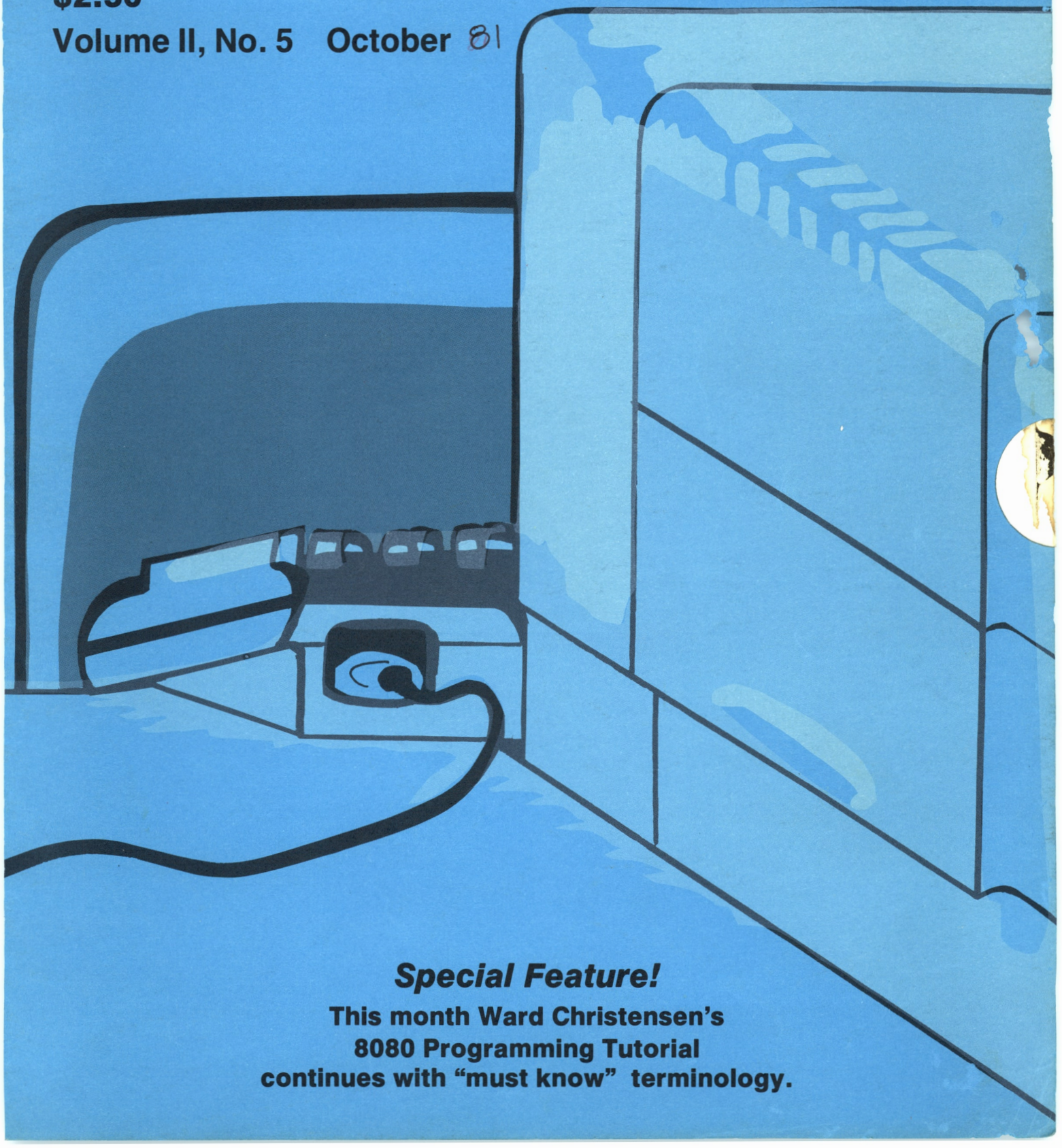
LIFELINES®

The Software Magazine™

$2.50

Volume II, No. 5 October 81



Special Feature!

This month Ward Christensen's
8080 Programming Tutorial
continues with "must know" terminology.

LIFELINES®

Editor-in-Chief: Edward H. Currie
Managing Editor: Jane V. Mellin
Production Assistant: K. Gartner
Administrative Assistant: Susan M. Sawyer

Volume II No.5 October

CONTENTS

 Lifelines, Volume II, Number 5. Published monthly by Lifelines Publishing Corporation, 1651 Third Ave., New York, N.Y. 10028. Telephone: (212) 722-1700. The single copy price is $2.50 domestically, including the U.S., Canada, and Mexico. The single issue price for copies sent to all other countries is $3.60. A one year's (12 issues) subscription is priced at $18.00, when destined for the U.S., Canada or Mexico, $40 when destined for any other country. All checks should be made payable to Lifelines Publishing Corporation. Foreign checks must be in U.S. dollars, drawn on a U.S. bank; checks, money orders, VISA, and MasterCard are acceptable. All orders must be pre-paid. Please send all correspondence to the Publisher at the above address. Postmaster, please send change of address to Lifelines Publishing Corporation, 1651 Third Ave., New York, N.Y. 10028. Application to mail at second class postage pending at New York, N.Y.

Editorial Comments

The Quiet Revolution

The microcomputer age has been in a state of continual evolution since its beginning in 1975 with the introduction of the Altair.

Though not many realize it, almost every aspect of the microcomputer industry had its beginning with the early Altair development in Albuquerque, New Mexico: the first computer in kit form, the first microcomputer high level languages, the first microcomputer convention, the first microcomputer publication, the first software publisher for micros, the first computer within the reach of all levels of society, etc.

The Altair Age began quietly, without anyone really suspecting that it would spawn several hundred new companies, nearly a hundred publications, a dealer network of some two thousand dealers, and an installed base of literally hundreds of thousands of microcomputers throughout the world, an unending demand for floppy disk drives and employment for millions.

How would one have guessed that this microcomputer would be capable of compiling virtually every language available for computers of any type, playing games, music composition and production, splendid graphics, teaching its owner a wide variety of skills, linking up with other micros around the world via the ordinary telephone line, speech recognition and speaking itself, and a myriad of other activities? And it all began when one man decided that he wanted to provide a "real computer" to the masses.

How could anyone have ever dreamed that entrepreneurial companies like Microsoft, Lifeboat and others could end up with well worn paths to their doors beaten by the major manufacturers, who were somehow unable to come to grips with the software required for these machines?

All that came after was largely recastings of that which went before. Consider for example languages such as BASIC. Microsoft BASIC is the BASIC of the entire industry, and whether you're running BASIC on the Apple or Radio Shack machines (or for that matter on most any other machine: 8080, 8085, Z80, 6502, etc.), if you look closely enough you will find to your amazement that they are all derivatives of the early Altair BASIC which was written by Microsoft.

And each time that this revolution has appeared to stabilize another new development has sent it hurtling forward again; whether it's the introduction of the Z80 or the 8086 or the 68000, or the advent of the mini-Winchester, the momentum is continuing to build.

And just as we begin to suspect that there is a clear view of the future at hand, perhaps the most significant event of the decade occurs.

IBM enters the market with the IBM personal computer. This machine, with its powerful 8088 and enough RAM storage to contain an entire floppy, clearly marks the beginning of a new era.

Its introduction has been largely without fanfare and has gained little real attention by the press, who are content to describe IBM's hardware and assume that the there will be a big "fallout" of small manufacturers. (Incidentally, these same self-styled gurus have been incorrectly predicting this "fallout" for the last five years).

The fact that IBM has chosen to standardize on an operating system provided by Microsoft has also gone almost totally unnoticed. It seems incredible that while the industry has drifted into widespread usage of a *de facto* standard operating system (due mainly to the large number of formats introduced by Lifeboat Associates), suddenly a new star appears on the horizon and is provided by the same company which gave many of most significant languages available for microcomputers.

Microsoft is hard at work creating what undoubtedly will prove to be the most powerful operating system in widespread use in the near future. This will do much to stimulate production of large numbers of 8088/8086 applications for both office and home, as the several hundred small companies which make up this industry rush to follow IBM's lead.

So watch carefully the developments of the next eighteen months and you will have a unique opportunity to observe what may well be the most exciting chapter of all in what began as "The Age of Altair".

Edward H. Currie

The Pipeline

by Carl Warren

About software development
If you've been agonizing over a software project of your own, or trying to find out why a purchased package appears not to work as advertized, here are some things that will most likely save you time, money, headaches, and make you a hero all at the same time.

When designing a program, you might do well to take the time to really think it all the way through before beginning the coding process. This is what oldtime application designers do.

The process involves taking a notebook—an engineering notebook with E4 grids is best—and developing first a rough outline of what you plan to do. Then take each part of the outline and expand it to the subfeatures. What you end up with is a basic specification. Your next chore is to further define the specification as to what the inputs and outputs are to look like, what error and help messages are to say, and what causes each to happen.

Once you've developed this, you can move on to developing the logic. And yes, flowcharts do help regardless of what anyone says. There are several steps in developing logic for an application program and these include:

☐ Break down each functional part of the application. Take the install portion, for example, define the logical flow of the operation in words and direction.

☐ Develop a logic flow chart of the operation. Indicate the start, middle and end, and branches to the subroutines that will be used.

☐ Tie in the logical physical device operations; printer, disk, terminal and so forth.

When you've developed the logic for each of the subpart of the program, develop a box diagram of flow of the entire program showing how each subpart fits together.

Should you be using a terminal that offers screen handling functions, include tables of the control and escape sequences that will be required and where and how they fit into the program.

Once you've completed all the above, you're still not ready to code. What you must do now is determine how to use the program; that is, develop a user's guide based on your design criteria. Remember though that this isn't etched in iron and will probably change when you finally get to a fully debugged and operational program; but it does give you a target to shoot at.

Now you can begin coding the application. The interesting thing to note here is that coding will only take a fraction of the time it would if you took a head on approach. All that is required is to follow your logic flow and watch the code appear.

When the program is coded there is still more work to do. This involves testing to see if it works for all cases as designed.

