-domain Basic software that is easily adapted to Pascal. In the process, you can usually spruce up the program a good deal, because Pascal is so much easier to work with than Basic. It will be important, in addition, for the users to begin a library of Pascal procedures and functions to handle the more common programming problems. For example, we need a set of mathematical functions for complex variables, statistical functions, and basic business software support (routines to translate integers into dollars and cents and vice versa) to realize the full power of the language. I am presently writing a program which will automate the production of CRT screen masks and (more exciting) handle data input from the CRT directly. This program will accept in simplified form the desired CRT mask directly from the system editor, plus needed data about the variables to be typed in by the user. The program will generate Pascal source code for incorporation into your applications programs and handle automatically things like goof proofing (preventing the program from crashing if the wrong types of data are entered), variable declarations, saving the CRT mask data within the procedure or on the disk as you wished, etc.

By the way, since it is relatively trivial to transfer text from the UCSD disk format (similar to the PDP-11 RT11 format) to CP/M, all volumes will also be available on CP/M disks for those using CP/M-compatible Pascal. However, you will be on your own to get the programs to fit into your memory (remember that native code versions are three times larger) and adapted to your system. We will be happy, though, to accept Pascal software on CP/M disks if it can be readily adapted to the UCSD system.

As of December 1, 1979, there is enough software and other goodies for the first two volumes; contact Datamed Research directly for the exact number of volumes and their contents. All documentation will be on the disks. The disks will include:

1) The powerful pretty printer and formatting programs, to beautify Pascal source code, from the Pascal News Vol. 13.

2) A Pascal driver for a D. C. Hayes modem, so your computers can talk with one another. This should not be hard to modifiy for other modems. Two versions of this program have been prepared by different authors.

3) A file printer offering several options in page headings and page numbering, as well as single to quad line spacing. Good for programs and manuscripts.

4) An assortment of games, ranging from CHASE to SKYLANES. (As of yet, no STARTREK.)

5) Two programs in Pascal to convert UCSD-format disks to the CP/M format, and vice versa.

6) A nifty restructuring of the 8080/Z-80 interpreter and BIOS to support disks formatted with 512-byte blocks (single or double density), for a 23% greater disk capacity and a breath-taking increase in disk access speed, as well as a slight shrinkage of the interpreter. The modified BIOS accepts 128-byte and 512-byte sectored disks transparently, although you have to reboot if you change disk density. In addition, the BIOS contains complete cursor-handling routines for a dumb terminal such as an ADM-3A.

7) If you've tried to get up UCSD Pascal

in a CP/M environment, you've noticed that the UCSD system is considerably more demanding of your BIOS than was CP/M. We will have notes on providing the extra functions required, plus many of the aspects of the 8080/Z-80 implementation not documented.

Possible Future Directions

Since there is little question that Pascal will grow by leaps and bounds, the major question is how to keep up with the machine-readable formats required by the various micros and minis utilizing Pascal. For now, we are limited to standard 8-inch floppies, and if there were enough demand I would consider distributing the volumes in hard-copy form at a modest increase in price for those who could not utilize full-size disks. But perhaps there are others with the hardware capabilities of transferring programs from one format to another (e.g, to North Star or Apple disks) or who are willing to copy the smaller disks for distribution. If we can provide these services, then other formats could be distributed as well.

Although programs would be the main emphasis, I hope to have other features on the disks as well as software. Information on programming tips would certainly be a useful addition. For example, there are a number of "hidden gotchas" in the UCSD system, as well as features that are inadequately documented. Also, I don't think it's clear just which programming techniques are the most portable from system to system. For example, including a PAGE (OUTPUT) within a program in my system clears my screen or causes a formfeed on the printer. It does nothing in systems that do not recognize an ASCII formfeed character (12 decimal). One could clear the screen on ANY UCSD system by jumping to the bottom of the screen, doing 24 WRITELN's, then jumping to the upper left of the screen. Is there an easier way that will always work?

Finally, we should share algorithms and reviews of commercial Pascal software.

For Further Information

You can find out more about the present status of the users group by writing the address at the beginning of the paper and including a selfaddressed, stamped envelope. Alternatively, 8-inch floppies can be ordered at $10 per volume; the number of volumes available at the time this is published will be at least two. Make sure you specify UCSD or CP/M format.

# HOME BUS STANDARDS ASSOCIATION, WHAT IS IT AND WHAT DOES IT MEAN?

Robert J. Richardson
Director, Consumer Electronics Department
SRI International
333 Ravenswood Ave.
Menlo Park, CA. 94025
(415) 362-6200 Ext. 5448

## Introduction

The personal or home computer has often been mentioned in conjunction with the concept of overall management of home environmental control and monitoring systems, home entertainment, and information systems. One of the factors inhibiting wide acceptance and realization of this concept is the fact that connecting to control and monitoring points of the home environmental systems is a complicated and costly process. The concept of the Home Bus Standards Association provides an optimum approach to eliminating the economic and organizational inhibiting factors.

The Home Bus Standards Association (HBSA) is a non-profit (IRS501C3) membership organization for the purpose of establishing a widely accepted set of communication protocols, allowing all household electrical devices to interact as parts of a modular intelligent network, using powerline carrier digital packet radio transmissions.

The consumer benefits of a Home Bus system include direct savings from reduced energy consumption, improved personal safety, and the convenience of remote and automatic control and monitoring of every system in the home. These benefits can be provided at little or no additional cost to the consumer, due to the recent advances in microelectronic technology combined with high volume production of standardized "Bus Compatible" components suitable for use in a broad variety of applications.

HBSA's objectives are to serve as a neutral focal point for development of an industry-wide monitoring and control signal language, and to provide fundamental public education informing consumers about the advantages of having Home Bus type technology.

HBSA is needed because no current organization covers the diverse spectrum of products potentially benefiting from bus compatibility: appliances, heating and air conditioning equipment, home entertainment devices, utility meters, the telephone, lights, locks, alarms, and so on.

Through HBSA, the central nervous system of the computerized home of the future can be quickly defined, thereby facilitating the linkage of advanced technology's capabilities with immediate public needs.

## HBSA--Why?

"Intelligent" products are proliferating in the consumer field at an ever increasing rate. Microprocessors are finding their way into everything from door bells, light switches, furnace and air conditioning systems, to blenders and microwave ovens. New security devices, clock radios, stereos, and the ubiquitous personal computer contain powerful microcomputers whose current capabilities far exceed the effective applications. By the end of the 1980s, according to many observers, microprocessors will be found in almost every electrical product selling for over $20, and the typical home will contain a couple of dozen devices incorporating "electronic intelligence."

Once a manufacturer has made the decision to include a microprocessor in a consumer product, the challenge then becomes finding ways to usefully exploit its tremendous power. The recent advances in semiconductor price/performance have been so vast that, for all practical purposes, "intelligence is free." Now the "problem" is applying these capabilities to the service of genuine consumer needs.

One of the major aspects inhibiting the true potential for effective utilization of these products for the benefit of consumers is the fact that they are unable to receive changing control directives and report current status.

Nearly every major consumer electronic equipment manufacturer and every utility is exploring individual approaches to remote control and status reporting for their individual products and services. Various remote control products and techniques have already been introduced to the market.

Transferring this concept to the consumer market requires establishment of a common "language" allowing various products made by various manufacturers to talk with one another. Several large electronics companies have already begun internal programs to develop such a language. If the industry continues in this fashion, the "tower of Babel" syndrome is likely to occur with many different incompatible schemes developing which will ultimately be in conflict, confuse the consuming public, upset the regulatory authorities, and inhibit the growth of interactive systems of all types.

Clearly it is difficult for individual companies and interested individuals and organizations to work together without the necessary unaffiliated, non-profit minded focal point. This is, therefore, the reason for founding the Home Bus Standard Association.

Bus Standard--What Is It?

The term "Bus Standard" is used in the computer industry to describe the means of linking together the modular elements of a system. For example, many computers consist of large boxes filled with stacks of plug-in printed circuit boards. The "Bus" is the set of wires interconnecting all the printed circuits.

During the past few years the phrase "Bus Standard" has broadened to also include methods for tying together other types of electronic equipment. For instance, the "General Purpose Instrument Bus" (IEEE 488) allows a wide variety of laboratory instruments to communicate with each other as parts of an inteligence network. The Home Bus Standard transfers this concept from the laboratory to the home, by defining a system which enables virtually all of the household's electrical and electronic devices to send and receive monitoring and control information.

To avoid the inconvenience of having to install special cables, Home Bus is a packet radio communications protocol. It is designed for FM transmission over the existing 110 volt house wiring in a manner similar to "wireless" intercoms. A "Bus Compatible" product will be instantly "on-line" when plugged into any wall socket.

Role of HBSA

The emergence of multiple incompatible Home Bus type communication protocols would damage the interests of both consumers and business by inhibiting the acceptance of beneficial new technology. A single standard, comparable to our national 110 volt, 60 cycle electric power standard, would be to everyone's advantage.

Establishment of such a standard will most likely occur in one of two ways. The first scenario would be for a very large company to unilaterally announce a communications protocol applying to all of its products, with hopes that other companies will follow the lead by also making their products compatible. The problem with this approach is that other companies may be unwilling to concede the substantial marketing advantages accruing to the originator of the "standard." An example of this response can be found in the incompatible Betamax and VHS formats for video cassette recorders.

The alternative mechanism for creating a standard is a neutral, non-profit association. This approach has several significant benefits. Because the association's standard does not create any unfair marketing advantages or disadvantages for the participating companies, there is less chance of a competitive reaction resulting in an incompatible protocol. Because no single company produces all the items likely to become bus compatible, the association's ability to consider input from all affected business sectors should result in a technically superior protocol for industry-wide use. Finally, the association approach is much less likely to encounter legal problems with the Department of Justice Anti-trust Division or the Federal Trade Commission.

HBSA--Objective

Home Bus Standard Association, Inc., headquartered in Washington D.C., is a non-profit corporation qualified for non-profit and tax exempt status under Section 501 (c) (3) of the IRS code of 1954 as amended.

The objective of the Association is to conduct applied scientific research relating to the development, maintenance, and promulgation of a home bus standard. Initially, the primary objective of the association is to develop and define the optimum standard which will benefit the consumer and allow full utilization of the capabilities of intelligent microelectronics in the various electrical, consumer products found in the homes of today and tomorrow.

Secondarily, once this common standard has been established, the role of the Association will shift to one of promulgation, education, and maintenance of the standard in the community.

Consumer Benefits

The existence of a Home Bus Standard makes possible a variety of features and services to more effectively meet consumer

needs. Such features and services include energy conservation, personal security and convenience systems.

Energy Conservation. Home Bus will carry heating and air conditioning sensor and actuator signals, allowing easy, owner-installed retrofit of more sophisticated and efficient systems. It will also provide for monitoring and control of electric power consumption, reduce utility costs by connecting the gas and electric meters to the telephone--eliminating the need for meter readers. In addition, Home Bus will facilitate automatic control of passive solar devices such as electric window coverings helping to increase the energy efficiency of most existing structures.

Personal Security. This application class includes fire and burglar alarms, control of lights for simulated occupancy, and remote surveillance of children, swimming pools, and the like. Home Bus will also allow medical monitoring for patients who may otherwise require hospitalization or nursing care. For example, many senior citizens living alone would like a portable "panic button," worn on the body so that it is always available to summon emergency aid.

Convenience. In comparison to the previously described life and energy saving benefits, the convenience of remote and automatic control of household systems seem frivolous. Yet these features are probably the greatest source of consumer appeal.

The easiest way to visualize the possibilities is to imagine a cordless handheld device slightly larger than a pocket calculator that is both a wireless telephone and a remote controller for everything in the house: the T.V., stereo, lights, drapes, intercom, heat, door locks, hot tub, clock radio, washing machine, and so on.

It is important to recognize that the system does not require a home computer, though it augments its power if you have one. For example, you go to the department store and only buy a clock radio and a hot tub. If both are Bus compatible, you can send a message saying "Warm the water to 103° and ring me back when you are ready."

Bus compatibility will extend even to the kitchen blender, some of which already contain a microcomputer chip. Because the blender also already has a keyboard, the incremental cost of making it a Bus "talker" is virtually nil.

## Industry Benefits

The existence of a Home Bus Standard offers a number of advantages to the consumer electronic and appliance industries.

First, Home Bus makes possible a new generation of advanced products with significantly greater consumer utility. This will result in an increase in the industry's primary demand.

Second, Home Bus is the ideal vehicle for the consumer electronic industry to make its contribution toward dealing with our national energy crisis. This is important because much of the public now perceives electric appliances as part of the problem, rather than part of the solution.

Finally, Home Bus can serve as the key factor in coupling the enormous power of the new large-scale integrated circuits to the actual needs of the consumer marketplace. It is, in short, the missing link for the computerized home of the eighties.

## HBSA--When?

The Association has initiated a formal membership drive. At the same time, the Association is beginning an information gathering phase to establish contact with those who may contribute technical information, expertise, or other data pertinent to the subject matter.

It is anticipated that within a short time the primary criteria for the technical development will be defined and that the technical development of the protocol will be underway. At that same time, the formation of various evaluation committies which will consider issues of social, operational, and product requirements will be established and will begin developing criteria for considering the optimum choices of the possible approaches to the standard.