To fully test your program (and incidentally, those that you buy) develop a test script. The script should be developed from your specification, or in the case of a bought package, the operator's manual. Each input and expected output is listed, and room made for what actually occurs. You even put in those inputs that will cause an error to see if the error traps work correctly.

The next step in writing the test script is to develop inputs that shouldn't work, or those that should only work with the operating system at command level. You may get some surprising results.

With your test script in hand begin testing as you written, don't deviate from the script or you won't be able to find and fix the problem. Perform the test at least three times to make sure you haven't missed anything. Then go back and try those inputs that caused a problem, noting exactly how the problem occurred and whether or not it was random.

Now you have a well thought out document that allows you to quickly go back to the code and make changes. Be aware, however, once the changes are made testing must be redone so you can ensure that a fix didn't cause another problem elsewhere.

It would seem that after you go to all this trouble, you wouldn't need to worry about bugs creeping into the program. But try as you will you won't find all the bugs crawling around, but somebody who buys your package will. Consequently, you will need to keep a functional problem log (FPL).

The purpose of the FPL is to keep track of reported problems, could they be repeated, the system configuration, and what was done to fix them and when. The FPL will not only add to the viability of the program, but serve as an aide to future updates, or to new packages that use routines all ready being employed.

It's interesting to note that apparently very few software houses employ these techniques, nor do software reviewers. And you as a software buyer can end up suffering.

What you might want to do is to develop a test notebook for packages you buy with a test script, and FPL. When you're unhappy with a package you can point to real evidence. Note though that hardware problems can contribute to a problem as well, and should be kept track of in the FPL.

A typical test script looks something like this:

INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	COMMENTS

Notice, that you have an expected output and an actual output. What you will want to note is whether or not the output is where it's supposed to be on the screen or printer, the

comment line notes the FPL document the problem was fixed on.

The FPL is similar but includes room for notations of what the detailed problem was, what caused the problem, when and how it was fixed.

A very easy way to set up this documentation is to employ WordStar, or ideally, dBase II. This way you can use your computer to handle some of the mundane tasks required for good application design.

Of course all programs aren't applications but deal with system software, like an O/S for example. The same techniques can be employed to ensure a good design.

This approach to software design isn't new, but has been employed for a number of years by major software developers for mainframes and minicomputers.

The problem that seems so rampant in the microcomputer industry is that too little time is spent on the initial program design. This as a consequence, is the basic reason for those $25 updates and lack of functional characteristics that ZOSO likes to talk about so much.

Along the same lines as basic software development is Computer Aided Instruction program design. Similar techniques are employed in creating a viable teaching package, but other things must be taken into account. Among these are: Who is going to learn from it, what is it **really** going to teach, and did it teach it.

The first two criteria are the easiest find out and develop using outlining techniques and surveying the perspective audience. The latter is the most difficult, since it means doing some kind of testing to make sure you wrote the package correctly.

To understand interpreting the functionality of a C.A.I. lesson, you must first understand C.A.I. Unfortunately, many who are developing computer oriented training programs are unaware of how a lesson goes together. A C.A.I. program is very much like the FORTH language in concept-it relies on threads to provide continuity.

C.A.I. programs are made up of end lessons, which are in turn made up of frames. These frames are linked together by 'threads', or roadways, that are taken dependent on the answer given, or the path chosen by the designer. Multiple threads leading ultimately to the same place must be built-in to ensure that the maximum amount of information is conveyed to the student.

When developing a C.A.I. program, decisions must be made about the contents of any given lesson in the program, as to length, type of information, and degree of difficulty; moreover, what each frame will contain, and how they will be threaded together.

The best method of creating a C.A.I. program is to employ a story board, One story board per lesson. You can do this with a large chalkboard. Draw the number of boxes (frames) that you think you will use—you should know almost exactly if you did the specification correctly, then put in a one or two sentence abstract of the information the frame will convey. Next number each frame as to logical sequence, then connect them together by sequence dependent on answers given.

Once you have done this you can bring in someone unfamiliar with the project to see if you're following an understandable path. Now go back and using screen layout sheets develop detailed displays of what each frame will look like. You'll want to Xerox these because your next task is to put them together in the various orders, based on the decided upon threads, that each frame can be displayed.

Now begin the coding process. The best method involves coding text material with an editor or a graphics package and saving each frame as an overlay or callable file. If the program is short enough code them as subroutines. You might want to look at using Pascal as the language, since it offers some facilities that make it almost ideal for C.A.I., or you could use assembly and create a small main line program that treats everything as overlays. What you want to try and avoid is hardcoding the frames since you will more than likely want to change them as time goes by.

Oh yes, be sure to keep good logs on how well the program is working. Who knows, you might end up with a product Lifeboat will want to carry.

ZOSOTTTT

While I was away on holiday last month, some really great topics came to mind. Instead, let's discuss these: 'CRT Terminals - The Missing Keys', 'Floppy Disks - The Good the Bad and the Ugly', 'Word Processing - A Tool for the Famous?', 'Couples Apart - Is the 8080 to Blame?', 'More Powerful Processing - Some Modest Proposals', 'Zoso T Shirts - Wear With Pride', 'Lapel Buttons - I Can Use the Publicity', and finally, a contest. If you think I'm kidding, think again!

Como estan Vds?

The courtesy and sincerity of this Spanish greeting can not quite obscure its illiteracy. By way of lame excuse, I think it's the fault of [most] terminal manufacturers who just don't find it worth their effort to include those inverted question marks which 'flag' the beginning of a properly formed Spanish question. Also not to be found are the tilde (which should appear over the 'n' in espanol), the cedilla (required by the 'c' in your typical French garcon and a variety of other words in French, Rumanian, Portuguese and Turkish). While I'm on the subject, they've also omitted accented vowels, the ever popular German umlauts, and the slashed 'o' without which Scandinavian languages lose a lot (correctness for example). What our overpriced cataract boxes offer instead is numeric keypads (often as costly options), to duplicate keys which are already there. I guess the guys who design these things must think we're all accountants or keypunchers (in the most menial sense). On the plus side of the ledger, for a goodly sum one can acquire an APL terminal to communicate to a more select audience. ('... plus side of the ledger...', did I really say that? Hmm... maybe they're right).

Do you remember 'R' (the local repairman with the imposing collection of broken SuperTerms)? Every couple of years, he runs exhaustive tests on brand name 8" disks. His most recent tests were conducted this Spring.

Notice that I said 'brand name disks'. 'R' has a drawer full of drive heads which have been abraded into premature oblivion by bargain specials. One of those heads was mine. I guess it's just another example of how outrageously expensive mini-educations can be these days.

The disks which 'R' tested most recently were BASF, Wabash, Verbatim, 3M, Dysan, Maxell and Memorex (all single-side, single-density). His tests involve reformatting several samples of each brand for extended density. This blows a few ducks out of the water, as they say. For the better disks, his computers keep track of errors which accrue during roughly 12 hours of demanding seek-read-write cycles.

Two brands distinguished themselves by performing flawlessly. These two were further tested by a specially 'gimicked' drive in which the head amplitude had been deliberately reduced out of 'spec'. Only one survivor remained. May I have the envelope please...
The Winner: Maxell
The Runner-up: Dysan
Also rans: All the others

I have every confidence in the integrity and objectivity of the test methods which 'R' uses, and, as far as it goes, I concur with his resulting conclusions, still he and I do not look for the same things. His tests help him decide which single brand of floppy disk to use and sell (Maxell, since I've known him), and I mainly look for brands not to buy. 'R' and I differ in another major way; he disdains the use of a hole punch to make 'flippy disks'; I do not. Here's what I think of the brands he tested:

I agree that Maxell disks are the best I have had no bad samples to date. The side they want you to use seems to be 99.99999% reliable at 600k, even when certified only for single density. The reverse sides seem just as good. My one trivial gripe concerns the labels they provide; the colors are bilious, and once a Maxell label is applied, nothing short of a heavy duty belt sander stands the slightest chance of removing it. More than one [Maxell] insider has told me that all of their 8" disks are subjected to the same rigorous testing. What this means is that any premium you might wish to pay for extra density certification or for factory punched two-siders is money that might best be spent elsewhere. More or less the same remarks apply to Dysan.

It seems to me that Wabash is continually experimenting with quality and packaging. Amongst the SD/SS eight inchers they have sold in the last few years, I have seen everything from attractive and reliable disks in beautiful cardboard library cases to unmarked horrors in plain white sleeves. Amongst the Wabash disks which I might otherwise have rated 'best-buys', I have found far too many samples which emit very strident, high-pitched squeals as they spin in their jackets. This sound reminds me of the nerve-jarring collisions between chalk and blackboard which used to interrupt my naps in the classroom.

BASF now produces a distinctly average disk, and compared to some of their earlier products, this represents a major improvement. Once they get the disks right, they should work on the packaging; flimsy boxes with poorly scored labels in any color you want, just as long as you want battleship gray.

Memorex has invested in many a TV ad to convey the notion that their audio cassette tapes were capable of shattering glass. Despite this, all I saw were ho-hum tapes in original but nonetheless wretched cassette boxes. Now they sell disks and little has changed. The Memorex tradition of user-hostile packaging continues unabated. The culprit this time is their idea of protective disk sleeves; singularly bad! I'd like to move on, but I'd be remiss if I failed to mention that like Wabash, Memorex has frequently provided me with some bonus sound effects; very rarely it's near silence; more often it's what sounds like an electric floor scrubber, and on occasion what sounds like an extremely unbalanced tire hopping down the freeway.

Verbatim disks have purportedly undergone top to bottom overhaul and redesign in recent months. According to Verbatim's ads, this has resulted in a substantially better product. I'm inclined to believe this because they really had no other way to go. I've had too much bad luck with Verbatim disks to seriously consider trying a box of the latest 1981 models. I do however remember this much

⇒

about the ones I didn't used to like; no competing brand ever took a back seat to Verbatim when it came to hard errors and unwanted sound effects, but it's all a fading memory. As of now, I'd hazard a guess that Verbatim is mired in an image identity problem. Their ads imply that the new 'Datalife' disks are the best money can buy, yet they are widely available for only pennies more than unbranded specials. Pay a lot, pay a little... Not me, pal.

3M disk labels have a very professional look. They are my favorites. So much for the good news. In each box of disks you are virtually assured of getting a few which sound like a drill smoking its way through one of your wisdom teeth. The liner material could probably stop a copper jacketed 30-06 slug fired from just inches away. Some would call this quality; all I know is that punching holes for the flip side is nigh unto impossible. Besides, I'm inclined to question 3M's quality control because I once bought a box of ten, five of which had no indexing holes punched in the jacket. Initially, I was going to send those useless disks back to Minnesota along with a nasty letter, but I figured they'd just replace them, so I opted to save my time and my stamp as well.

An interesting little article about word processing appeared in TIME magazine (August 10, 1981 issue); and I do mean little; the cover story that week was about ice cream. Anyhow, they reported that ex-President Carter is in love with his word processor (especially when brownouts don't wipe out a days worth of his memoirs), and that novelist John Updike can't be bothered to learn how to use one. Draw your own conclusions!

A few weeks ago, while in a Rastafarian haze, it suddenly dawned on me that computers can do some subliminal numbers on our heads. Check this out (8080 mnemonics):

```
;
BEGIN:   CALL    MOM
         CALL    DAD
         CALL    POP
;
         JMP     INLAKE
;
MOM:     RET
DAD:     RET
POP:     RET
;
INLAKE:  HLT
         END
```

This program will not work, and what that implies is shocking. If you are the kind of son or daughter who wants to stay in touch with your parents (who have split), you are partly out of luck as far as the people at Intel are concerned. It's OK to give Mom a ring, but Dad (or Pop if you prefer) will just have to sit by his lonely telephone waiting for the call that can never be made. In the face of this sort of injustice, it's little wonder that his drinking problem has gotten out of hand.

The big problem with machine instructions (regardless of how advanced the processor) is that none of them do very much. Just as an example, consider these familiar codes:

```
PCHL
MOV M,A
RNZ
XRA A
etc.
```

Big deal! If computers are becoming so powerful, why can't simple instructions get them to do some real work. For example:

```
CIA          ;Compute Interest (on) Account
FBI          ;Find Best Investment
DM D,L       ;Do My Dishes & Laundry
CCCC         ;Cancel Credit Card Charges
             ;note: external Modem required.
CRA '%'      ;Compute Ratios As: ;Percentages
FDS          ;Find Date (for) Saturday
TF BS        ;Try For:
             ;--» Brooke Shields
SHAM         ;Slip Her A 'Mickey'
FTA          ;Finish This Article
```

These hypothetical mnemonics are just quickie examples which perhaps should have been chosen more carefully. I get this nagging feeling that I've some of them in other contexts. Nonetheless, this is the section to reread if you plan to enter the contest.

Next on the agenda is the T shirts. These are original designer items (designed by me), available in beige, white and gray. The name of yours truly is boldly silk-screened where you might otherwise expect to find an alligator. You can win one of these priceless keepsakes by submitting useful material* to *Lifelines*, sending me letters or ideas which I find worthwhile for whatever capricious reasons, or by distinguishing yourself in the contests which we'll be having from time to time. I'm told that the shirts will also be given as a bonus present for those who give a *Lifelines* subscription to a friend during the coming holiday season. There are also lapel buttons to honor lesser achievements. I travel quite a bit and I'm planning some nice surprises for anyone I meet who is wearing either an official shirt or lapel button. So even if you live in what used to be a two bit whistle stop before the railroad spur was abandoned, don't count yourself out!

Finally, we come to the contest. The best five entries will win a shirt and five assorted lapel buttons. A pair of lapel buttons will be awarded to the ten runners up. In this particular contest I am looking for whimsical mnemonics to be used by a mythical machine. A good example of the genre can be found a few paragraphs back.

Official Contest Rules: Enter as often as you like. There is no restriction to the number of entries which may be submitted in a single envelope. Don't you wish other contests were as lenient? Address entries c/o me, Zoso, to: *Lifelines*, 1651 Third Avenue, New York, NY 10028. Be sure to include shirt size and color preference just in case you win. Entries will be judged mainly on originality and wit, but it can't hurt if you use every trick in your arsenal to influence the judges favorably. All entries must be postmarked before December 1st, 1981. I should probably add the standard "Void where prohibited by law" nonsense, but I don't think I will.

Good Luck,

Zoso

*Letters, comments, jokes, and other useful tidbits are welcome and will be rewarded with a Zoso shirt and/or lapel button.-Ed.

A REPLY TO ZOSO

4 August 1981

Gentlemen (and Dr. Zoso),

In regard to your 'review' of the program Disk Doctor (*Lifelines* Volume 2 Number 3 August 1981), I would like to set the record straight. I am the original author of this program and wish to say that while no program is ever perfect, I am proud enough of this one to sign my real name to it. Perhaps some day you will feel that way about your column.

While you complained about (and exagerated[sic]) the humor contained in some of the prompting menus of Disk Doctor, you made ample use of the medical allegory to cover your lack of legitimate complaints about the product. Since I believe Disk Doctor to be the only comprehensive, CP/M compatible, disk recovery program for the *nontechnical* user, I think you have done your readers a disservice.

The only legitimate complaint that I could decipher was that Disk Doctor would not install properly on systems that contained different density system and data tracks. This has been corrected and you would have received notification that you could receive this corrected version free if you had completed your registration card.

As to your complaint about having to run the installation program before using Disk Doctor, this program is one of the strong features of the product. The semi-automatic install program (written at SuperSoft) adjusts Disk Doctor to the skewing pattern and sector organization of your equipment. It need only be installed once for each system, and not each time the program is run (in the event that this was your problem). Additionally, Disk Doctor is shipped installed on the format ordered, and the install program is provided to allow installation on *other* systems that the user may have.

You stated that Disk Doctor is 'scandalously overpriced' at $100. Since there is no other CP/M disk recovery program with Disk Doctor's combination of automatic features (to my knowledge), price comparison is not easy. One of the four functions of Disk Doctor is the ability to display recoverable erased files and 'unerase' them. Two other products are being marketed that provide only this function and cost $36.50 and $60.00. The less expensive of these two products is only available for two disk formats.

Another function of Disk Doctor is the 'reclaimation'[sic] of the disk when bad sectors arise. There is a single function product that offers this capability except that unlike Disk Doctor it will destroy any files on the disk, and it can only be used on CP/M version 2.2 (Disk Doctor works on both 1.4 and 2.2). This other product is priced at $80. Put together, these programs provide less than half of the capability of Disk Doctor.

The wide variations of primitive disk functions found in various CP/M systems makes a product like Disk Doctor more expensive to market than one that works on the uniform system calls of CP/M. In any event I make no apology for the price, and yours is the only complaint I have heard.

As for your doubt of the claim that Disk Doctor typically recovers 87% of all disks with directory failures, the number is easily derived. Most systems contain the directory in a single 'group'. This group typically consists of 8 (or more) physical sectors. If any sector will not read CP/M will perform no operation on the disk. Since wear is uniform on this track, failures are randomly distributed. Thus, at least 7/8 ths of the entries can be recovered, and 7/8 ths is 87.5%. The .5% was discarded to allow for a few very large multiple extent files that might contain entries in two sectors of the directory.

Your reference to the dangerous vivisection' done by Disk Doctor is substantially incorrect. Each function of Disk Doctor is carefully planned to avoid leaving the damaged disk *worse* than it was. During recovery operations of a type that indicates the disk may not be writing correctly, no write operations are performed on the damaged disk.

Finally, in regard to your apparent obsession over SuperSoft's spelling of 'Diskettes/Discettes' perhaps you should consult Webster's Dictionary; both roots are acceptable. Memorex sells 'Discs' and Radio Shack sells 'Disks', but both fit nicely in my drive!

While I welcome honest criticism of any product with which [sic] am associated, I do not appreciate reckless defamation. I have been very gratified by the customer response to this product, and it is also gratifying to know that as I write this, someone may very well be recovering weeks of hard work through the use of Disk Doctor.

Sincerely,
John M. Holland

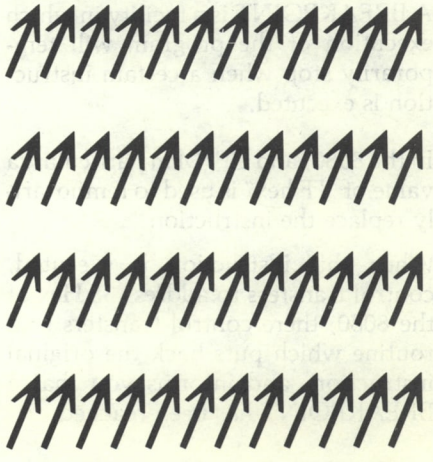
Editor's Note: When *Lifelines* went to press, Zoso was away and could not be reached for comment on Mr. Holland's letter.

Coming Soon

In coming months we'll be hearing more on 8080 programming from Ward Christensen.

A review series on sorts is also in the works, along with an exhaustive Pascal review. And of course, our data base management series will continue to investigate the many products of this nature available.

Do you have ideas on what should be reviewed, or would you like to evaluate software for *Lifelines*? Send your suggestions in. If you'd like to write for us, send us an account of your software experience and note your areas of expertise.



8080 Programming Tutorial - Terminology

by Ward Christensen

BIT

Stands for BInary digiT, the smallest piece of information which a computer can deal with. It is usually represented as a number which can take on the value 0 or 1. It may be loosely compared with a switch, which is either open or closed. The programmer usually deals with collections of 8 BITS, called a BYTE (See BYTE).

Bits are numbered from **right** to **left**, thus the rightmost bit is numbered 0.

BOARD

Means a "circuit board", usually a **printed circuit**. (Thus the common abbreviation P.S. Board).

Some computers occupy a single board. Others, such as the APPLE or S-100 systems, consist of multiple boards.

One board may be a SERIAL PORT, another a RAM MEMORY board, another the CPU board, etc.

BREAKPOINT

When DEBUGGING a program, you often want to execute portions of it at full speed, rather than step-by-step. This may be because that part is known to be OK, or because it contains time-dependent routines, such as might be used to access a FLOPPY DISK.

A BREAKPOINT is a facility in which execution of the program will temporarily stop when a certain instruction is executed.