While it is impossible to set an exact date for the final standard determination at this time, it is understood that time is of the essence and that the determination would be finalized as soon as possible. Realistically, that final determination cannot be anticipated sooner than six months. It must be established within eighteen months to carry the necessary impact that will benefit all concerned.

HBS--How?

The Association has retained SRI International (formerly Stanford Research Institute) as its prime contractor, due to SRI's existing expertise in almost all the technical and business areas involved in the Home Bus effort. SRI is in the process of concluding a preliminary feasibility study for HBSA.

The various review committees covering such areas as social impacts, human engineering, and the various product categories are being organized by SRI and will be composed of appropriate individuals from institutions and industry with the prerequisite expertise. At various stages in the development process, the alternative techniques and issues will be submitted to the membership for review and comment.

Companies who become members are encouraged but not bound to contribute technical expertise, human interface, product market and business criteria to the Association. SRI will assure that all information submitted will not be attributed or linked to a specific company during the course of the process without specific authorization to do so.

Companies who participate are encouraged but not bound to use the Home Bus Standard, once defined, in their products.

Consequently, from the point of view of a strategic planner, the best solution is establishment of a non-profit trade association tasked with creating, maintaining, and promoting a Bus Standard. This would provide a neutral common ground where representatives of industry, government, and concerned consumer groups could come together for the benefit of all involved.

# A LINEAR SCROLLING CRT WITH STANDARD PARTS

John P. Cater
Manager, Intelligent Systems Engineering
Southwest Research Institute
6220 Culebra
San Antonio, TX 78284

## Introduction

Have you ever heard of speed reading a program on a CRT display? Not likely, because most displays use a method of scrolling which approximates a rachet motion. I call this type of scrolling "racheting". Using some crafty software and a standard CRT controller chip, a CRT display can be easily constructed which provides program listings and visual outputs exactly like the rolling movie credits on your favorite TV show. The following discussion details the software and hardware necessary to get you started on your own rolling CRT display.

## Background

To illustrate the significance of this feature in a CRT display let me remind you of the many times that you have typed "LIST" into your personal computer. You then sat there in pure bewilderment as your program flashed by your eyes at breakneck speed. After several minutes of angry but determined leafing through your instruction manual you found a way to either slow down the listing process or list only several lines at a time so that you could finally read that cherished program.

The problem described above is not really caused by the computer's listing speed, but by the process of standard scrolling. Let us assume that the text is being put on your CRT screen at a rate of 10 lines per second. If your display has a capability of 16 or 24 lines of text, this means that the particular line you were searching for was somewhere on your screen for 1.6 to 2.4 seconds. Now, surely your eye could find a line of 10 to 20 characters and read it in that time.

Not so! Since the lines of text move up the screen in a jumping motion from one discrete row to the next, the persistance of your eye causes each display row to become a blur of characters. If your CRT screen were 1000 lines long and you therefore had 100 seconds to find a specific line, you never could because the scrolling mechanism is built for your computer, not you.

Now, let us assume that we could take those 16 or 24 discrete text rows and give them around 500 possible jumps up the screen, moving only one character dot at a time. The result would be a display that moves smoothly up the screen. And since the maximum overlap occurring during the scroll would be one row of character dots, the eye persistance problem is eliminated.

The bottom line of the above comparison is one very visible difference. A scrolling rate of 10 lines per second appearing as a complete blur on a "normal" CRT display is completely readable on a linear scrolling CRT display. Such a display can be built using a CRT controller which allows dot by dot vertical positioning of the video display.

## The CRT Controller

The CRT 5027 Video Timer and Controller Chip used in this display system is a register programmable 40 pin COMPLAMOS® N channel MOS/LSI device. It contains the required internal logic functions to generate all timing signals for presenting and formatting an interlaced or non-interlaced video display. Since the controller chip provides horizontal sync, vertical

sync, and composite sync (including inter-sync pulse serrations) the CRT monitor used may be driven from composite video or individual sync signals. Display flexibility has been insured by the use of software programmable registers which control characters per data row, data rows per frame, and raster scans per data row and per frame. An additional register included as a data row counter facilitates text line scrolling associated with most CRT displays.

The key feature included in this CRT controller chip is the ability to vertically position the start of the display by scan line anywhere on the CRT screen. I assume this was originally provided by the designers at Standard Microsystems Corporation as a method of exact vertical centering of the display. However, with software which slowly raises the display on the screen and at the same time rachets the text on the screen upward, a rolling text effect is easily accomplished.

Before we examine the operation of the software let us look at the hardware configuration. Since there are no special hardware tricks required for the rolling display, the design of the system is straight forward. Interfacing the SMC 5027 to any microprocessor may be done using data given in the SMC 5027 data sheet and application note 1-1.

Figure 1 shows the structure of the system (including a few frills such as 32 possible character attributes and alternate character set capabilities). Existing character attributes consist of all combinations of flashing, half-intensity, inverted, and blanked characters and character oriented graphics.

The video clock is phase locked to the 60 Hz line frequency (or external sync) so no display "swimming" occurs. This key feature also allows the video clock to automatically adjust for display format changes such as changes in number of characters per row or character rows per frame.

The Peripheral Interface Device (PIA) has five output bits dedicated to supplying the attribute write code when any character is written into the video memory. This technique allows the microprocessor to output, for instance, the specific code for character flash, half-intensity to the PIA port and then any subsequent characters written into video memory will flash with half intensity.

The bus arbitration in the system allows the microprocessor to access the video Random Access Memory (RAM) anytime the CRT 5027 is not requiring new line data. Since the video line buffer is filled from video RAM only during the first of 14 scan lines for each character row, the video RAM is available for microprocessor use about 92% of the time. This low overhead permits rapid text changes and real-time graphics animation.

This versatile display system was interfaced to a 6502, as shown in figure 2, which makes the design information directly usable on the 6800 series processors.

Since the internal CRT 5027 registers are either read or write only, there is no need for the Read/Write line to be used. Some care should be taken, however, to write only to input registers or there may be a data bus contention problem. Other problems might occur if the microprocessor inadvertently outputs stray addresses which could strobe erroneous data into the write registers. This rather disasterous occurrence is prevented by the DATA STROBE pulse gated with the desired address location.

Address decoding is done with standard decoding techniques such as 74LS138 selectors. Since there are 16 internal CRT 5027 addresses, decoding should be down to the four least significant bits. This assures that the register bank appears in memory without redundancy.

The CRT controller, when interfaced properly to the microprocessor, provides extremely flexible display formatting through seven 8-bit control registers.

Two additional registers are provided to store the cursor character and data row addresses for generation of the video cursor signal. Controller commands such as reset, self-load, up scroll, and timing chain start are given by the microprocessor using the remaining seven addresses. A complete visualization of all addresses is given in figure 3.

All of the control registers must be preset with desired values before the system will function properly. This particular system uses 64 characters by 32 lines, (interlaced) and hence is programmed prior to use with 32 data rows per frame and 64 characters per data row (Registers 3 and 2, respectively). The horizontal line count register Ø is preset with 85 characters total horizontal width to allow an overscan margin.. Register 4 is preset with 525 scan lines per frame for full interlace. The other registers except for 5 and 6 may be preset to provide the most desirable display centering and size. These two registers are the secret to the linear scrolling effect.

To illustrate the mechanism behind the software rolling text mode, figure 4 gives the mechanism of the linear scroll cycle. Register 5 (VERTICAL DATA START) contains a $20_{10}$ at the beginning of a line scroll. This means that the first set of dots on the top line of text starts 20 scan lines down from the end of the vertical sync pulse (figure 4A). At this time the LAST DISPLAYED DATA ROW is set to $32_{10}$. Remember there are 32 total text lines so line 1 starts the display here.

When rolling starts, the VERTICAL DATA START register is slowly decremented one scan line at a time causing the entire display to rise slowly up the CRT screen. When the second character row overlays exactly where the first row started and VERTICAL DATA STARTS at the seventh scan line, decrementing stops. Then the microprocessor sets the LAST DISPLAYED DATA ROW register 6 to the beginning row 1 causing the top line (text line 1) to move to the bottom of the screen. The next instruction replaces the original $20_{10}$ in the VERTICAL DATA START register 5 resetting the display for the next roll cycle, except now character row 2 is the first line of the display. The microprocessor next replaces the text in the old memory location line 1 with that of line 33 and the sequence continues.

Although it may seem that the row 2 would appear twice during the last data row and vertical data start register change, the microprocessor changes these values within 15 microseconds so the action is "quicker than the eye can see" and the text appears to roll smoothly up the screen.

## The Software Driver

The software needed to produce the desired rolling text mode is extremely simple due to the CRT 5027 register capibilities. Figure 5 is a very simple 6502 assembly listing which produces either text rolling or scrolling depending on the entry point. Text roll speed may be changed by varying the time delay inserted by subroutine DELAY to accomodate any level of speed reading.