In the 8080, and CP/M in particular, a value of "FF hex" is used to temporarily replace the instruction.

When this instruction is executed, control transfers to address 38 Hex of the 8080; there control transfers to a routine which puts back the original instruction, and informs you that a BREAKPOINT has been reached.

BUFFER

An area of MEMORY which contains related data, such as that input from a keyboard, or to be printed (which would be referred to as 'input buffer' and 'output buffer' respectively).

BYTE

A collection of 8 BITs. A BYTE can contain one of the following:

* An unsigned number from 0 to 255
* A signed number from -128 to +127
* A single character (such as the letter 'A')

You may also consider a BYTE to be 2 HEX digits, 3 OCTAL digits, etc. For example, the ASCII character 'A', may be represented as:

'A' (character) or 01000001 (binary)
or 101 (octal)
or 64 (decimal)
or 41 (hex)

In the CP/M 8080 assemblers ASM and the macro assembler MAC, these are represented, respectively:

'A'
01000001B
101Q
64
41H

A series of BYTEs next to each other may contain a message ('END OF PROGRAM') or may contain numbers larger than can be contained in 1 BYTE. See also INSTRUCTIONS.

BUG

A BUG is simply a programming problem, which must be solved either by looking at the source program, or, by using some tool to assist stepping through the program while it is executing.

See also DEBUGGING.

BUS

Refers to the electrical and mechanical layout of the connectors in a computer, into which circuit BOARDs are plugged.

The products of a particular manufacturer typically all conform to that manufacturer's bus. Other manufacturers can then make boards which are compatible with this bus.

INTEL, Motorola, and DEC, are typical manufacturers whose microcomputer lines have a particular BUS.

Computer hobbyists most frequently come in contact with the "S-100" bus (See S-100).

CARRY

Technically, CARRY is one of the BITS in the PSW of an 8080 or other microprocessor.

As such, it "holds" a bit which is placed there by some instructions. For example, SHIFTing a register usually places one of the bits shifted, into CARRY.

Also, when doing certain arithmetic instructions, CARRY is used to indicate something about the result: On an unsigned **add**, CARRY indicates the result was too large to be held in the REGISTER.

In an unsigned compare, carry indicates the difference. I use a memory aid "CAL" meaning "**C**arry if **A**ccumulator **L**ower".

CHIP

A small "morsel" of chocolate, which enhances the flavor of certain otherwise "less interesting" cookies.

Also a common term for a "packaged", or "integrated" electronic circuit. It may refer, for example, to a MEMORY device, or to an electronic electronic circuit such as the 8080 MICROPROCESSOR itself. The 8080 'CHIP' has the REGISTERs on it, and the control logic to cause PROGRAM

execution. To have a MICROCOMPUTER system, memory, input and output devices, and a power supply are required in addition to the "Microprocessor CHIP".

CLOCK

A MICROCOMPUTER has a CLOCK to determine at what speed the electrical signals occur. It may be 1 MHz, 2 MHz, 3 MHz, or may be some "odd" number--I have seen a system running at 1.7777 MHz. (See MHz)

CLOCK may also refer to the speed of SERIAL data transfer from a TERMINAL.

CLOCK CYCLE

A CLOCK CYCLE is the shortest interval in which the microprocessor does something. Several CLOCK CYCLES are required to perform the most simple instruction. Many are required for more complex instructions.

CLOSE

When you have created a FILE on a FLOPPY DISK, CLOSE writes information back to the DIRECTORY so that you will be able to find the file for later use.

CMOS

(Not really 8080, assembler, or CP/M related, you may wish to skip this term. However, it is of common interest, and being found more frequently.)

A "family" of INTEGRATED CIRCUITS which is known for its low power consumption. Only a few microprocessors use CMOS for their circuitry, such as the RCA COSMAC series. They are ideal for battery powered operation due to their low power.

One other characteristic of CMOS devices, is that their power consumption goes up as their CLOCK speed goes up. Thus CMOS may run slowly, consuming little power, or faster, consuming more power.

CMOS stands for "Complementary Metal Oxide Semiconductor". CMOS low power results from the 'oxide", which is an insulating layer exists between the input signal and the part of the circuit being controlled. Thus, very little power flows--the effect is capacitive, not a direct connection.

COMMENTS

To be most readable, an assembler program should contain comments stating what the function of the various instructions is.

One caution to the beginning assembler programmer: As you experience the joys of learning what the instructions do, it is tempting to place comments in the program explaining what, 8080-wise, the instruction does, instead of explaining how the instruction relates to the program. For example,

```
MVI   B,7  ;SET B TO 7
```

is a bad comment. Unless specifically designed to do so, a program should not teach the instruction set. You should assume your reader has some other source, such as this tutorial, for learning the instruction set itself. Thus a better comment would be:

```
MVI   B,7  ;INIT LOOP COUNT
```

It is "excusable" to comment "how an instruction works" when using it in a non-obvious way, or if it is one of the seldom-used instructions.

COMPILER

Though not directly related to assembly programming, people sometimes confuse COMPILER, ASSEMBLERs and INTERPRETERs, since all provide alternate ways of making programs.

A compiler is typically a program which takes source programs in some **high-level** language, such as "C", COBOL or FORTRAN, and produces a runnable, OBJECT PROGRAM.

When talking about assembly language programming, it is most correct to call the program that translates the SOURCE PROGRAM into the OBJECT PROGRAM, the ASSEMBLER.

CP/M

'Control Program for Microcomputers" an operating system for 8080, 8085, and Z-80 microcomputers written by Dr. Gary Kildall, and Digital Research,Inc.

It loads and executes programs, and relevant to this tutorial, provides the environment which supports the basics for assembly language programming: an editor and assembler, and debugger.

Of interest are the following programs supplied with CP/M:

ED the editor
ASM the assembler
LOAD to load the output of ASM into a runnable form.
DDT to debug programs

C.P.U.

Stands for Central Processing Unit. This is a rather "old" term, and often referred to a 'huge" cabinet and the electronics which it held.

In this tutorial, it will typically mean the 8080 CHIP, or may refer to the circuit BOARD on which the 8080 is mounted, as in "What kind of CPU board are you running"?

DATA

Synonymous with "information". Often used as a "generic" word, instead of saying the more detailed word such as "BIT" or 'BYTE".

DEBUGGING

Bill Precht, a consultant friend of mine, once walked into a customer's office, and seeing someone working, asked "What are you doing?". The reply "Bugging". Bill asked "What do you mean?". The reply: "Well, tomorrow when this program is written, I'll be **de**-bugging, so today I must be **bugging**."

CP/M supplies a program to help debug programs, called DDT, or Dynamic Debugging Tool. It allows single stepping a program, or running it full speed, stopping only when a particular instruction is executed.

A better one, SID, is available for an additional charge. I use SID for all my DEBUGGING as it makes use of SYMBOLS produced by the ASSEMBLER. ⇒

DIRECTORY

The portion of data stored on a FLOPPY DISK which keeps track of where the various FILES on the disk are.

DISKETTE

A synonym for FLOPPY DISK. See FLOPPY DISK.

EPROM

"Erasable Programmable ReadOnly Memory". An INTEGRATED CIRCUIT (or CHIP) which stores data permanently (i.e. with the power disconnected), but which may be erased by being exposed to ultraviolet light. See PROM.

FILE

The name given to a collection of information, usually stored on some medium outside of the computer, such as cassette, or more commonly, a FLOPPY DISK.

A file may be a program, or data, such as names and addresses, financial figures, experimental results, etc.

FLOATING POINT

A means of storing numeric data where the magnitude is stored with it. Contrast this with INTEGER.

Typical microprocessor BASIC INTERPRETERS handle about 7 decimal digits, and allow the exponent of the number to be between -38 and+38. Thus, in scientific notation, the smallest number they can handle is 1 times 10 to the -38 power, and largest is 1 times 10 to the + 38 .

Some BASICs offer an extended floating point, which increases their precision from 7 decimal digits, to 14.

ASSEMBLER programs usually do not deal with floating point numbers, as (1) the hardware of most microprocessors does not provide it, and (2) subroutines to simulate floating point are not generally available.

FLOPPY DISK

A circular disk of magnetic material, enclosed in a square, usually 5.25" or 8" plastic envelope, upon which data is magnetically recorded.

Recording may be at various "densities" i.e. closeness-of packing of the BITS, and may be written on either one, or both sides.

Floppy disk capacity runs from under 100,000 BYTES for a single sided, single density 5.25" disk, to more than 1,200,000 for an 8", double sided, double density disk.

Data is stored in "tracks", i.e. concentric circles which can be accessed by the "read head" without anything moving except the disk spinning.

The tracks are divided into SECTORS, typically from 10, to as many as 58.

The "IBM Standard" single density diskette has 77 tracks with 26 sectors of 128 bytes on each, for a total capacity of just over 250K BYTES.

See also HARD DISK.

FUNCTION

A SUBROUTINE which typically acts on data passed to it, and returns some result. Ordinarily, a SUBROUTINE simply does something, such as print a line or character, write a record to a FILE, etc.

A typical FUNCTION would be one to compute the absolute value of a number, i.e. if given the value -345, would return a +345.

HARD CODED

This term typically refers to Input-Output coding, which is placed directly in a program.

In a CP/M program, it is possible to write an assembler program which accesses the disk, console, and listing device, without "hard coding" and I/O.

However, there are conceivably times when a programmer **must** 'hard code' I/O, such as for a modem, or, because of some function CP/M doesn't supply, such as STATUS of the printer (normally you can just send a character to the printer, with CP/M waiting if the printer is not ready. Only in more recent versions is there a way to **test** whether the printer is ready or not.)

HARD COPY

Refers to output, typically to a printer. Yes, you could just say "printed output" and not have to say "hard copy".

"Soft copy", a term much less used, refers to data which you "cannot put on paper", such as data being shown on a video display.

HARD DISK

Refers to magnetic disk storage, in which the disk medium is usually a rigid aluminum disk, and which typically is not removable.

Hard disks have capacities significantly beyond that of FLOPPY DISKs. For example, the highest common capacity for an 8" floppy disk is 1.2 MB, an 8" hard disk may have a 40 MB capacity. (Note: these numbers are rapidly changing as technology improves.)