## References


1. Lewis, Don; SMC Applications note 1-1, Hauppauge, NY 11787

2. SMC Data Sheet - CRT 5027, Standard Microsystems Corporation, Hauppauge, NY 11787

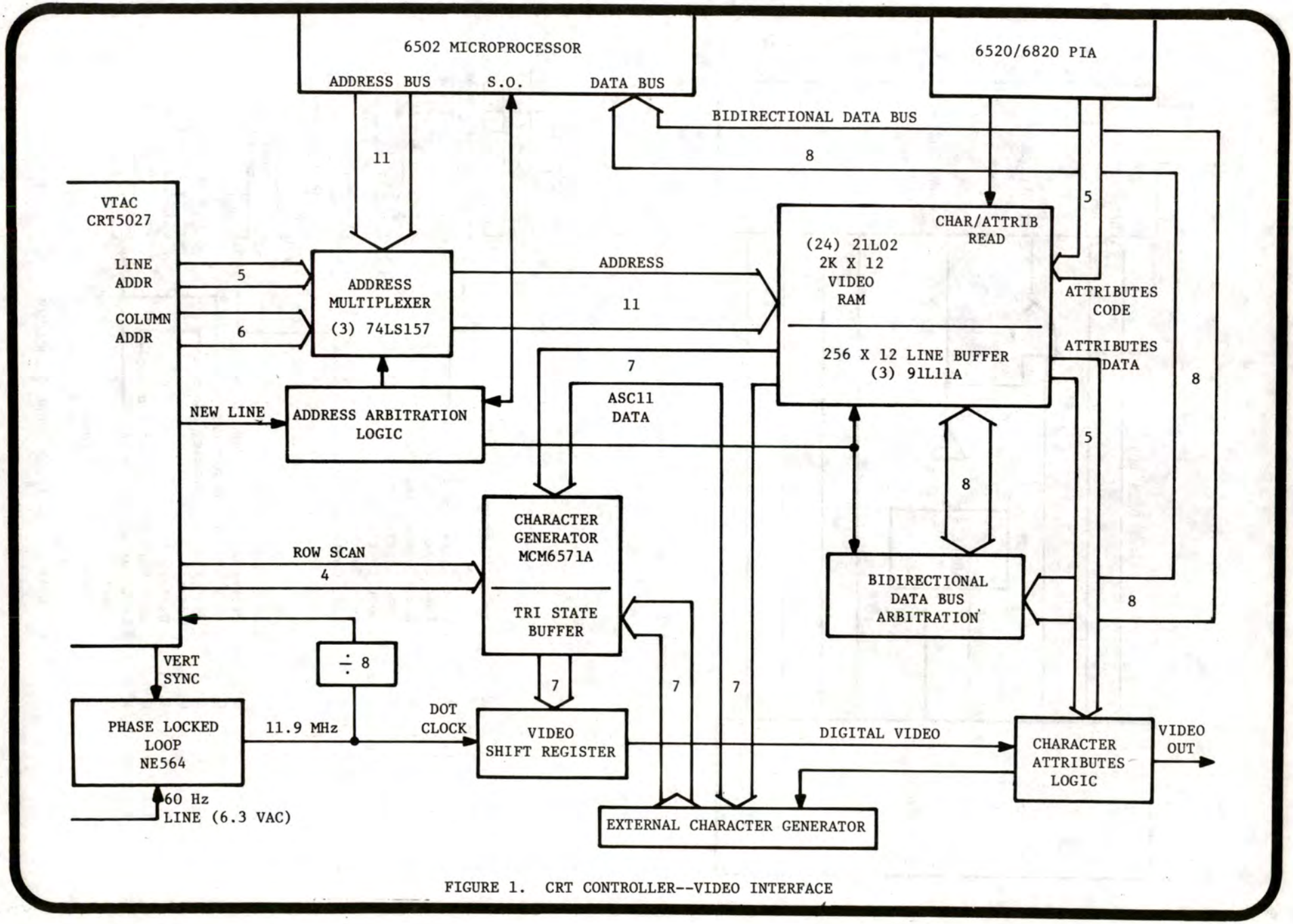



FIGURE 1. CRT CONTROLLER--VIDEO INTERFACE

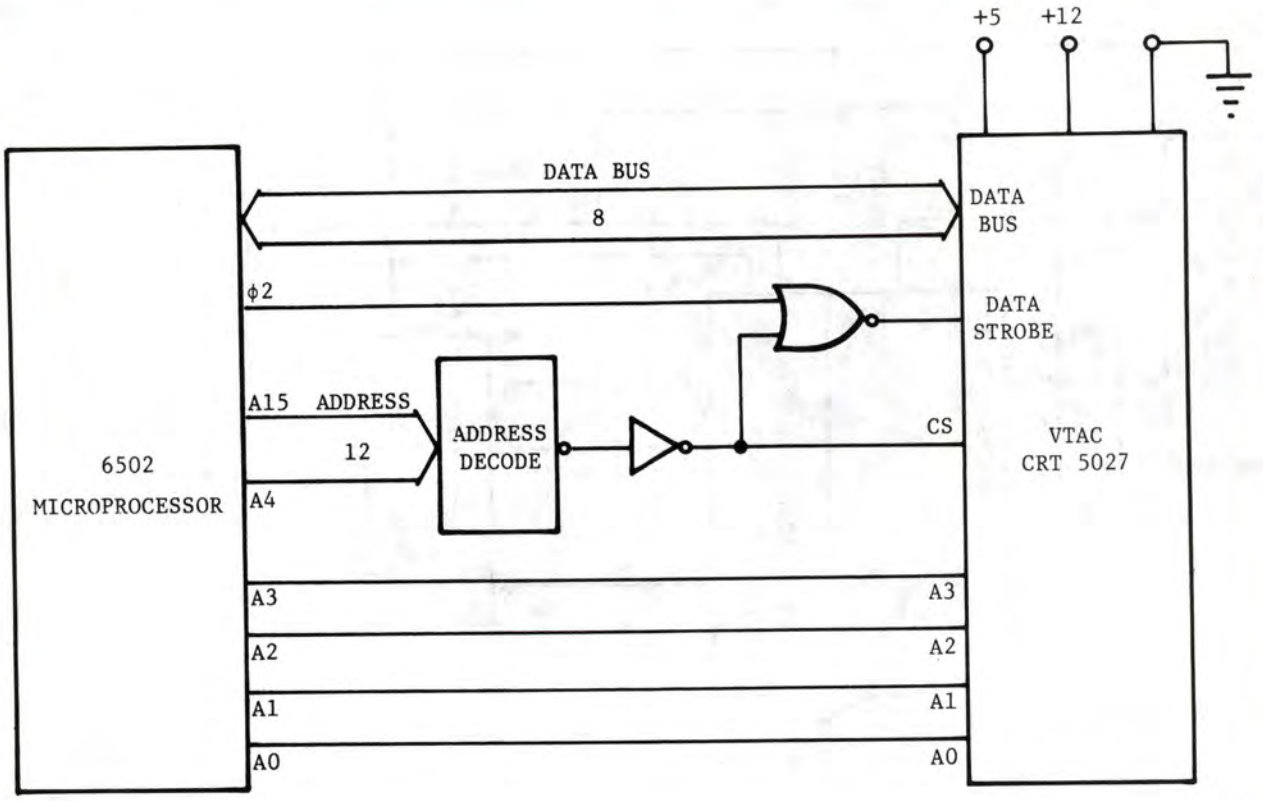



FIGURE 2. MICROPROCESSOR--CRT CONTROLLER INTERFACE

| A3 | A2 | A1 | A0 | Select/Command |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Load Control Register 0 |
| 0 | 0 | 0 | 1 | Load Control Register 1 |
| 0 | 0 | 1 | 0 | Load Control Register 2 |
| 0 | 0 | 1 | 1 | Load Control Register 3 |
| 0 | 1 | 0 | 0 | Load Control Register 4 |
| 0 | 1 | 0 | 1 | Load Control Register 5 |
| 0 | 1 | 1 | 0 | Load Control Register 6 |
| 0 | 1 | 1 | 1 | Processor Initiated Self Load |
| 1 | 0 | 0 | 0 | Read Cursor Row Address |
| 1 | 0 | 0 | 1 | Read Cursor Character Address |
| 1 | 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | 1 | Up Scroll |
| 1 | 1 | 0 | 0 | Load Cursor Character Address |
| 1 | 1 | 0 | 1 | Load Cursor Row Address |
| 1 | 1 | 1 | 0 | Start Timing Chain |
| 1 | 1 | 1 | 1 | Non-Processor Self Load |

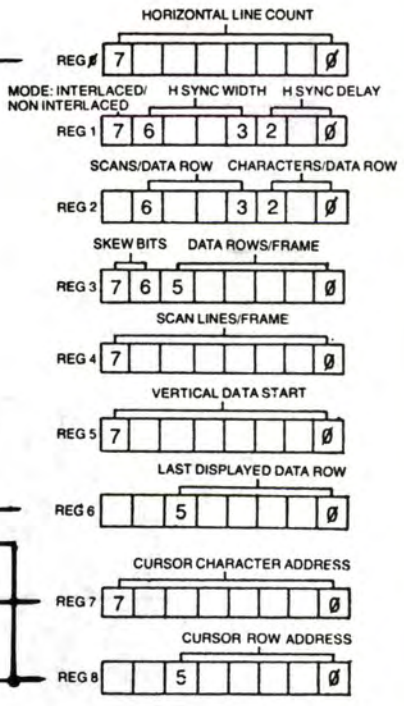



FIGURE 3. REGISTER ADDRESS LOCATIONS

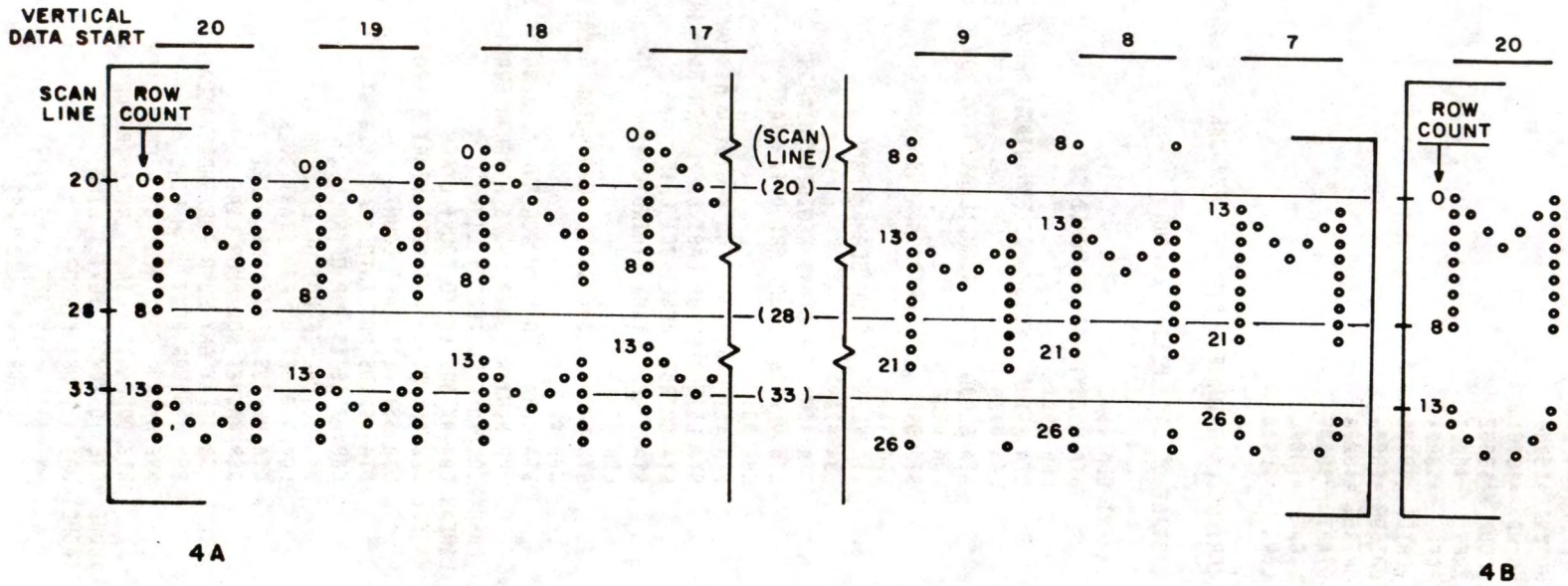



FIGURE 4. LINEAR SCROLL CYCLE ILLUSTRATION

```
APPLE

LINE#  LOC     CODE       LINE

0001  0219                      *=$2000
0005  2000                HOZCNT =$8000
0010  2000                HSYWD  =$8001
0015  2000                SCCDR  =$8002
0020  2000                DRFR   =$8003
0025  2000                SCFR   =$8004
0030  2000                VERTDS =$8005
0035  2000                LDISDR =$8006
0040  2000                CRTRST =$800A
0045  2000                START  =$800E
0050  2000                LCCA   =$800C
0055  2000                LCRA   =$800D
0056  2000                ;
0060  2000                ;CRT ROLL SOFTWARE COPYRIGHT 1979 JOHN P. CATER
0061  2000                ;
0065  2000                ; START OF CRT INITIALIZATION
0080  2000                ;
0090  2000  A9 4C         INICRT LDA #$4C
0095  2002  8D 00 80             STA HOZCNT  SETUP HORIZ CNT
0100  2005  A9 B1                LDA #$B1
0105  2007  8D 01 80             STA HSYWD   SETUP HOZ SYN WIDTH AND DELAY
0110  200A  A9 6B                LDA #$6B
0115  200C  8D 02 80             STA SCCDR SETUP SCAN ;CHAR/DATA ROW
0120  200F  A9 5F                LDA #$5F
0125  2011  8D 03 80             STA DRFR   SETUP DATA ROWS/FRAME
0130  2014  A9 06                LDA #$06
0135  2016  8D 04 80             STA SCFR   SETUP SCANS/FRAME
0140  2019  A9 14                LDA #20
0145  201B  85 50                STA $50 VERT START REGISTER MIRROR
0150  201D  8D 05 80             STA VERTDS SETUP VERT DATA START
0155  2020  A9 1F                LDA #31
0160  2022  85 51                STA $51 LAST DATA ROW REGISTER MIRROR
0165  2024  8D 06 80             STA LDISDR SETUP LAST DISPLAYED DATA ROW
0170  2027  8D 0A 80             STA CRTRST  RESET CRT CHIP
0175  202A  8D 0E 80             STA START  START TIMING CHAIN IN CRT
0180  202D  A9 01                LDA #1
0185  202F  8D 0C 80             STA LCCA   LOAD CURSOR CHARACTER ADDRESS
0190  2032  A9 00                LDA #0
0195  2034  8D 0D 80             STA LCRA   LOAD CURSOR ROW ADDRESS TO START OF SCREEN
0200  2037  4C ** **             JMP MONITR (GOTO USERS PROGRAM HERE)
0205  203A                ;COME HERE TO ROLL TEXT 1 LINE
0210  203A  A2 07         LINEAR LDX #7 SET UP TO COUNT LINES
0215  203C  C6 51         NXTROL DEC $51 DECREMENT VERT START MIRROR
0220  203E  A5 51                LDA $51 GET START COUNT
0225  2040  8D 05 80             STA VERTDS PUT INTO START REGISTER
0230  2043  C9 07                CMP #7 IS IT TOP OF ROLL?
0235  2045  D0 ** **             BNE DLYCAL IF NOT GO CALL DELAY
0240  2048  A9 14                LDA #20 SET UP FOR START OVER
0245  204A  8D 05 80             STA VERTDS NEXT LINE START
0250  204D  20 ** **             JSR RACHET NOW BUMP LINE
0255  2050  20 ** **      DLYCAL JSR DELAY ANY SUBR FOR SHORT DELAY
0260  2053  CA                   DEX NOW FOR NEXT SCAN LINE
0265  2054  D0 E6                BNE NXTROL CONTINUE FOR 1 CHARACTER
0270  2056  60                   RTS DONE WITH LINEAR ROLL ;RETURN
0275  2057                ;COME HERE TO SCROLL TEXT 1 LINE
0280  2057  A5 50         RACHET LDA $50 GET MIRROR
0285  2059  29 1F                AND #$1F MASK OFF USED BITS
0290  205B  8D 06 80             STA LDISDR PUT INTO LAST DISPLAYED DATA ROW
0295  205E  E6 50                INC $50 BUMP MIRROR TO NEXT LINE
0300  2060  60                   RTS RETURN TO CALLER
```

FIGURE 5. CRT SOFTWARE DRIVER

# AN OVERVIEW OF SERIAL COMMUNICATIONS IN MICROPROCESSOR SYSTEMS

Frank L. Toth, Manager, Microprocessor Marketing
American Microsystems Inc.
3800 Homestead Road
Santa Clara, California 95051
(408) 246-0330

## Introduction

There are two basic modes of Serial Communication that take place in microprocessor systems: Asynchronous (unclocked) and Synchronous (clocked). Within any given communication network there exists a protocol or set of rules that ensure that data transmitted is received as it was sent.

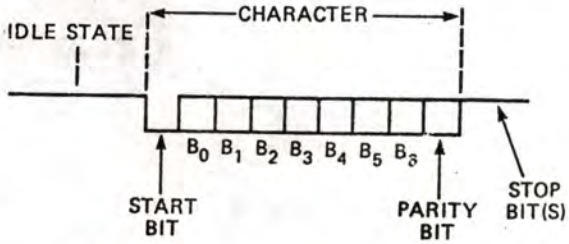
Synchronous communication is used in relatively complex computer and multiple microcomputer/terminal systems where a relatively high data rate is required. The data transmitted can be either character or bit oriented.

Asynchronous communication is used in relatively simple systems where low speed or irregular transmission rates are acceptable.

In a typical microprocessor system, the data from the microprocessor bus must be converted from a parallel to a serial format through some type of interface device. In simple asynchronous systems, this device is usually an asynchronous receiver and transmitter (UART). In more complex systems using Binary Synchronous Communications (Bi-Sync) a Synchronous Communications Adapter (SCI or UCIA) is utilized to convert parallel microprocessor data to a serial bit stream with a clock. In extremely complex Bit Oriented Protocol (BOP) systems having multipoints of reception/transmission, a complex device such as an Advanced Data Link Controller (ADLC) is used to transfer data from a computer to a remote location.

## Asynchronous Communications

In asynchronous communication networks, the lack of a clock forces the system to establish synchronization between the incoming data and the internal system on a bit-for-bit basis. The communication line is kept in a known idle condition when information is not being transmitted. To establish the required synchronization, each character is transmitted with a start and stop bit to indicate the beginning and ending of each character. An idle state is assumed between each character. (See Figure 1.)




TYPICAL ASYNCHRONOUS DATA STREAM

Figure 1

An additional character known as the parity is used for error checking on a character-by-character basis. Asynchronous character length varies, but typical codes range from 5 to 7 bits. If a code of 5 bits is used with a total of 3 control bits (1 start, 1 stop and 1 parity), a total of 8 bits are transmitted which is an overhead of 37.5%. Thus more than 30% of the total transmission time is used to transmit control bits.

A typical interface in an asynchronous transmission system between a microprocessor and a serial transmission line is a Universal Asynchronous Receiver/Transmitter (UART) which is shown in Figure 2. The data from the microprocessor is loaded into the data holding register and then transferred into the shift register where it is serially shifted out one bit at a time. Logic circuitry within the UART takes care of adding the start and stop bits and determining and adding the parity to the asynchronous transmission. The 8-bit parallel data is transmitted to the outside world one bit at a time.

The receiver operates in exactly the opposite manner. The serial data is shifted into the shift register one bit at a time and then is transferred into the data holding register and then to the microprocessor.

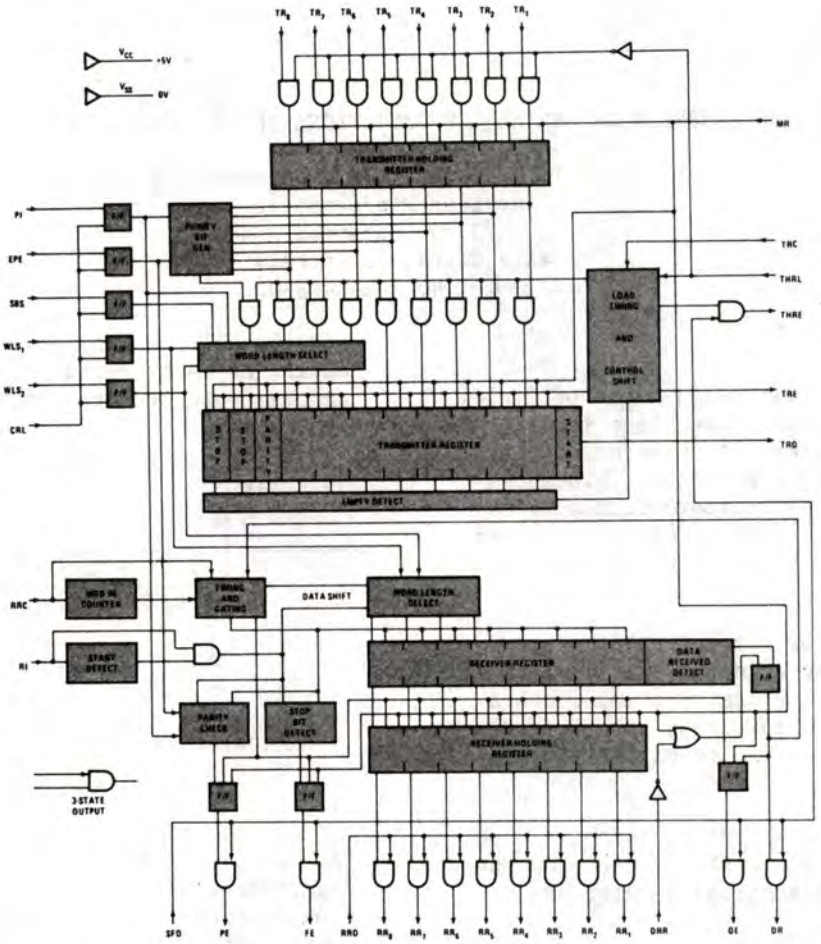



Figure 2: Typical UART Interface Circuit

As mentioned previously, asynchronous data communication is usually limited to low speed applications such as teletypes. There is nothing that precludes asynchronous communication at high data rates; however, where high speed is desired, a reduction in the total overhead of the system is also desired. The ultimate goal of the system is 100% utilization or zero overhead, that is, only the data bits are transmitted with no protocol or start/stop bits. By simply eliminating the overhead, the overall data transfer rate can be increased without increasing the band rate.

Synchronous Communication

One method of eliminating the extraneous bits required for transmission and at the same time insuring synchronization is through the use of synchronous or clocked communication (see Figure 3). Each of the bits of data transmission is synchronized to the clock and this data bit and clock stream is used to ensure synchronization between devices. The idle mode for a synchronous data link is determined by the individual system and may include either mark idling or the repetitive transmission of a special character. Information is usually sent in blocks or frames that can contain many characters. These blocks of information are typicaly preceded by one or more sync characters which are used to supply block or frame synchronization. The principle advantage of synchronous transmission is the enhanced efficiency of the communication channel through the elimination of the start and stop bits for each character. A typical synchronous data transmission is shown in Figure 2.

A uniform method of sending and receiving information is required in order to allow communications between various computers and terminals in a data link. Protocols (or sets of rules) are used to perform the following functions:

* establish and terminate the conversation between two stations,
* identify sender and receiver, and
* initialize stations.

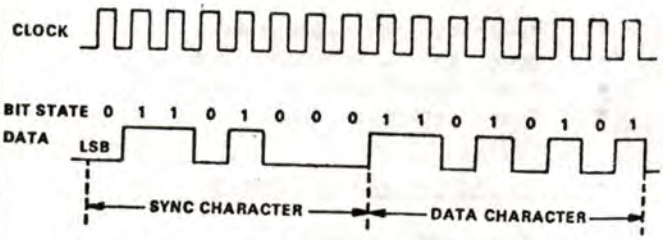



Figure 3: Synchronous Data Character Format

The procedure and function performed depend upon the protocol employed. Synchronous data protocols can either be Byte Control Protocols (BCPs) or Bit Oriented Protocols (BOPs). In the case of the BCP protocols, a byte wide set of control characters effects the synchronization and operation of the synchronous data transmission.

Before the advent of BOP, the industry standard was Bi-Sync (Figure 4) in which the information is transferred in the form of a block consisting of two or more sync characters, an address, an information field, control characters, and an error checking code. Control character bit patterns (SYN, DLE, ETX) are used to insure both synchronization and the proper operation of the communication network. A restriction of the information field is that a bit sequence that matches any of the block control characters cannot be allowed to occur undetected as this would be interpreted as a control rather than data. Exception: transparent Bi-Sync. In this mode, data that matches a control character can occur in the data stream if it is preceded by a DLE character.

| SYNC CHARACTERS | ADDRESS CHARACTERS | CONTROL CHARACTERS | INFORMATION FIELD | ERROR CHECKING |
|---|---|---|---|---|

Figure 4

Transmission of information in Bi-Sync is limited to half-duplex. The protocol requires an acknowledgement of the receipt of each block before another block can be transmitted. Once a communication channel is established and the transmitter sends one block, it stops and waits for an acknowledgement (ACK) signal before sending other blocks. The receiver acquiring the block checks for errors and then sends an ACK control character to the transmitter indicating that the block is correct, or a NACK control character to indicate an error. Data is thus transmitted in one direction at a time.

Character synchronization in synchronous transmission is accomplished by recognizing the SYN (synchronization) character. A typical IC circuit that does this type of synchronization is shown in Figure 5. This circuit is capable of both asynchronous and synchronous data communication depending on the programmed operating mode. The device senses the SYN characters and then "locks" the receive logic to recognize both the SYN character(s) and the beginning and ending of all subsequent data characters. This circuit also ensures character synchronization throughout the desired message by inserting SYN characters where needed to ensure synchronization when data is not being transmitted (idle state). If for some reason the SYN character is not received by the UCIA, the device initiates a search mode to re-establish synchronization. Character synchronization relies on the receipt of an entire sync word rather than on the timing of individual bits.

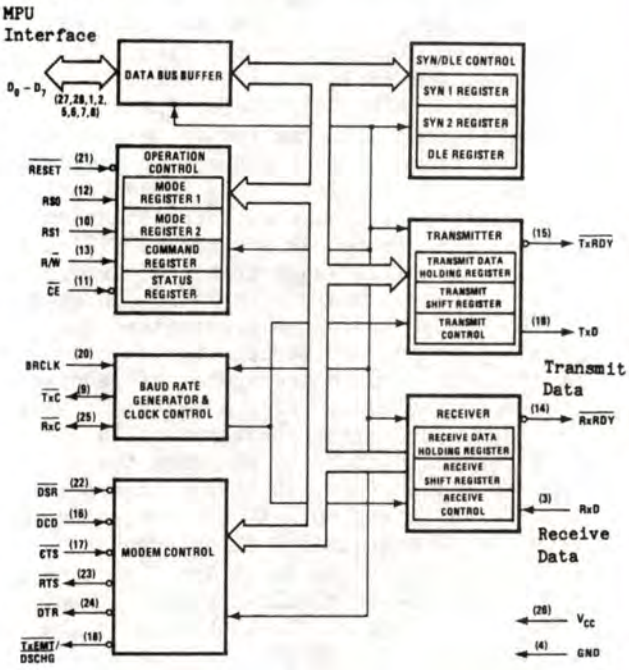



Figure 5: Typical Syncronous/Asynchronous Microprocessor Interface Circuit

## Bit Oriented Protocols

There are fundamental problems with Bi-Sync. First, control characters cannot be used as data codes, and bit patterns that can occupy these control characters cannot be transmitted as data. Second, information must be acknowledged before transmission can continue. To solve these two problems and further enhance communication network capabilities, Bit Oriented Protocols (BOPs) were conceived. The BOP

communications network has the same high speed synchronous serial applications as the character oriented protocols, but avoids the Bi-Sync, half-duplex requirement. In addition, blocks received do not require acknowledgement each time they are sent. As a result, full duplex operation is allowable in BOP networks. Because of this ability to operate in a full-duplex mode coupled with fewer inherent handshake sequences, BOP networks can communicate at more than twice the effective data rate of Bi-Sync networks.

All communications in a BOP system are in the form of frames of a uniform format. The frame (see Figure 6) is composed of a number of fields, each with a definite location and precise meaning. The beginning/ending frame character, abort, and go-active-on-poll are the only three characters that control the data flow on the communication link. The zero bit insertion technique makes certain that these codes never occur inside a frame, therefore allowing complete code transparency. As a result, the code bookkeeping required in Bi-Sync is simplified. In addition, the implied acknowledgement technique used in BOP enables frame acknowledgement information to be included within a frame primarily transferring data to the remote station. This procedure is accomplished by assigning identification numbers (called sequence numbers) N(R) and N(S), to received and transmitted frames. These numbers contain information in regard to the number of frames transmitted and received by the individual station. A station can compare the number of received frames to the number of transmitted frames, and take appropriate recovery action should a discrepancy exist, by checking these numbers. Therefore, it is not imperative to operate in a stop-and-wait situation and send frames containing acknowledgement information only (as in Bi-Sync) permitting BOP to operate in a full-duplex (two-way simultaneous) mode. This ability is the primary advantage of BOP. Bit Oriented Protocols are thus capable of achieving more than twice the throughput rate of Bi-Sync while employing the same type of communications facility.

The factors involved in this higher throughput rate include the following:

* fewer handshake sequences,
* fewer acknowledgement frames,
* fewer synchronizing characters, and
* fewer interblock gaps.

## Typical Bit Oriented Protocols

There are currently three major BOPs in usage: Advanced Data Communication Control Procedure (ADCCP), High Level Data Link Control (HDLC), and the Synchronous Data Link Control (SDLC). All of these protocols have similar features, but the most widely used is the IBM SDLC. A block diagram of a typical IC that can handle these protocols is shown in Figure 7.

With the SDLC protocol, information is transmitted in frames, and a frame is made up of a number of fields. Since this is a Bit Oriented Protocol, each bit in each field has a specific meaning. In addition, there are two types of stations: primary and secondary.

The SDLC protocol, as are most BOP type protocols, is code independent. The only requirement is that the data being transmitted be contained in the information field of each frame. This concept allows for full duplex operation since the number of frames sent and received is recorded in two control fields within the information frame. In this way, secondary stations can transmit back to primary stations the number of frames that have been received. If this number does not match the number of frames sent, a request for retransmission can occur without confirmation of receipt of each frame.