HEX or HEXADECIMAL

A numbering system in base 16. It is derived from the HEX (meaning 6) and DECIMAL (meaning 10). It is much more commonly referred to as 'HEX'. It is used because of the difficulty in working with BINARY. When 4 BITS are grouped together, they are described as a HEX digit. The following table shows the relation of decimal, BINARY, OCTAL and HEX for the numbers 0-15:

DECIMAL	BINARY	OCTAL	HEX
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

HIGH-ORDER

In an 8080, an ADDRESS occupies two bytes. Thus we need a term to refer to one, or the other.

The one which contains the most significant part of the address is called the HIGH-ORDER BYTE; the one which contains the least significant is called the LOW-ORDER BYTE.

LOW and HIGH-ORDER also may refer to BITS, such as the "LOW-ORDER BIT of a BYTE".

IMMEDIATE

Some instructions manipulate data which resides in REGISTERs or MEMORY. Others make use of a value which is "immediately at hand", right in the instruction.

This type of instruction is called an IMMEDIATE instruction.

The IMMEDIATE DATA in 8080 INSTRUCTIONS may be either 8 or 16 BITS.

INDEX REGISTER

A REGISTER, which contains some value, which is added to the value in some other REGISTER, to obtain the ADDRESS of an item of DATA to be processed.

The Z-80, and some other microprocessors, contain "true" INDEX REGISTERs, i.e. you may change the value or "displacement" stored in the INDEX REGISTER, and keep the "base" value the same.

For example, suppose you have a table of 50 1-byte values, and you want to step through that table. You can set a "base" register to the table, then use an INDEX REGISTER to indicate what 'displacement" into the table you are currently processing.

At any time, you can set the INDEX REGISTER to 0, and will thus be pointing back to the beginning of the table.

The 8080 does **not** have a "true" index register, i.e. such that by placing 0 in it, you will be pointing to the "beginning of something".

Instead, what are **called** index registers, might better be called BASE registers, as they point somewhere, but when you have changed them, there is no systematic way of returning them to their original value, without an explicit "load" instruction.

I have not found this to be any hardship: Index registers **are** very useful in large computers, where you frequently deal with **fixed length** data, such as 80 column cards, or fixed length disk records.

In that case, you want to be able to quickly go some **fixed** displacement into the record, so index registers are handy.

In the 8080 however, data is more frequently known not by its **position** in a record, but by the **context** of the data, such as being separated by a comma from the item preceeding and following it. Thus you might "scan" the record, skipping three commas, to get to the fourth field, rather than going to "column 42" to find the start of the field.

INPUT/OUTPUT (I/O)

To do useful work, a computer must (1) input data; (2) process that data; and (3) output the data. Thus the abbreviation I/O is frequently used when talking about the devices or programs which get data **into** and **out of** the computer.

INTEGER

An item of numeric data, which cannot take on fractional values. It may or may not have a sign.

There are several examples under the term BYTE. However, an integer may also be longer than a BYTE, and typically in assembly language programming, is 16 bits.

A 16 bit integer can contain an unsigned value from 0 to 65,535, or signed values from -32768 to +32767.

INTERPRETER

Since I have mentioned ASSEMBLER and COMPILER, the definition of INTERPRETER will round out the 3 common types of programs used to process source language programs into a runnable form.

An INTERPRETER is a program which "looks at" your source program, and does what you tell it to do, immediately.

Contrast this with an ASSEMBLER or COMPILER, which **processes** your source program, producing a runnable OBJECT PROGRAM.

The advantage of interpreters is that they allow errors to be detected, and quickly corrected, without having to leave the INTERPRETER "environment".

The disadvantages of interpreters are that (1) they typically take quite a bit of room, limiting the minimum sized program you could effectively run; (2) they are the slowest means of executing programs, since they have to execute machine instructions "on behalf" of what you want done, instead of "directly executing" machine instructions generated by an assembler or compiler.

(There are exceptions: The language FORTH is interpreted, yet is still quite compact and fast. It suffers from you having to "help" it by translating programs into "reverse-Polish", as used by most of the HP calculators.)

INSTRUCTIONS

The 8080 microprocessor reads INSTRUCTIONs from MEMORY and takes action based upon what the INSTRUCTION is. Instructions may be 1, 2, or 3 BYTEs long. They may be as simple as one which increments the value in a REGISTER, or as complex as one which calls a SUBROUTINE based on some condition.

I have broken down the 8080 instructions into various categories which will be individually covered in the tutorial:

* Data movement-between registers and between registers and memory
* Arithmetic-only add, subtract, increment and decrement (The 8080 has no multiply or divide instructions –you must use subroutines). ⇒

* Stack instructions
* Execution control
* Input-Output
* "Other" (not covered above)

INTEGRATED CIRCUIT or I.C.

An electronic circuit combining many transistors, resistors. The MICROPROCESSOR CHIP is an IC, as are the TTL CHIPs used in MICROCOMPUTERS.

INTERRUPT

Refers to hardware which signals the MICROPROCESSOR that some external event has happened. It is called an INTERRUPT because it interrupts whatever program is executing. Con trol is then transferred, via a specific ADDRESS, to a special routine.

This routine typically saves the registers, executes some specific function, then restores the registers and returns to the program which was interrupted.

For example, you could have a keyboard which INTERRUPTs the 8080 whenever a key is pressed. The INTERRUPT handling program would store the character read into a BUFFER, and might take some special action if the character keyed was a carriage return (end of line). Most hobbyist microcomputers do not use interrupts, unless they are 'factory built in' such as in the HEATH H-8, or the Polymorphics 8080 microcomputer.

The reason interrupts are seldom used, is that they add complexity to the system, and, in a single user system, typically do not significantly increase performance.

For example, for keyboard in put without interrupts, you just test a BIT read from an INPUT PORT to see when a key has been pressed. Another use for interrupts might be to maintain the time of day (some hardware device interrupts the MICROCOMPUTER every second, and the program then increments a counter).

K

'K" is the abbreviation for KILO. It means, not 1000, but rather two to the 10th power, or 1024, of "something". For example, a "16K" RAM CHIP means it has 16x1024 or 16384 BITS.

A MEMORY board, which has 16K means it has 16384 BYTES.

Sometimes "K" is used to mean speed, as in a 9.6K baud terminal, which (oops) means 9600 (i.e. NOT 9 x 1024). (Nobody said this microcomputer area is consistent.)

LOW-ORDER

(See HIGH-ORDER)

MACRO

In ASSEMBLY PROGRAMMING, a means of coding one INSTRUCTION which in turn **generates** others.

This makes programming easier, since where you might have to code, for example, 9 instructions in CP/M to OPEN a file, a MACRO could generate those 9 instructions for you, when you code:

```
CPM   OPEN,MYFILE
```

MB

Abbreviation for Mega Byte, i.e. one million bytes. For example, a 10 MB HARD DISK contains 10 "MEG" bytes. See also MEG.

MB may also be an abbreviation for the speed, in BITs per second, at which a particular device operates, such as a HARD DISK.

MEG

A shortening of "MEGA" meaning 1,000,000. However, since computers are based on the BINARY numbering system, MEGA usually refers to, not 1000 x 1000, but rather 1024 x 1024, or 1048576.

MEMORY

The part of a computer where the PROGRAM, and data, are stored. MEMORY is divided into BYTES, each of which has an ADDRESS. The amount of MEMORY is frequently described by saying how many 1024-BYTE blocks you have, each 1024 being called a K (for Kilo). Thus a computer with 8K has 8 x 1024, or 8192 BYTES. The 8080 is capable of directly ADDRESSing 64K BYTES of MEMORY. This value is 65,536 decimal, and sometimes is thus wrongly called 65K.

MEMORY MAPPED

When transferring data between a microprocessor and a particular device, you typically address it by its PORT. However, some devices may be addressed as though they were MEMORY installed in the computer. Computers such as the 6800 and 6502, do not have PORTS, i.e. they do not have electrical lines to signify that data is being sent to or requested from anything other than memory.

Thus, whenever an input or output device is referenced as though it were memory, it is said to be MEMORY MAPPED. It may be as simple as an address from which you read some switches, or may be one or two K bytes, representing a "memory mapped" video screen. To "out put" data to such a screen, you just store data in the particular one or two K portion of memory allocated to the screen.

MHz

An abbreviation for "Mega Hertz", or "millions of cycles per second". It is called 'Hertz" to honor one of the early pioneers, just as we have done with "Watt", "Ohm", "Volt", and others.

A typical 8080 CPU runs at 2 MHZ, which means it is capable of doing the "most simple" thing, 2,000,000 times per second. By "most simple", I mean something like "access memory", 'add 2 numbers", etc.

The minimum number of these "most simple" things it can do at one time is 4. For example, adding two BYTEs together, which are both in REGISTERS, takes 4 cycles, or 1/500,000 of a second. The reason for taking 4, is that it has to "fetch" the

instruction from memory, "decode" what that instruction says to do, "do" the actual operation, and then prepare to do the next one. The longest 8080 instruction takes 17 cycles to run.

Common usage of MHz is to say "how fast is a given C.P.U.". For example, a 4MHz Z-80 (*) will execute instructions two times as fast as an 8080. (Actually even a bit faster, since the Z-80 executes some instructions in 4 cycles that the 8080 takes 5 to do).

MICROCOMPUTER

A computer based on a MICRO PROCESSOR CHIP. A typical microcomputer contains a power supply, MEMORY, and input-output devices such as a keyboard, and a display, or a teletype. In addition, a MICROCOMPUTER might have a cassette or floppy disk interface, a modem for phone communications, etc.

MICROPROCESSOR

An "integrated" circuit which provides the arithmetic and logical, addressing and control functions, basic to a computer. With the addition of a power supply, memory and usually devices to input and output data, a MICROPROCESSOR becomes a MICROCOMPUTER.

See also "CHIP"

MNEMONIC

2, 3, or 4 letters grouped together to represent an INSTRUCTION in the 8080. In this tutorial, the phrase 'OP CODE' will be used instead.

An example is "MVI" which stands for "move immediate". It is followed by the OPERANDS, which consist of (1) the register the data is to be moved into, and (2) the value to be moved.