The IBM SDLC protocol uses a beginning and ending flag for each frame. Zero bit insertion is used to insure that the code is completely transparent. The flag is the only protocol-derived control character. Once the opening flag is sent, the protocol requires that for every five 1's a zero is transmitted. The receiver strips out the zeros automatically.

| OPENING FLAG | ADDRESS FIELD | CONTROL FIELD | INFORMATION FIELD | FRAME CHECK SEQUENCE | CLOSING FLAG |
|---|---|---|---|---|---|
| 01111110 | N OCTETS | 8 OR 16 BITS | VARIABLE | 16 BITS | 01111110 |

TYPICAL BOP FRAME

Figure 6

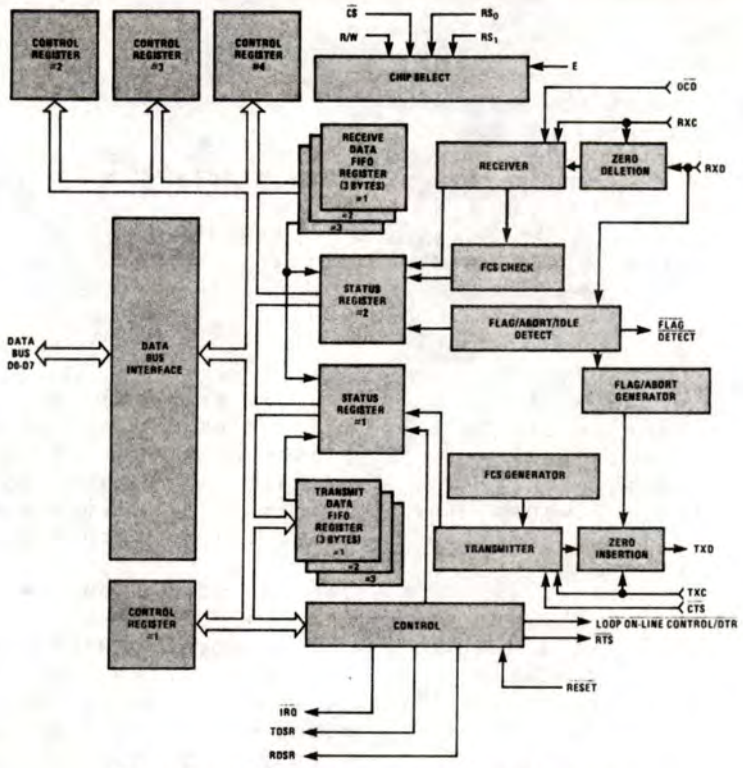



Figure 7: Typical IC Designed for Bit Oriented Protocols

The IC shown in Figure 7 can handle the SDLC, HDLC, and ADCCP protocols with the ability to do automatic flag detection, synchronization, zero bit insertion/deletion, and an automatic frame check sequence generation and check.

The frame check sequence looks at every bit (except the flags) and operates on each bit with a fixed algorithm which then generates and transmits a frame check sequence. This frame check sequence is compared to the received frame check sequence. If a match is found, the transmission/reception is verified. If no match is found, a retransmission is requested.

Transmission ceases with the initiation of an abort sequence. Also, the transmitter can go into an idle state and the IC can be placed into a loop-back self-test mode for self test.

Summary

There are two types of serial data communication: Asynchronous and Synchronous. Synchronous communication can take place in two main protocols: Byte Control Protocols (BCPs) and Bit Oriented Protocols (BOPs). Bit Oriented Protocols are divided into three major types: ADCCP, SDLC and HDLC.

The most widely used BOP is the IBM SDLC which consists of a frame with several control, data and error check frames.

Asynchronous data transmission can take place using a UART as an MPU interface device.

Synchronous data interface and transmission uses either a UCIA (Universal Communications Interface Adapter) for BCPs or an ADLC (Advanced Data Link Controller) for BOPs.

# Seeing Motion with the Mind's Eye

Sam Hersh and Al Ahumada
Aero/Astro, Stanford U., Stanford, CA (415) 497-1526

## Abstract

An extraordinary ability of the mind to see motion where none exists is the basis for animated visual displays. For example, if two neighboring figures are successively flashed, the figures appear to move smoothly from one position to the other when the time interval between flashes is between 30 and 200 msec. Many related phenomena can be demonstrated and investigated using an inexpensive video processor instead of standard electromechanical instruments which are less versatile.

## Introduction

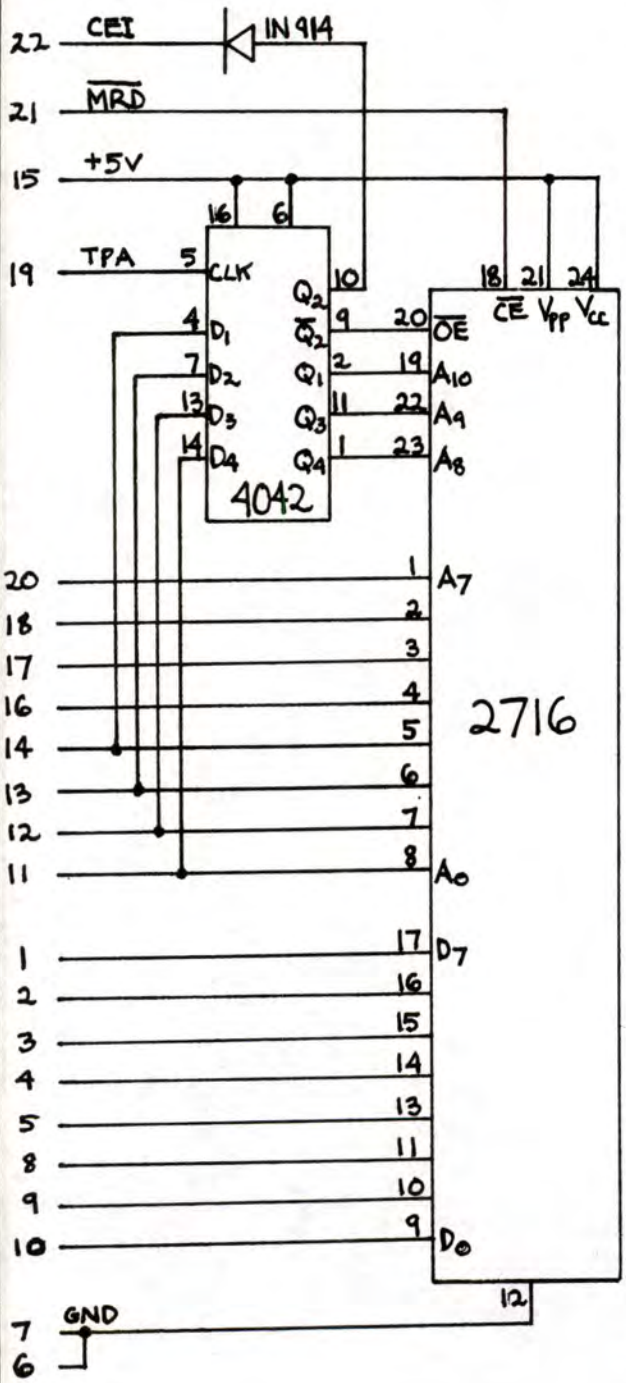
Motion illusions have fascinated the public at least since 1867 when Milton Bradley patented an animation toy called the Zoetrope or "Wheel of Life." One of the earliest motion picture machines was made by Plateau, a Belgian vision scientist, in 1833. His device, the stroboscope, consists of a sequence of still pictures printed on a disk which are viewed as they spin behind a series of slits. He gave a prototype to a countryman, Quetelet, the founder of statistics, who eventually gave it to Michael Faraday. Shortly thereafter the stroboscope and related animation devices were widely sold as parlour toys for the children of Victorian intellectuals. In the later half of the nineteenth century, the German scientists Helmholtz, Mach, Wundt and Exner were among the first to make precise measurements of apparent motion. Exner's method was to present two separate successive electric sparks and ask observers to judge the order of presentation. His device is a forerunner of one of the most widely used instruments in vision research, the tachistoscope. The t-scope is an instrument for controlling the exposure of visual displays. Typically, fast light sources are used to illuminate one of two fields containing drawings or photographs that can be replaced manually. An enormous body of data in vision science has been collected using this instrument.

## The Phi Phenomenon

A major breakthrough occurred at the turn of the century. Legend has it that Max Wertheimer, a vacationing young philosopher travelling from Vienna to Berlin, purchased a toy stroboscope of the type constructed by Plateau. Wertheimer disembarked prematurely at the Frankfurt Station in order to conduct an experiment in the lab of a colleague, Professor Schumann. Schumann, an expert on visual illusions, was pioneering a new research method known as experimental phenomenology. Wertheimer borrowed a t-scope, the services of two recently graduated research assistants, Wolfgang Kohler and Kurt Koffka, and a bit of lab space in which to systematically study a motion effect that he called the phi phenomenon. Wertheimer remained in Frankfurt for two years and then published his famous 1912 paper on motion perception which signalled the birth of gestalt psychology.

In it, he shows that varying the time interval between separately exposed lines produces three distinct phenomenological reports. If the interval is greater than 200 msec, the lines appear successively. For intervals less than 30 msec, they appear simultaneously, with the optimal interval for movement about 60 msec.

Wertheimer proposed that apparent motion is due to a neural short circuit in the brain. There is still no satisfactory explanation for the scores of phi phenomena that have since been described.




A Smart Tachistoscope

During 1977-78, we developed hardware and software to do pschological tests using an RCA 1802 Video Interface Processor (VIP) and soon became interested in having several less expensive systems which could be used by students for individual projects on visual perception. We decided to use the VIP as a development system for the RCA Studio II Home TV Programmer. The Studio II is an 1802 based CMOS video game computer with a 1 kbyte residient ROM interpreter (0000-3FFF), RAM workspace (0800-08FF) and display RAM (0900-09FF) which accepts 1 Kbyte ROM cartridges (0400-07FF). It has two decimal keypads for input and built in RF modulated video, beeper and LED for output.

At the time, Radio Shack was selling RCA's remaining inventory of Studio II's for $60 each. We bought as many as we could find (5) and constructed an EPROM board which has only 2 IC's (a 2716 and a 4042) and substitutes for RCA's ROM game cartridges. A simple modification to the Studio II must be made to provide TPA at pin 19 of the plug in connector, namely, cut the trace to pin 2 of the onboard 4001 and jumper pins 1 and 2. This mod is compatible with RCA's ROM cartridges. We have been able to run the EPROM board using stock Studio II power supplies although some onboard reglators have failed without this additional load.

A program was developed to facilitate demonstrating and studying phi phenomena. It converts the Studio II to a programmable t-scope with four user defined shapes any or all of which may appear in one of two alternating display fields. Shapes may be moved horizontally or vertically during field alternation Both the duration of display fields and the interval between fields are independently variable.

Any video game system that uses a standard microprocessor is a candidate for an inexpensive prototype. Of course, development is much easier if the manufacturer is willing to cooperate, as was RCA, by providing information. A reasonable development system might consist of a microcomputer with the same processor, a printer and mass external storage. A resident EPROMmer, like the one RCA makes for the VIP, would significantly reduce development time compared with using one on another system. Ideally, a battery powered RAM board with write protection would be a convenient vehicle for transferring programs from the development system to the prototype for debugging. Another obvious development target is the very popular ATARI Video Game System with development on one of the 6502 based home computers, like the ATARI 400/800 or APPLE.

Phi Phenonomena will be demonstrated.

References
---


1. Kolers, P. Aspects of motion perception.
2. Hersh, S. & Ahumada, A. Psychological tests with video games. Proceedings of the Second West Coast Computer Faire, 1978.
3. Anon.,"A cheap graphics computer," VIPER, 1(6), 1978. ARESCO, P.O. Box 1142, Columbia, MD 21044. (ARESCO publishes excellent monthly newsletters on the VIP and Studio II)

# BREAKING INTO WRITING

# FOR THE MICROCOMPUTER FIELD

Sharon Rosa
Nicholas Rosa Associates
1901 S. Bascom Avenue, Suite 337
Campbell, California 95008

Writing "about computers" is technical writing. This is true even if you intend to write for popular magazines or newspapers rather than for a computer company. The rock-bottom requirements are the same.

Technical writing is one of the easiest fields to get into, and one of the hardest to stay in. Why easy to get into? Because there is a crying need in the microcomputer industry. Every week another company wakes up to its need for good technical communicators. Why hard to stay in? Because it is demanding, exacting; because the results of your work will be out there for all the world to see. (And for your boss to see.) If you're a dud, it will show. But if you are competent, conscientious, professional--- then Silicon Gulch can be a technical writer's El Dorado.

## What Does It Take?

First, you must be literate. You must at least know what good writing is. You should have been good at writing essays and stories for your high school paper, at minimum. This implies that you like to write. If you are a programmer, you like to document your work. (That makes you a rare and precious bird indeed.) A technical writer needs an organized mind. Are you good at pulling order out of chaos? The technical writer needs judgment. Can you differentiate between the way you may have learned some process, by yourself, and the best way to teach it? A good technical writer is an intuitive psychologist (like any good writer). Can you identify and ask those nitty-gritty questions that make the difference between a mediocre manual and a really good, really useful one? (Your writing must anticipate all the reader's stumbling blocks.) Finally, you must have discipline. It isn't always easy to make yourself get to it and write. But no "waiting for inspiration" is allowed. Technical writing is no field for dilettantes.

Those qualities---literacy, organization, question-asking clairvoyance, and iron discipline---are to an extent inherent personality traits. You can develop one or the other or two of them, but it's a lot easier if they come naturally to you.

## The Writer's Background

The backgrounds of technical writers are incredibly varied. I have yet to meet a technical writer who had any notion of being one when he was in school. I think of technical writing as "the default profession." In the computer world, at least, the profession seems to be manned by refugees from teaching or journalism, plus a nice sprinkling of degree-holders in French, philosophy, and medieval history. Along, of course, with renegade engineers, technicians, programmers, and systems analysts. So, if you feel that you can become a technical writer, you probably can.

To evaluate your potential, check your experience and education against this list:

1. General writing experience (even unpublished---do you write?)
2. Courses or experience in technical writing (any industry)
3. Courses or experience in computer programming (even self-taught)
4. Any training in electronics (even self-taught or hobby)
5. Any training in computers (even self-taught or hobby)
6. Work experience in computers or electronics
7. Teaching experience

Numbers 1 and 3 or 5 or 6 are most important. You have to know what you are writing about. Any genuine teacher of any kind of writing implores you to write only about the things you know about. So, if you haven't already acquired a microcomputer, haven't at least joined a computer club, aren't also (and not only) reading everything you can find about microcomputers, get cracking. If you've already got item 3, 4, 5 or 6, start hitting 1 and 2. Hard.

You don't need an A-plus on every item in that list. But you need some meaningful combination.

Technical writing seems to attract two kinds of people: the "technical" types and the "writer" types. If you would forever be content to be a low-paid hack, be "purely" one or the other. If you are reaching for success, you must be both. Having built (or even designed) hardware, or having written programs, is not enough. Being a fine "creative" writer is not enough. You must have technical understanding and you must have sharply honed writing skill.

Getting Started

It is not likely that your talent or background alone will get you into a writing job in the microcomputer industry. (But yes, technical writing in other fields would certainly help.) You must have experience and SAMPLES.

"But I can't get experience. They won't give me a job." You can get around that. You can bootstrap yourself. The samples are your boothook.

Everything you write or have written, published or not, is a sample. If it's good, use it. No matter what it's about. If you participated in writing something, or helped edit it, get a copy. Save it. Cherish it always. Never let go of originals. Don't lend them out, don't leave them with anybody---you may never see them again. Get the best quality machine copies to send out, lend out, or leave with prospective employers.

If you already know a lot about computers, write a few short dummy articles on what you know best. Of course, try to get them published. But bring good Xerox copies of your manuscripts around to job interviews. (Don't worry if your attempts at publication get you rejection slips at first. Review the pieces, rewrite them, go from there with the improved versions.)

If you are confident about being a good writer but still don't know much about computers, write an appropriate piece or two about whatever you do know. That shows you're learning, you're interested, you're putting in effort on learning and writing. Write about any problems you or others might be having in learning about computers. You can find something about computers to write about. Gather up any other good writing you have done on any subject, even clippings from your college paper or letters-to-the-editor of your local newspaper: whatever shows that you can write.

Have you run across a bad computer manual? Think you could do a better job? Pick out a section and rewrite it. To job interviews, bring the original ("before") as well as your rewrite ("after") for comparison. Memorize your notes on just why you thought the original was bad; be prepared to show why yours is better. Again, have machine copies to leave with the interviewing executive. DON'T PART WITH YOUR ORIGINALS.

Join a computer club. You can learn fast there, even by osmosis. If possible become the club's newsletter editor. At any rate write all you can for the newsletter. The editor should be hungry for material. Write for the newsletters of any other kind of club you belong to if you are not doing so already. It's all writing experience, and it gets you samples. BUILD YOUR PORTFOLIO.

Joining the club gets you exposed to computers, gets you exposed to computer-wise people, exposed to ideas, exposed to software, teaches you a lot in a pretty good hurry, and makes you contacts.

Buy a computer. Work with it, play with it. "But I can't afford it just now." Well, for less money, you can buy good self-teaching kits, that show you how components work, teach you the principles of programming, and ultimately endow you with a very small. limited, rudimentary but real computer.

Are you a teacher? Write programmed learning modules and save them. Any subject ---though math and science might have an edge. There was a time when computer companies did not want to hire ex-teachers to write manuals. Don't ask me why. But now many do. They are beginning to appreciate the contributions that a person with teaching experience can make. Teachers know how to present materials in an organized manner. Teachers are good at estimating the reader's technical level and pinpointing where the reader is likely to have trouble. If you are a teacher, point these facts out to prospective employers.

Another thing a teacher can do to prepare for a tech-writing job is to use a

microcomputer in the classroom. You can get hold of teaching software for use with children. And for heaven's sake, as long as you have this access to a computer, learn BASIC on it. Then write a program to teach the kids about muskrats or math and write a small manual to go with it. Prospective employers will love it!

(That will give you something to publish in a teachers' magazine or computer magazine. You'll have another sample.)

Of course you are going to subscribe to computer magazines (or regularly pick them up at some computer dealer's). When you have a feel for the kinds of material each magazine publishes, write and submit articles. If any of them scores, you have a glamorous sample.

Wherever you go, you are going to carry that ever-growing portfolio of your best samples (and machine copies to part with).

Presentation

The most crucial part of your battle, in the job-interview office or in the editorial office, is presentation. Appearance counts heavily. You need the best quality typing on good heavy bond paper. (Do not use "script" or other gimmicky typefaces.) Everything should be absolutely free of spelling or typing errors, preferably with no visible corrections. It should be professionally formatted, so have at least one short manuscript done by a professional typist. Proofread the typist's output and get corrections made. Use that one as a model for anything you thereafter type yourself (and use the best typewriter you can, preferably with a carbon ribbon). If you can get it prepared on a word processor, all the better. A good professional typist these days charges $8.00 to $10.00 per hour but will do up to 10 pages an hour. Word processing might cost $12.50 or so. A reasonable investment, for a "money's no object" purpose.

Security versus Freedom

There are three ways to work as a technical writer. The first is "having a job." Suppose we refer to that as captive employment. Then there is "job shopping." You sign a contract with a technical skills agency and they keep locating spot jobs for you. (A "spot job" might last for months.) The third way is free-lancing: you're on your own, out in the blue.

Suppose we compare their advantages and disadvantages:

A. Captive Employment

ADVANTAGES: 1. Job security.
2. Absolutely regular paycheck.
3. Minor emotional strain.
4. Benefits package: vacations, paid holidays, sick pay, etc.
5. Getting paid while you learn.
6. Companionship of peers.

DISADVANTAGES: 1. Hassles with people (office politics).
2. Less chance for really big money.
3. Lack of freedom.
4. Getting buried, anonymity.
5. Boredom.

B. Job Shopping

ADVANTAGES: 1. Excellent hourly pay rates.
2. Little need to market yourself--the job shop does it for you.
3. Frequent new experience, new company contacts.
4. Regular pay, while you are working (usually weekly).
5. Withholding tax, unemployment insurance, etc., taken care of.
6. FREEDOM! You may quit a job that doesn't suit you, with no disgrace.
7. Variety of experience.
8. Getting "in" with a variety of companies.

DISADVANTAGES: 1. You're still commuting to work, "punching a clock."
2. You're temporary.
3. Free-lancing money could be even better.
4. Difficult to establish a reputation, since the client tends to associate your good job with the job shop rather than with you.
5. Contractual restrictions keep you from going captive or free-lancing with a "good" company they sent you to, until after a certain time period has passed.

C. Free-Lancing

ADVANTAGES: 1. Chance at top money.
2. Freedom to space jobs as you like.
3. Usually working in your own home or office.
4. Flexibility in hours.
5. Broad range of projects.
6. The great feeling of "doing it for yourself."

DISADVANTAGES: 1. High stress---

when you work, you work really hard.
2. Frequent jobless periods---more stress.
3. Loneliness of working alone.
4. Need for managing money for taxes and slack times.
5. No benefits package.
6. Constant new and unfamiliar situations.
7. Constant need to market yourself.
8. Late pay---up to six weeks (or more!) late.

Probably everybody dreams of being a freelancer. I can warn you it's scary. It's not for everybody. Most tech writers who are professionally but not temperamentally suited for free-lancing opt for job shopping. For either, you've got to qualify. It's hard, if not impossible, to get a job shop interested in you as a rank beginner. It's suicide to try free-lancing as a beginner, unless you have a captive job (any kind of captive job) and can try writing as a moonlighter. But when would you get a chance to sell your moonlight technical-writing service? (Maybe you could have a night captive job and sunlight your technical writing.) And you can moonlight your magazine and newspaper article writing, of course. But don't forget, to free lance you have to be a reliable self-starter.

How Much Can I Earn?

As an entry-level captive employee, you might command $8.00 to $10.00 per hour. That's around $18,000 a year plus a benefits package. Because of inflation the rates keep changing; no figure I can give you is firm. When you have risen to the senior tech-writer level, you might realize $25,000 a year or so. You might get to be "senior" in a rather few years if you've really got it.

A job-shopper can expect between ten and fifteen dollars an hour. It depends on the particular job and company. The job shop will get you the best deal it can. But it has to charge the company an override on you, so that puts a ceiling on what they can ask. Once you have had years of experience and they know you are tops, they will bite companies for $20.00 or $25.00 for you. But you may not be working all the time: for all of their clients, you are temporary.

Don't begrudge the job shop its override. The company pays, not you. You are employed by the job shop; it bears all overhead, including your unemployment insurance, etc. They, not you, did the marketing to get you the initial client interview. You might not be working at all without them. They got you a better rate than you could get captive-hire. And you are making great contacts while you work with them. If you want to go captive again some time, or want to try free-lancing, those contacts could pay off.

The free-lancer tries to be his or her own job shop. Theoretically you should be able, in effect, to collect the job shop override in your hourly fee. (But remember, you are competing with the job shop.) It looks attractive.

Still, you are likely to start out free-lancing at $12.00 to $15.00 an hour. Once you are seasoned you might demand $20.00 or $25.00 or even $30.00---"whatever the traffic will bear." Broadly, you can expect to make between $10,000 and $50,000 per year. I would guess that most freelance writers in computers or electronics earn between $20,000 and $30,000 a year. It's a matter of how much work you get, or how much you like to work, and how much you can charge. You have to charge high---but remember, clients don't like to pay high. You risk pricing yourself out.

Even $15.00 looks good, doesn't it? Why, that should come to $31,200 per year! That is better than 95 percent of the captive technical writers get, by far. But that's if you work 52 weeks' worth of work every year. Pick any rate you think you could get, multiply it by 40 hours and then 52 weeks, and then start subtracting:

| | |
|---|---|
| Planned vacation (unpaid) | - 2 weeks |
| Unplanned vacation (no work) | - 9 weeks |
| Holidays | - 8 days |
| Illness | - 1 week |
| Marketing, 5 hrs/wk | - 6 weeks |
| Making wrong deals, doing too much for too little | - 3 weeks |
| Taking the dog to the vet, going to the dentist, taking your mother out to lunch | - 2 weeks |

As you can see, this makes a difference. When you subtract nonproductive time from your "potential" you find you are working only about 35 weeks a year, at best. Each week, you will bill only about 35 hours because of marketing time (in my calculations I subtract 5 hours per week 52 weeks per year) though marketing "costs" only during those weeks you are working.

When you are not working you had better market---but then it's "free."

When all is considered, your best course is probably to start out as a captive employee and get all the on-the-job experience you can. (And get paid while you're getting it.) For that, you are going to have to market yourself, get your foot in the door.

Marketing Yourself

I can't turn this into a general "how to get a job" course but can offer specific tips to the would-be technical writer.

Of course you will have a good resume. Have it professionally typed, and all copies should be made through top-quality printing or Xeroxing, on good paper. If you haven't any tech writing experience to show, list your other work experience plus all writing experience, not matter how Mickey Mouse. (As time goes by and you gain experience, you may start dropping the Mickey Mouse and the other work.) List all computer experience, whether big-computer or microcomputer, from work, schooling, hobby, or club. List courses you have taken on computers, electronics, technical or creative writing. Ask knowledgeable people whom you have associated with in connection with writing, computers, or both for permission to list them as references.

Don't forget to bring your portfolio to job interviews.

I can't tell you how to conduct yourself at the interview. You undoubtedly know the rules and have followed them before: dress neatly and conservatively, exude quiet confidence, show specific interest in the company, etc., etc., etc.

I can suggest ways to get yourself to the point of interview:

1. Join one or more computer clubs. You will learn much and meet people. You'll be surprised at whom you may meet at meetings---even executives from inside your target companies. Your eventual strategy will be to get around the guardian dragons of the Personnel Office ("Please fill out this application form so we can file it and forget it.") Some day you may inspire somebody inside the company to tell the executive who is going to hire a new tech writer about you and your talents. Most good jobs actually come to you: somebody inside the company reaches out and pulls you in. But he or she has got to know about you. (Meanwhile, keep charging those dragons at the Personnel Office---get whatever job you can.)

2. Go to computer shows. The executives in a Computer Faire booth are harried---they aren't going to give you a job interview on the spot. But you can, at some tactful moment, ask one for his or her card---even exchange your own card for it. And of course, collect their literature---get familiar with your target companies. How do you choose targets? Go home and read everybody's literature.

3. Send your resume with a brief cover letter to any contact you make, explaining that you would like to work as a technical writer for his or her company. Remind the contact that you met briefly at a computer show (name the show, the city it was in, the date). Your contact will usually refer your letter to the "right" person, if he or she is not that person. Follow up your letter with a phone call. See if you can wangle an "advice" interview.