MODEM

Stands for MOdulator/DEModulator, i.e. a way of taking digital data ("BITS") and sending them on a "medium" such as telephone lines, which is not directly capable of handling them.

The modulator changes the data on the transmitting end, into electrical tones, and, on the receiving end the demodulator changes them back into digital form.

NOT

Just as in "plain English", NOT means to "negate" the meaning of something.

In programming, NOT means the same thing, i.e. "IF A NOT EQUAL TO B THEN ..." (a valid portion of a COBOL program).

It may also mean to "flip" a bit, from 0 to 1, or from 1 to 0.

It most frequently is applied to hardware, where it means to change an electrical signal's meaning from TRUE to FALSE, or from FALSE to TRUE.

Note: Here we end this month's installment of Ward Christensen's excellent tutorial. You'll want to save these terms for reference in the future.

A Note On SBASIC

A bug has been reported which affects sequential file I/O. If a sequential file channel is opened for input or output, that channel will be available only for input or output throughout the program, whether the file has been closed and re-opened or not.

The fix is to open other channels under the same filename to use for input or output.

There are a few other problems users sometimes have with SBASIC. These are not bugs. When random files are used people frequently forget that the first record number is 0, not 1. Writing to a file with record number "1" as the first record will produce "Read/Write pas eor" error messages. When reading or writing binary strings, many users forget a that a length byte precedes each string, so that a string of length five actually requires 6 bytes of storage in the buffer or on the disk.

"On error goto" statements cannot be placed inside the bodies of procedures or functions. This statement can only be used in the mainline program.

ZSID Application Note

This application note is copyrighted by Digital Research, Inc. and is reprinted here with permission of Digital Research, Inc.

The restart instruction TRACE and HIST use for setting breakpoints and tracing program execution can be chaged by modifying the restart vector location in TRACE.UTL and HIST.UTL.

The following procedure shows how to change the breakpoint vector address to 0040H.

```
A› REN TRACE.LTU=TRACE.UTL
A› ZSID TRACE.LTU
ZSID VERS 1.4
NEXT  PC  END
0600 0100 nnnn
#S268
0268 39 40
0269 00 .
#S2B8
02B8 3A 41
02B9 00 .
#G0

A› SAVE 5 TRACE.UTL
.
.
.
A› REN HIST.LTU=HIST.UTL
A› ZSID HIST.LTU
ZSID VERS. 1.4
NEXT  PC  END
0600 0100 nnnn
#S27C
027C 39 40
027D 00 .
#G0

A› SAVE 5 HIST.UTL
```

Assembly Language Interface to PL/I-80 ..... Part 2

by Mike Karas

The following short file is a single statement PL/I-80 declare statement that defines in PL/I-80 syntax, all entry points that are given in the preceding assembly language program. The file is used so that the same set of declared variable names can be used in multiple PL/I-80 source programs without having to type them in all the time, risking errors or omissions. The PL/I-80 language contains a feature called the "%include" statement that allows the following file to be put in as part of the source code as simple as typing the following code as part of your program:

```
%include 'bioscall.dcl';
```

This will insert the following text into the program source code at the point where the %include appears. Note that in the above syntax the file 'BIOSCALL.DCL" is expected to be contained upon the same disk drive as the source program. Explanation of the statement formats of the declared entry points will be left for the examples below. An example of the "%include" function appears in the disk copy program below.

```
File "BIOSCALL.DCL"
/*Define names and procedure characteristics for the*/
/*MICRO RESOURCES Direct BIOS Access Procedures of */
/*'PLIBIOS.ASM" and PL/I-80 linkable module */
/*'PLIBIOS.REL' */
dcl
        cboot       entry,
        wboot       entry,
        cstat       entry       returns (bit(8)),
        conin       entry       returns (char(1)),
        conout      entry       (char(1)),
        list        entry       ( char(1)),
        punch       entry       (char(1)),
        reader      entry       returns (char(1)),
        home        entry,
        seldsk      entry       (binary fixed(7)),
        settrk      entry       (binary fixed(15)),
        setdma      entry       (ptr),
        read        entry       returns (bin fixed(7)),
        write       entry       (bin fixed(7))
                                  returns (bin fixed(7)),
        lstat       entry       returns (bit(8)),
        sectran     entry       (bin fixed(15))
                                  returns (bin fixed(15));
```

An example of how to make use of the assembly language BIOS access routines is presented below in the form of a PL/I-80 source program that is a floppy disk copy program. First let me describe the intent of the program and then show a few examples of how the external BIOS entry points are accessed. The diskette copy program allows full copying of a diskette in Drive A: to the diskette in Drive B:. Tracks are buffered in memory in track buffer arrays so that a whole track may be read at a time to make the program run reasonably fast. (Incidentally, this copy operation runs almost as fast as most assembly language copy utility programs I've seen except for those that buffer more that one track at a time.) Tracks are read sequentially from the HOME position of Drive A: and written to the corresponding positions of Drive B:. After writing, the Track is reread from Drive B: and compared byte for byte with the image read in originally from Drive A:. Any errors are reported as to track and sector number to inform the operator of any difficulties encountered along the way. At the completion of the copy process the operator is prompted to see if another copy is desired before reshooting the system disk in Drive A:.

All console interface and program logic structure is implemented in the easy to program PL/I-80 syntax. Speed dependent and hardware specific I/O access in the program makes use of the BIOS access external entry points. The following examples show some of the external procedure references made to the entry points defined in "PLIBIOS". In the following paragraphs some familiarity with the PL/I-80 language structure is assumed. A tutorial on the structure and syntax of the language is somewhat beyond the intended subject of this article.

EXAMPLE ONE. Calling The CONSOLE INPUT ENTRY POINT (CONIN)

The console input entry point is declared as an external entry point in the BIOSCALL.DCL include file as follows:

```
DCL conin   entry returns(char(1));
```

This defines conin as a function procedure with procedure invocation by name in an expression. The implicit CALL to the routine will return the declared type of value, CHAR(1) in this case, to the point of the call. The example from the disk copy program has the conin call at the point of waiting for an operator response. A character string, declared as follows:

```
DCL resp   char(80)   /* operator response buf */ ;
```

is designated to receive one console character with the following statement. The NULL parentheses reference designates conin as a function procedure with no passed parameters. In this case resp

```
resp = conin();
```

becomes a string of one entered character padded on the right with 79 blanks.

File dskcpy.pli

Full disk copy program to demonstrate the direct CP/M BIOS access facilities presented in the accompanying article, 'Assembly Language Interface to PL/I-80.' Track by track copying is utilized through the direct BIOS access facilities of the linked assembly language program 'PLIBIOS'.


```
dskcpy:
     proc options(main);

     %replace
          trk—per—disk   by     77, /* logical bios tracks/diskette */
          sec—per—trk    by     26; /* logical cp/m sectors/track */

     /* external CP/M bios entry points */
     %include 'bioscall.dcl';       /* bring in the bios entry point
                                       definition file that defines
                                       bios function entry points*/
     dcl
          resp                          char(80), /* operator response buf */
          inbuf(sec—per—trk)              char(128)
                                           based(p), /* input buffer */
          outbuf(sec—per—trk)             char(128)
                                           based(q), /* output buffer */
          trkno
          secno                         bin fixed(15),
          (i,j,k,l)                     bin fixed(15),
          (p,q,s)                       pointer,
          e5var                         bit(8) based(s),
          e5char                         char(1) based(s),
          e5trk                         bit(1),
          skew(sec—per—trk)             bin fixed(15) static init(
                                        1,7,13,19,25,5,11,17,23,3,09,15,21,
                                        2,8,14,20,26,6,12,18,24,4,10,16,22),
          e5string                      char(128),
          partab                        pointer; /* disk parameter table ptr */


     /* Print initial message for Diskette copy from drive A: to B: */

     put edit('MICRO RESOURCES Diskette Copy program in PL/I-80 ',
          '^m^j',
           'Version 1.1 of August 3,1981')
           (col(1),3(a));

copyloop:
     put skip edit('Insert Source Diskette in Drive A:',
             '^iDepress ‹cr› when ready ')(col(1),a);
     resp=conin();

newdisk:

     put skip(1) edit('Insert Destination Diskette in Drive B:',
               '^iDepress ‹cr› when ready or ‹ctl-C› to quit')
                (col(1),a);
     resp=conin();
     if resp='^C' then do;
               put edit('Remember to Re-Insert your CP/M ',
                     'System Diskette in Drive A:',
                     '^iDepress‹cr› when ready ')
                    (col(1),a,a);
               resp=conin();
               stop;
               end;
```

RENEW

Did your subscription begin in November of 1980? If so, you have received a letter from us about renewing your subscription. Don't put off renewing or you'll miss out on our upcoming issue.

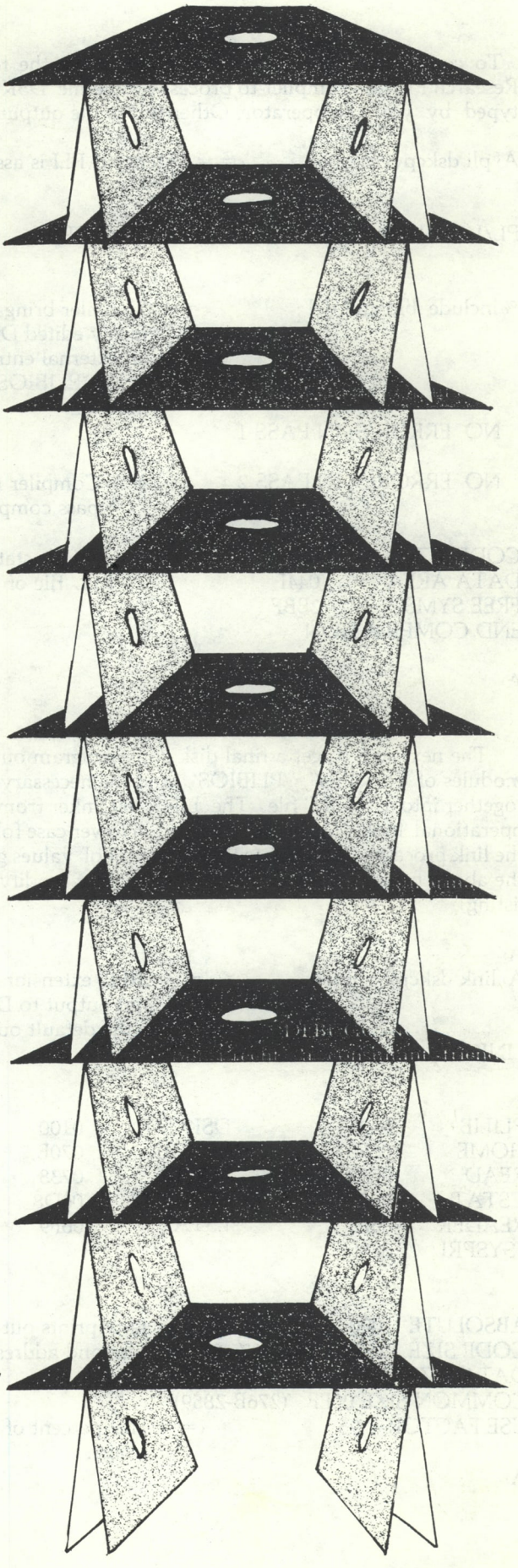
Rumor Has It...