4. Hand out your card to everyone remotely connected with computers. Cards are your cheapest advertising. "Betty Allen. Technical Writer."

5. Of course answer the classified ads. Go after any job you think you can land.

6. Spend several hours each week on marketing yourself. Get those letters and resumes out, those ads answered. The jobs are not going to come to you---not if you just sit there.

7. If you already have a job of any kind, hang onto it while you look for a tech writing job. You are more attractive to prospective employers if you already have a job. What if you don't like your first writing job once you have landed it? Stick with it while you mount a secret campaign (back to the letters, resumes, classified ads) to get a better one. Remember, it's experience.

To reiterate, you can't be a writer in this field unless you get closely acquainted with computers somehow. If you already "know all about" computers you have to polish and polish your writing, to make sure you can communicate what you know. Either way, you have to get into the habit of writing from the reader's point of view, serving his or her needs, not your ego. Your aim is to be a professional technical communicator.

Good luck. If you can write, the industry needs you. So much of the existing documentation is still so bad. You could be a great marketing asset to your company if you can produce manuals and documentation that make life easy for the end-user.

About the Author

Sharon Rosa is a partner in a firm of technical writing consultants. She is a software documentation specialist.

# IS ELECTRONIC TECHNOLOGY MAKING MANKIND AN ENDANGERED SPECIES?
## (or: Carbon Chemistry Chauvinist? - You bet!)

By: Don Perry Dunlap, President, Entecon Corporation, 1208 Apollo Way, Ste 502
Sunnyvale, California 94086 Phone (408) 245-9272

Introduction

We in the "Intelligent Machines" industry are unavoidably among the shapers of not only Tomorrow's world, but probably what will become of mankind itself. Perhaps that is important enough for us to pause a moment in our efforts at creating the means of achieving the future and contemplate what we really want the ultimate fruits of our labors to be. Presented here are a perspective and a few bases for formulating some choices that may become increasingly significant in guiding the directions of our work and innovations in the future.

---

In 1930, the renowned British Astronomer, Sir James Jeans wrote: "We are living at the very beginning of time. We have come into being in the fresh glory of the dawn, and a day of almost unthinkable length stretches before us with unimaginable opportunities for accomplishment. Our descendents of far off ages, looking down this long vista of time from the other end, will see our present age as the misty morning of human history. Our contemporaries of today will appear as dim, heroic figures who fought their way through jungles of ignorance, error and superstition to discover truth."

In 1936, Pulitzer Prize Winning Poet, Carl Sandburg hailed the achievement and glory and tenacity of man in his epic, "The People, Yes":

"The people will live on;
The learning blundering people will
  live on.
They will be tricked and sold
  and again sold,
And go back to the nourishing earth
  for rootholds;
the people, so peculiar,
  in renewal and comeback.
You can't laugh off
  their capacity to take it.
The mammoth rests
  between his cyclonic dramas.
Between the finite limitations
  of the five senses,
and the endless yearnings of man
  for the beyond,
People hold to the humdrum bidding
  of work and food,
While reaching out when it comes
  their way,
For lights beyond the prism of their
  five senses,
For keepsakes lasting beyond any
  hunger or death;
This reaching is alive.

Man is a long time coming,
Man will yet win.
Man is a wanter and a seeker and a
  hoper."

And then, for all their seeking and wanting and hoping, for all their reaching to become better than they are--to project their progeny into a higher existence than their own, a loftier humanity than what they are a part of. For all this--what are they offered now?

In February, 1978 in a Time Magazine Essay by Robert Jastrow, their answer and recompense are made--shall we say--"Crystal" clear:

"Today the best computer models can be wired up to learn by experience, follow an argument, ask pertinent questions and write pleasing poetry and music. They can also carry on somewhat distracted conversations so convincing that human partners do not know they are talking to a machine.

"The computer imitates life like an electronic monkey. As computers get more complex, the imitation gets better. Finally the line between the original and the copy becomes blurred. In another 15 years or so we will see the computer as an emergent form of life."

"Human Evolution is a nearly finished chapter in the history of life--a new species will arise out of man. Only

a carbon chemistry chauvinist would assume that the new species must be man's flesh and blood descendents with brains housed in fragile shells of bone. The new kind of intelligent life is more likely to be made of silicon."

It's my intention here, obviously, to challenge Dr. Jastrow's contentions and put them in a broader perspective, not on the grounds of their possibility or even their so confidently prophecied inevitability. I contest them purely on the grounds of their desirability.

It is becoming an increasingly familiar exercise for futurists to insist on outcomes for humanity that involve the obliteration of humanity itself, as we know it. What Dr. Jastrow and others are saying is: "This is the way it's going to be--like it or not. If you want things to be otherwise, get off the earth. Your wishes have nothing to do with it.

I'm telling you here and now that I own a computer - I use it - I hope to sell a few computers in years to come. But I don't want to be a computer. I do not want my grandchildren to become computers and I do not want my great-to-the-sixtieth-power grandchildren to be silicon whiz kids. Am I a Carbon Chemistry Chauvinist? YOU BET!

It seems to me that the entire meaning of what has been termed the "Existentialist Revolution" has been grossly misinterpreted or at least misunderstood by Dr. Jastrow and other advocates of the Human-Computer symbiosis-becoming-identity. It involves an inherent philosophical contradiction. On the one hand man is the creater of his own evolution--an essentially existentialist notion--and at the same time on the other, man is being swept along by unfolding technological events to a predictable destiny he may or may not desire to reach - a blatantly deterministic view. In point of fact, what the "Existentialist Revolution" really means, is that man, Homo Sapiens, for the first time in history, through science and technology, can exercise the free will to become anything he wants. It is a new and ponderous responsibility for deciding what that "anything" should be that faces us now so squarely and somewhat frighteningly, not the apocryphal command that we simply "get used" to the idea of what is going to happen to us no matter what. The prospect of being able for the first time to begin to "invent" what we will evolve into is far more exciting to me than accepting a future for mankind that someone says is inevitable. I thought science and technology were supposed to be busily at work removing the shackles of predestination from our future, not forging them.

But the ideas that form the branches of the futurists' philosophical schizophrenia are not new. They are rooted in long established idea systems with which every serious student of philosophy is familiar. Creation of a new race, beyond man, for instance is an idea traceable to Joachim of Floris in the twelfth century and the notion of man having a role or responsibility in his own rebirth--which has now taken a technological turn--goes back to Pelagius, a monk living in the time of Constantine, in the 4th century.

Further, Dr. Jastrow is not the first and certainly not the only contemporary advocate of the Transformation. In 1964, the widely read French Social Scholar Jacques Ellul, painted a rather gloomy inevitability of the New World and the New Race created by irresistable, autonomous technological forces beyond the control of Humanity. It is by no means intended to be read as Science Fiction. Ellul does not write in that milieu. Ellul, however, seeks more to warn than to tout but warnings of hopeless outcomes differ little in effect from advocacy. However, the strident voice on the present scene and probably the one to whom Dr. Jastrow owes his discipleship in propounding the case for replacement of humanity by a "New and Better" race of beings, is Richard Landers, the TRW Engineer, turned Social Philosopher. Landers not only predicts but proselytes with the fervor of the true believer, an age of convergence and identity of man and machine in a world transformed from the biosphere we know to what Landers calls the "Dybosphere" which is dominated by artificial, manufactured forms and substances with few or no vestiges of biological life. An environment peopled not by men but by intelligent artifacts "Superior" to anything human

beings could ever have become. Please note that Landers is still at work building a machine that will reproduce itself, a rather chilling corollary to the tale of the "Sorcerer's Apprentice."

Contributions to the total concept of man becoming non-man have been made from a variety of sources,some even differing widely in background, motive and predicted paths to the same ultimate ends. Most well known of these are: B.F. Skinner who deals with the psychological aspects of the social reorientation and conditioning to social control necessary for the Transformation; Arthur Clarke, of 2001-A Space Odyssey fame, who explores with a more investigative coolness, various ways in which man may achieve symbiosis with machine, but who seems nonetheless certain that through one means or another, the merger is bound to occur; Marshall McKluhan who centers the means for his conception of the transformation of man around the reality of the growing interaction of people on a global scale through electronics.

Perhaps the Utopian Prophet of the millennium of man's transformation who has the broadest scholastic and intellectual base, certainly the largest international following, is the French Jesuit social philosopher, Teilhard De Chardin. His writings and ideas on this subject are far too voluminous to pursue in any detail here. His central theme, however, is the essentially identical nature of the spiritual and the physical universe, a basis which leads ultimately to the dispatch of mankind in favor of a creature more favorably suited to both these universes becoming one.

There are several common denominators detectable in these separate ideas. One of them is especially interesting. It is the similarities that can be found with totalitarian ideals: the essential "messiness" of humanity, its unpredictability and lack of susceptibility to control and direction, its overall inefficiency. One is somehow reminded that Adolph Hitler's obsession with having the trains run on time was not so far removed, in essence, from his ambition to "decontaminate" the world of its "undesirables" by means such as Auschwitz, Buchenwald and Treblinka, or his penchant for wanting to create a Super Race. Another similarity is the total abnegation of the individual in favor of focusing on the masses. Time doesn't permit a thorough analysis of the origins of these concepts, but historically, the controversy on the probable future of man evolves primarily on four "Schools" of Weltanschauung or view of life and existence.

1. Teleologism - A view holding that all things in the universe including life and their ultimate forms and outcomes are predetermined by some overall design or purpose of nature--a purpose not yet entirely known or understood, but even if it were, it could not be changed or appreciably influenced in any way by man or another intelligent form of intervention. It is based on the assumption or Doctrine of Final Cause.

2. Vitalism - The view that the life sometimes called the "Elan Vital" in all living organisms is caused and perpetuated by a special principal in nature, entirely apart and distinct from chemical and physical forces and, as such, life is largely self-evolving and self-determining.

3. Mechanism - The Doctrine propounding that all phenomena, including life, had its origins, therefore has its explanations in physical and chemical matter and that the separation of the inorganic and organic is a question only of degree, not of kind.

4. Existentialism - The view that existence preceded the essence or identity of man and that man alone, both collectively and individually, is solely responsible for his ultimate destiny, his survival or demise, his triumph or destruction, through exercise of his free will. In this view it is only the exercise of the will in opposition to hostile forces that will determine man's survival and triumph; the earlier existentialists emphasized the aspects of nihilism, despair and help-

lessness of man. Later ones have focused more heavily on the power of the will and the responsibility of man for his own destiny. Jean Paul Sarter, for example, insisted that: "Man is alone and without excuse," and fully accountable for what he makes of himself.

While these four views in many ways conflict with each other--Mechanism and Vitalism obviously couldn't coexist very harmoniously--the first three do have in common the notion of determinism or the belief that forces independent of and beyond the control of man are shaping his destiny and all he can do is try to understand them--he cannot change that destiny even if he wants to. Existentialism, however, stands in direct opposition to the determinism of the other three views. Indeed, they are logically mutually excluding. Yet the advocates of the New Utopianism embrace both of these opposites, crunch them into some grotesque whole and hurl the monster with evangelistic zeal into the face of reason, and into ours, with the panting expectation that we will clutch to our collective breast the notion of evolving ourselves out of existence as the only available salvation for our future.

But this contradiction only serves to arouse some suspicions of the New Utopians. It still does not deal directly with the desirability of realizing their predictions.

We have to look at the question of desirability from two angles: First, we have to ask and at once answer for ourselves the question, "What is Man? What is the essence of the identity of our species?" Then after we have satisfied ourselves with an acceptable definition of what we are, we next have to ask: "Is man as we know him, or as the essence of being we have defined, or any part of it, worth keeping in existence?"

The latter question is largely one of conscience and answerably only privately by each of us but we will deal with it in part in a few moments. The question of a definition of man, however is more difficult. In fact, there can be no lasting answer, as is the case with any process--and man is a process. As yet we do not even have any clear idea of what man is entirely and what his capacities and capabilities may already be--not to mention what they will become--if the species is allowed to continue. If we can find even a modicum of value in this fragile stumbling, seeking, wanting, hoping creature we call man, it would seem utter nonsense to talk of foreclosing the possibilities that remain to be discovered about man and to abandon them in what is beginning to look like the first light of a yet unborn glory.

In the field of Parapsychology alone, we have but scratched what now appears to be a very deep surface of dimensions of man widely regarded as recently as two decades ago as hokum or fantasy or witchcraft. How can we say so confidently that we will not discover aspects of these dimensions that would yield capabilities no inorganic creation of even super-intelligence could ever achieve? How do we know? We don't. Indeed, how do we know that there is not something unique and of a distinctly higher order of existence in the organic? Would the "machine" part of the cyborg be in reality a handicap? How do we know? We don't. How do we know that whatever purpose in nature we exist to serve has been served, or will be before our heralded demise; and how do we know that anything other than man could serve that purpose? How do we know that? We don't. We have not yet begun to know and yet some are already closing the coor on the future of biological man, abandoning as yet unknown future by committing the patient to certain death to cure his ills because we cannot in our ignorance find any other immediately effective remedy. Even if we can find nothing about man now that it is worth keeping the species for, which is doubtful, is that sufficient reason to plan to stop looking?

Winston Churchill, when approached on the question of granting national independence for certain British protectorates replied: "I did not become the Queen's First Minister in order to preside over the dissolution of the Empire." Well perhaps he didn't preside over it and perhaps he didn't wish it, but it happened anyway.

Perhaps man did not become the remarkable phenomenon he already is becoming in order to devise the means to his own extinction. Perhaps we really don't want that result, but if our zeal to proliferate our technologies is some how irretreivably hooked up with anti-human, or super human ideas of evolution--a physiologically centered Utopianism--it may well happen anyway.

Contemplating these happy predicted outcomes for humanity tempts one to think that if T.S. Elliot had known what we in this room know today, he might have been tempted at least to phrase the now familiar lines from "The Hollow Men" somewhat differently. He might have said:

> This is the way the world ends;
> This is the way the world ends;
> this is the way the world ends;
> Not with a bang,
> But with the stone - silence,
> Of the infintesimal instant
> When the last bit of human
> intelligence,
> Takes its last, one way light
> speed journey to its brave new
> silicon home,
> Free at last,
> From the obsolete organism still
> so incapable of perceiving
> either the time interval for
> the transit
> Or for humanity to vanish.

Or if we really wish to strike a posture of acceptance, we might devise our own advance epitaph in this mode:

> HERE LIES MAN
>
> Circa 5,000,000 BC to Circa
> 5,000 AD
> An obsolete form of intelligence
> too encumbered by complexity
> of its internal information systems and an extraneous factor
> it called "emotions" to survive
> the inescapable rigors and stres
> of the universe.

But the point of all this is not to applaud despair and bow to inevitability. The point is to illuminate the fact that, although some would keep it from us, there <u>is</u> a choice.

Professor Victor Ferkiss of Georgetown University, for example, believes that everything the New Utopian Prophets are saying are distinct possibilities, if certain choices are not taken and if certain values are not defined <u>in our time</u> and maintained for the future.

In his book, "Technological Man," Dr. Ferkiss states, "Nor will technological man be a new biological type created either by manipulation of man's genetic structure or by carrying man-machine symbiosis to the point of altering human integrity.

"Such a development would mean that technological man had failed to come into existence and bourgeois civilization had fallen prey to monsters of its own creation."

Elsewhere Dr. Ferkiss is even more definite: "Linkages of man to machines and technologies that would make him irrevocably dependent on lower orders of reality would be anti-evolutionary. Man has avoided the error of the crustaceans in protecting themselves in a way that made future development impossible. Man has avoided the dead end of the social insects who have created a marvelous structure in which the nothingness of the individual and the mobility to change are opposite sides of the same coin. Man's destiny lies in continuing to exploit this "openness" rather than entering into a symbiotic relationship with the inorganic machine that, while it might bring immediate increments of power, would inhibit his development by chaining him to a system of lesser potentialities. Man must not only stand above the machine but must be in control of his own evolution."

I agree.

But if we are to do what Dr. Ferkiss suggests, if we are to design our own choices, we here in this fledging industry that will have so much to do with shaping the future, must first design the value guidelines by which those choices can be defined. We cannot escape being numbered among the early architects of what will become of Man of the Future: Technological Man.

In that direction I have proposed fourteen principles to govern the design, development and application of sentient

devices, inorganic forms of intelligence and artificial materials to emulate, augment or extend human capabilities. Copies are available at the door as you leave, or may be obtained by writing directly to me at the address shown in the "Proceedings." I would appreciate your reviewing them and letting me know your ideas, comments, criticisms or recommendations or whatever, by way of the mailable comment sheet attached. If I can get a substantial number of approving signatures, I would like to submit the guidelines to the National Science Foundation and other forums of scientific and technological thought as a statement or manifesto of ideals for future research and development.

I want to leave you with two thoughts, one mine, the other, Carl Sandburg's:

First, consider that intelligence can't be the only thing about man that is important; that intelligence, experience, knowledge, imagination and wisdom are not all the same thing. If these words all meant the same, we could easily do without all but one of them in the language. Though not the same, they do relate in some interesting, mutually clarifying ways. It can be said that wisdom is a combination of two things: Experience tempered by knowledge and knowledge seasoned by experience; "Wisdom", it's said, "comes with experience," but then many things come with experience when untempered by knowledge--prejudice for instance. Experience isolated from knowledge tends to regard itself as the universal sample, the arche type of all things possible. Knowledge isolated from experience tends to regard itself as a god. Experience can take you only to the limits of your senses; knowledge can take you to the boundaries of the universe; imagination can take you to the borders of the possible and beyond; wisdom will help you understand what to do when you get there. Intelligence enables you to survive, perhaps even prevail; wisdom enables you to help mankind do the same.

"In the darkness and with a great
bundle of grief,
the people march.
In the night and overhead a shovel
of stars for keeps-
the people march.

Where to, what next--
Where to, what next--"?

References


Time Magazine, Feb. 20, 1978, p. 59

Complete Poems, Carl Sandburg
Marcourt Brace, 1950.

"The Origins of Totalitarianism"
Hannah Arendt, Meridian Books
1958.

"Profiles of the Future," Arthur C.
Clarke Bantam Books, 1964.

Omni Magazine, Oct. 1979, Dec. 1979

"The Technological Society," Jacques
Ellul Knopf, New York 1964.

"Totalitarian Dictatorship & Autocracy":
Friedrich & Brzezinski Harcourt
Brace, New York 1956.

"Cybernation: The Silent Conquest"
Michael, Center for Study of
Democratic Institutions, Santa
Barbara, 1962.

"Medium is the Message," McLuhan Bantam
books, 1965.

"The Future of Man," Teilhard De Char-
din, Fontana Books, London 1964.

"Technological Man," Victor C. Ferkiss
George Braziller, New York 1969

# THE BEST OF THE COMPUTER FAIRES, VOLUME I:
## Conference Proceedings of the FIRST West Coast Computer Faire

**TABLE OF CONTENTS**

# THE BEST OF THE COMPUTER FAIRES, VOLUME II:
## Conference Proceedings of the SECOND West Coast Computer Faire

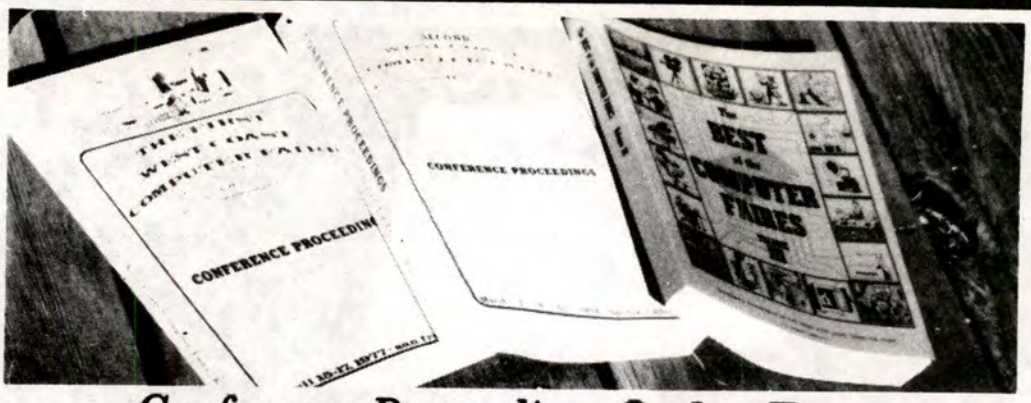
**TABLE OF CONTENTS**

# THE BEST OF THE COMPUTER FAIRES, VOLUME III:
## Conference Proceedings of the THIRD West Coast Computer Faire

TABLE OF CONTENTS

# THE BEST OF THE COMPUTER FAIRES, VOLUME IV:
## Conference Proceedings of the FOURTH West Coast Computer Faire

TABLE OF CONTENTS




# *Conference Proceedings* Order Form

☐ I enclose $13.72 (U. S. ) for each of the the following *Proceedings* (including shipping* by United Parcel Service)

___ of *THE BEST OF THE COMPUTER FAIRES, VOLUME I: Conference Proceedings of the FIRST West Coast Computer Faire*

___ of *THE BEST OF THE COMPUTER FAIRES, VOLUME II: Conference Proceedings of the SECOND West Coast Computer Faire*

___ of *THE BEST OF THE COMPUTER FAIRES, VOLUME III: Conference Proceedings of the THIRD West Coast Computer Faire*

___ of *THE BEST OF THE COMPUTER FAIRES, VOLUME IV: Conference Proceedings of the FOURTH West Coast Computer Faire*

___ of *THE BEST OF THE COMPUTER FAIRES, VOLUME V: Conference Proceedings of the FIFTH West Coast Computer Faire*

☐ Give me a **50% DISCOUNT** on pairs of Volumes I-IV.
I'm taking advantage of the *TWO-FOR-THE-PRICE-OF-ONE SALE†*.
I am only enclosing $_____

☐ Give me a **50% DISCOUNT** on any of Volumes I-IV.
I'm taking advantage of the *HALF OFF OFFER†* with the purchase of Volume V.
I am enclosing only $_____

***Please write for costs to shipping locations outside of the continental U. S.**

Please charge this to my ☐ Mastercharge, ☐ Visa Card # _____ expiring_____

*Signature*_____ Phone (_____)_____

Name _____

Street Address _____
*(UPS cannot deliver to a box address)*

City _____ State_____ ZIP/postal code_____

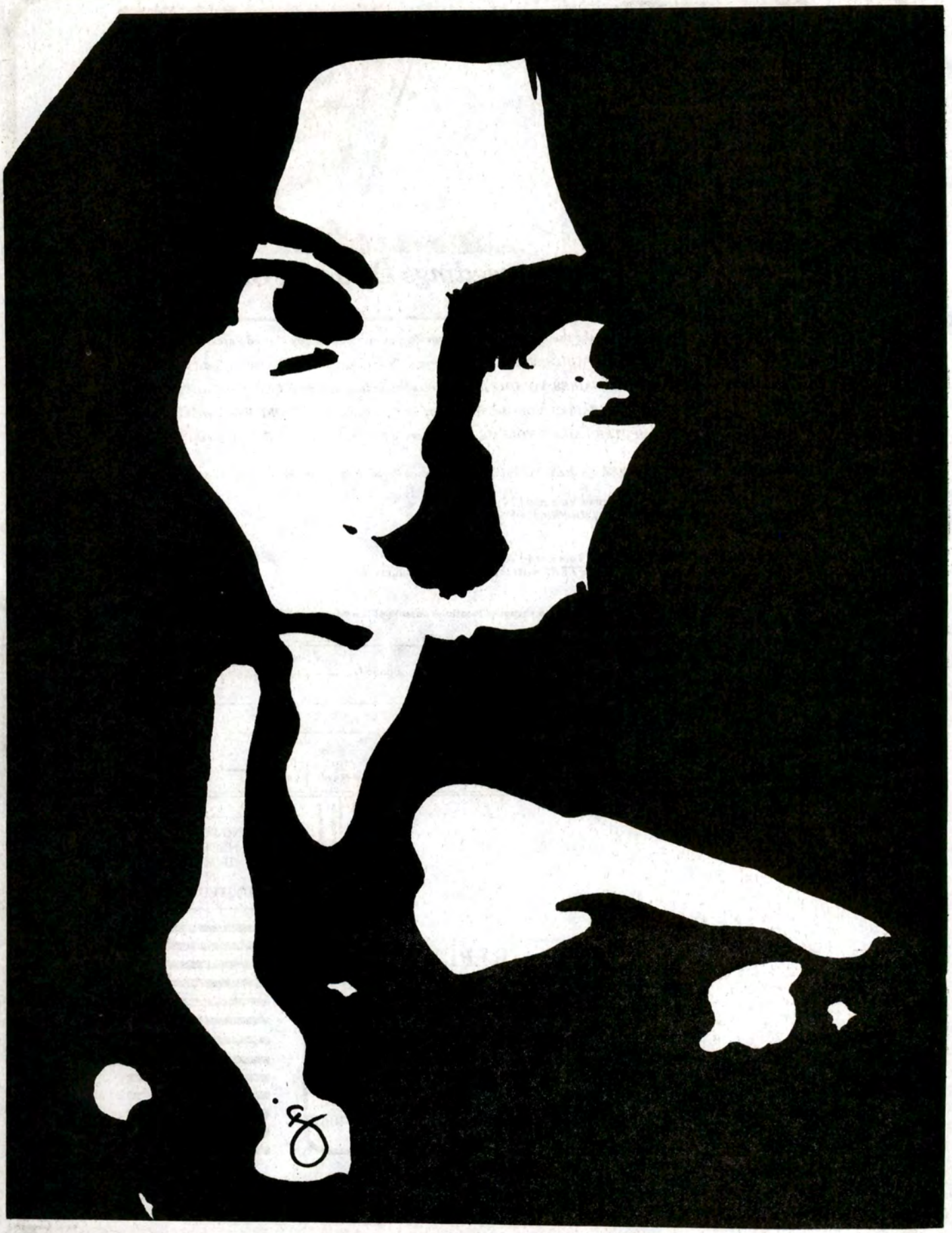
**† This offer will be honored only on orders received by March 20, 1980.**

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

**BUSINESS REPLY MAIL**
First Class Permit No. 169 Redwood City CA

POSTAGE WILL BE PAID BY ADDRESSEE

**COMPUTER FAIRE**
333 Swett Road / Woodside CA 94062
(415)851-7075



Faun Jackson, illustrator

Paul Jackson, illustrator