...that a new Spanish version of WordStar is in the works, and that French and German versions may also be planned.

...that the report generator for DataStar files will probably be released at the end of the year.

```
      put skip(1) edit('Copy in Progress...')(col(1),a);

      allocate inbuf set(p);  /* setup buffer storage */
      allocate outbuf set(q);
      allocate e5var set(s);        /* setup simple 0EH insertion */

      partab=seldsk(0);             /* select disk a: (0) */
      call home;                    /* restore it */
      partab=seldsk(1);             /* select disk b: (1) */
      call home;                    /* restore that too */

      e5trk='0'b;                   /* e5 end track scan switch off */
      e5var='e5'b4;

      do i=1 to 128;                /* fill e5 record check string */
      substr(e5string,i,1)=e5char;
      end;

trklp:
      do trkno = 0 to trk—per—disk-1 while(^e5trk);

      partab=seldsk(0);             /* select source */
      call settrk(trkno);

      j=0;                          /* set empty sector counter off */

      /* read source track loop */

            do secno = 1 to sec—per—trk;
            call setdma(addr(inbuf(secno)));
            call setsec(skew(secno));
                        /* sectors are zero based     */
                        /* for some CP/M 2.2 BIOS's   */
                        /* if so replace (secno) with */
                        /* (secno-1)                  */

            if read()^= 0 then do;
                  put edit('Read Error On Control Disk - Track
                        trkno,' Sector ',secno)
                        (col(1),a,f(3),a,f(3));
                  put skip(2) edit('Restart Copy ',
                        'Process with new Source Diskette.')
                        (col(1),a,a);
                  go to copyloop;
                  end;
            end;

      partab=seldsk(1);             /* select destination */
      call settrk(trkno);

      /* write destination track loop */

            do secno = 1 to sec—per—trk;

            call setdma(addr(inbuf(secno)));
            call setsec(skew(secno));
                        /* sectors are zero based     */
                        /* for some CP/M 2.2 BIOS's   */
                        /* if so replace (secno) with */
                        /* (secno-1)                  */
            if write(2)^= 0 then do;
                  put edit ('Write Error On Destination',
                        ' Disk - Track ',trkno,
                        ' Sector ',secno)
                        (col(1),a,a,f(3),a,f(3));
```

```
                put skip(2) edit ('Bad Diskette in Drive ',
                      'B:, try another.')(col(1),a,a);
                 go to newdisk;
                 end;
      end;

/* read back destination loop */

      do secno = 1 to sec—per—trk;
      call setdma(addr(outbuf(secno)));
      call setsec(skew(secno));
                        /* sectors are zero based     */
                        /* for some CP/M 2.2 BIOS's   */
                        /* if so replace (secno) with */
                        /* (secno-1)                  */

      if read()^= 0 then do;
                 put edit ('Read Verify Error On Destination',
                      ' Disk - Track ',trkno,
                      ' Sector ',secno)
                      (col(1),a,a,f(3),a,f(3));
                 put skip(2) edit ('Bad Diskette in Drive ',
                      'B:, try another.')(col(1),a,a);
                 go to newdisk;
                 end;
      end;

/* data compare loop */

      do secno = 1 to sec—per—trk;

      if inbuf(secno)  = outbuf(secno) then do;
                 put edit ('Verify Data Compare Error ',
                      trkno)
                      (col(1),a,f(3));
                 put skip(2) edit ('Bad Data on Diskette ',
                      'in Drive B:, retry same diskette.')
                      (col(1),a,a);
                 go to newdisk;
                 end;
      if inbuf(secno) = e5string then j=j+1; /* empty sector */
      end;

if j=sec—per—trk then do;
           e5trk='1'b;
           put edit ('Track by track copy complete ',
                'at track number ',trkno)
                (col(1),a,a,f(3));
           end;
end trklp;

free inbuf;
free outbuf;
free e5var;

put skip edit('Do you wish to Copy This Another Diskette (Y/N) ')
     (col(1),a);
resp=conin();
if (resp = 'Y') ! (resp = 'y') then go to newdisk;
                     else stop;
end dskcpy;
```



To compile the PL/I-80 program given above the following operational sequence would be used to invoke the Digital Research PL/I-80 compiler to process source file 'DSKCPY.PLI'. Lower case characters after the 'A› ' prompt are those typed by you the operator. Other text is the output of the compiler to the console screen.

```
A› pli dskcpy‹ cr ›          ‹== Extent of .PLI is assumed

PL/I-80  V1.3  COMPILATION OF:  DSKCPY

%include 'bioscall.dcl';     ‹== Compiler brings in prev-
                                 iously edited DCL file for
                                 all external entry points
                                 in the 'PLIBIOS.REL'
                                 module.
  NO  ERROR(S) IN PASS 1

  NO  ERROR(S) IN PASS 2     ‹== Compiler informs us of
                                 pass completions.

CODE SIZE   =   05A9         ‹== Compile statistics for
DATA AREA   =   044F             'REL' file of 'DSKCPY'.
FREE SYMS   =   2BBF
END COMPILATION

A›
```

The next step to get a final disk copy program out of all of this work is to link the 'REL' (relocatable object code) modules of 'DSKCPY', 'PLIBIOS', and the necessary subroutines from the Digital Research PL/I-80 run time library together into a '.COM' file. The LINK 80 linker from Digital Research does the job nicely with the following console operational 'look'. As before, the text in lower case following the A» prompt was that typed by the operator performing the link process. (Please note that the symbol values given may not necessarily coincide with those for your version of the above program; especially if you slightly modify the text strings and sign on messages in the PL/I-80 source code listing).

```
A›
A› link dskcpy,plibios       ‹== '.REL' extensions assumed
                                 and output to DSKCPY.REL
                                 is the default output.
LINK 1.3

PLILIB    RQST    DSKCPY    0100    CONIN     06CC    SELDSK    0705
HOME      06FF    SETTRK    070E    SETDMA    0720    SETSEC    0717
READ      0732    WRITE     0738    CBOOT     06BA    WBOOT     06C0
CSTAT     06C6    CONOUT    06D8    LIST      06E1    PUNCH     06EA
READER    06F3    LSTAT     06F9    SECTRA    0729    /SYSIN/   27E7
/SYSPRI/  280C

ABSOLUTE   0000              ‹== Link prints out module
CODE SIZE  266B  (0100-276A)     size and address statistics
DATA SIZE  06E2  (285A-2F3B)
COMMON SIZE 00EF  (276B-2859)
USE FACTOR  95               ‹== Hex percent of symbol table
                                 usage.
A›
```

The rest is up to you. I have provided an example of how assembly language routines can be tied to PL/I-80 to implement special functions that are beyond the scope or capability of the language. In this case they were simple examples of interface conversions for BIOS access. Other examples of assembly language routines that could be made include "IN and OUT" functions for direct language access to machine dependent input and output ports, high speed special access drivers for math chips such as the Intel 8087, or tightly coded interrupt service routines to increase program performance where PL/I-80 is used as the host development language in a real-time application.

Operation of the above example should be self-evident from a detailed study of the source listing. The best way to tackle your own problem is to dive right in. For those who feel that they need a little more structured approach or partly implemented feel for getting started, I am willing to provide the source code for all of the modules described in this article on a single density CP/M compatible 8 inch diskette free of charge if you just send a diskette, mailer, and return postage to the address given at the beginning of PL/I program listing. Feel free to make use of any new ideas you see here however you see fit.

KIB-BITZ

ABBS, CBBS, FBBS, RBBS, ETC., Maybe you'd like to start one too? ??????????????????????????

by Jim Mills

In this article I am going to attempt to tell you how to start a CBBS™ or "software swapboard" (CBBS «» software swapboard). I am limiting myself to Ward and Randy's CBBS since I used to run one, and to the RBBS-type of software swapboard system, since I have that software available to me. Perhaps someone familiar with an ABBS, a FORUM-80, or another system will write an article for a future issue. So much for objectives.

A Little History: CBBS

In February of 1978, only two years (and some odd months) after MITS Altair introduced the first "home computer," two computer hobbyists in the Chicago Area started the first microcomputer Computerized Bulletin Board System, or CBBS. Those two hobbyists are Ward Christensen and Randy Suess, who still operate their CBBS at 312-545-8086 (like the number?). It should be noted that the terms "Computerized Bulletin Board System" and "CBBS" are trademarked by Ward and Randy, a fact of which many seem to be ignorant.

Originally planned in January of 1978, the CBBS was at first meant to be a gathering place for the Chicago Area Computer Hobbyist Exchange (CACHE) newsletter articles. That idea was soon dropped and a messages-only system resulted. Ward and Randy have considered going to a software swapboard type of system, but the current CBBS is overly busy as it exists now. Software swapboards take significantly more time per caller than a message-only system. Also, a substantial amount of software rewriting would need to be done to convert CBBS to MP/M or some other time-sharing system, not to mention the hardware expense. Another alternative would be to go to a system using "slave" computers to answer the phones, each running CP/M and CBBS, and a "networked" master computer for the file handling (MP/M and CP/NET). Some changes to CBBS software would still be required, and the hardware costs require a wealthy owner.

A Little History: RBBS

RBBS (Remote Bulletin Board System?) is actually derived from RIBBS. RIBBS is a BASIC program written by Bruce Ratoff of the Amateur Computer Group of New Jersey (ACGNJ) and used under CP/M as a message program.

ALL software swapboards (that I have seen, anyway) hinge upon a program which ties the phone-answering modem to the console, usually with both the local console and the modem doing simultaneous console I/O (input/output). The original program for doing this was written by Dave Jaffe, formerly of CACHE, now somewhere in San Francisco hiding out from "BYE" users (just kidding, Dave!). BYE is the name of that original program, now greatly modified by many people.

How BYE Works

In CP/M, all transient programs load and execute at an address of 0100H, the TPA, or transient program area. Well, what BYE does is load into this area from disk, check to see if BYE is already running in upper memory (says "GOODBYE, CALL AGAIN" if so), and, if BYE isn't running, boots itself up to the top of memory (usually the top of memory above the CP/M O.S.) so that you can run OTHER programs at 0100H. BYE resides in upper memory and intercepts console I/O for your modem. Disk I/O, printer I/O, etcetera remain unaffected. Fairly simple to explain, not so simple to program.

Starting a CBBS

If you are planning to run a message-only system, such as CBBS, you needn't use BYE, although using BYE may save you having to chop up your CP/M BIOS (Basic Input Output System).

I have talked to Ward Christensen about running CBBS under BYE, and there is one drawback. If a caller "hangs up" while the CBBS is still doing file I/O, some files may not be closed or updated as is the case during normal CBBS usage (where the modem I/O is handled in the BIOS). Normally, the way Ward and Randy suggest you set up your I/O, the CBBS will detect the loss of carrier, do all the appropriate file I/O, and hang up the phone. But, if you run CBBS under BYE, BYE will detect the loss of carrier, hang up the phone, and wait for the next call despite the fact that CBBS may be running in lower memory with files still not updated. So, what does all this mean? Well, if you want to run CBBS under BYE as part of a software swapsystem, or just as a method of handling modem I/O, you will have to modify BYE (and maybe CBBS) so that BYE will know if CBBS is running and NOT hang-up, but let CBBS hang up, and maybe CBBS should test to see if BYE is running in order to set a switch in BYE so that BYE will KNOW that CBBS is running ... whew! That's a mouthful! As far as BYE goes, I think it also ought to be made relocatable (like DDT or SID) so that it will run in a full 64K system (at the top of the TPA). Maybe I'll tackle that project at some time in the future, enough sidetracking for now, back to the main discussion...

Normally you will have to add the modem I/O routines to your BIOS so that CBBS will talk to your modem as well as the local console. This is done in the CONIN, CONOUT and CONSTAT routines of the BIOS. It may be

simpler to modify and assemble BYE if you are not a "hacker" when it comes to 8080 assembler language. (I'll tell you later what a hacker is -if you don't know, you aren't). If you tell your users to wait for the CBBS's "END OF CONNECTION" message before hanging up, you shouldn't have any errors.

If you wish to use Ward and Randy's CBBS package, send fifty dollars to Randy, payable to Randy Suess, at the following address: 5219 Warwick, Chicago, IL 60641. You should also send a copy of the license agreement form which may be obtained from Ward and Randy's CBBS. See messages 9, 10 and 11. If you don't send the form, Randy will most likely send you the form to fill out before he sends the software. The CBBS software is provided on two disks, in 8080 assembler source code with numerous documentation files and utilities. The CBBS software is extremely modularized and is assembled with LINKASM, which is included. There are several modem I/O files, which brings us to the subject of what hardware is required for CBBS.

Note that the source code is 8080 assembler -this should tell you that you have to have an 8080, 8085, or Z80 processor chip, as well as CP/M, which CBBS runs under.

Well, you need a computer, of course, preferably one that doesn't object to 24-hour usage. You will need a modem that is capable of automatically answering the telephone. I recommend the MM-103 from Potomac Micro Magic in Virginia, but the DC Hayes and IDS work well, as do many "separate" modems that are made to be able to answer the phone, but usually cost more due to extra packaging, power supplies, etc. (The PMMI, DC HAYES, and IDS modems drop right into an S-100 slot).

Of course, you need a phone line for the CBBS, usually one that is a 24-hour computer line, although some folks run it part time. A clock board, such as the Sci-tronics is nice, but not an absolute requirement. So much for hardware.

Let's assume you now have a computer with a phone-answering modem, you have the CBBS software in front of you, and you are ready to make it leap over tall buildings, etc. Well, first and foremost, read the documentation files. (That's like reading the instructions before assembling the toy). Believe me, it'll help, unless your name is Ward Christensen. In fact, I recommend printing out all the .DOC and .ASM files before you do anything else!

Ok, you got them all printed out, and you've read all the documentation. Now what? Well, you should have a fair idea from the .DOC files, but the next thing is to start modifying the .ASM files to match up with your equipment. Do you use a 4 Megahertz Z-80? Do you use a PMMI, IDS, DC Hayes, or another modem? There are assembly language options for all of these variables, mostly true/false switches. If you use a PMMI, make sure the base address is the same as yours in the file CBBSPMMI.ASM so that the CBBS will find your modem where it expects it to be. You will want to modify two places in CBBSWORK.ASM & CBBSDISK.ASM to insert your name and phone number(s) in place of Ward and Randy's. You will want to tell the CBBS program whether or not to log to a teletype, or if you have a clock board. Then you're ready to assemble the program, which will take about 6 or 7 minutes on a 2 MHz 8080 using LINKASM, and another ½ minute or so to load using LOAD.COM. You may wish to set the 'TEST' switch to true and try the CBBS out without the modem I/O, just to make sure everything is OK before going on-line.

Now you have a "ready to use" CBBS program. How about I/O? The CBBS expects to find the modem I/O along with the console I/O. You can write up a special BIOS for CP/M, or you can use BYE. Use of BYE requires the computer to be running all the time, unless your CP/M is set-up to do "BYE /A" as the initial command. CP/M can boot-up a program on cold boot, if you do a start-up and reset when the phone rings. See the .DOC files that come with CBBS for more explanation if you are interested. Also, the PMMI MM-103 comes with a drawing showing you how to build a circuit that will turn your computer on when the phone rings. If you use the latter method, and if you want to run only CBBS, you must modify BYE to bring up a specified program after it loads, or you must have the modem I/O in your BIOS.

If you've gotten this far, you're ready to try it. If you don't have a second computer (or terminal with acoustic coupler) to call in on, then you'll have to have a friend try to dial it up. If it doesn't work perfectly, it's just like trouble shooting any other program -- take it a step at a time, and good luck.

Starting a Software Swapboard

Actually, starting a swapsystem is easier, in some ways, than starting a CBBS. In a CBBS the user calling in on the modem cannot get to CP/M command level -he is stuck in CBBS. On a swapsystem, you want him to be at CP/M's command level so that he can look at the programs that are on-line and "modem' programs to or from your system. You don't want him to be able to get your copyrighted software though! There are several ways to do this. One is to DDT your CP/M .COM file, find the commands in the CCP (look for DIR, TYPE, SAVE, etc) and modify them so that the user can't do a "USER" command, for example. Then, you place .COM files on disk that are named DIR.COM, TYPE.COM, etc. These types of files are usually made system files so that they are invisible to the user. It is assumed that the user will have some familiarity with the CP/M operating system. Usually the DIR.COM program will be SD.COM (in real life, like Clark Kent and you-know-who). Similarly TYPE.COM will really be MLIST.COM.

Ok, you've kept the user away from the normal CP/M CCP commands and put substitutes in their places as .COM files on disk. What next? BYE is next. You must make modifications to BYE.ASM to make it work on your system. This includes leaving enough memory above your CP/M for BYE to fit into, as it runs outside of the TPA. Once BYE is running properly, you're on-line! There are some other planning considerations, such as whether or not you plan to have a message-board such as CBBS and disk space. Disk space? Yes! Swapboard users want all kinds of free software. That's what swapboards are all about! That means you must have a lot of disk space. I have checked into some swapboards running on two five-inch drives, but

⇒

most swapboards use two or more eight-inch double density double sided boards. If you format with a 1K sector size that gives you about 1.2 Megabytes of storage per drive. If you have a hard disk drive (and money, incidentally) then you have it made. You can put all the CPMUG disks online on your hard disk, right? It better be a big disk! The CP/M Users Group is now up to 54 volumes. And you better leave room for growth -the group is constantly growing.

What most sysops (system operators) do is put on-disk the most popular and current files, and put special files on by request (with usually a day or so notice). This seems to work out fairly well, but it has been my experience that the most popular swapboards are the ones with the double-sided double-density 8-inch drives or hard disk drives (that means the most busy lines also, but it comes with the good service, unfortunately.).

Well, I hope that this article has helped some of you. Most of the documentation is pretty self-explanatory with the software that comes with these systems (CBBS, RIBBS, etc.), so you shouldn't have to much trouble. If you can't hack it, get a "hacker" to help you. Now you know what a "hacker" is!

The Dentist's Office: PAS-3 and Univair

by Tom Crites

Problems

The basic problem with a dentist's office, as with any office, is productivity. If an office wants to grow and expand it has to either increase its staff or increase the efficiency of its existing personnel. Looking at current trends: productivity on the decline and wages increasing—hiring new employees is not always the best solution.

Solutions

A major consideration should be buying a computer system. With the cost of computers on the decline, what was **not** cost justifiable a few years ago **is** very possible today. With a computer, an office can increase productivity without increasing its staff, by freeing up existing employees' time and allowing them to work on more important tasks. More important than the computer itself (hardware) is the software, or programs, which make the machine perform particular tasks; store data, recall data, process data, etc. Today, with hardware coming down in price and most vendors supplying very similar hardware, it is not as important a decision as it has been in the past. Today one must look closely at the application programs that are available and make sure what is purchased is exactly what is desired.

What a Dental Management program should provide

A good dental management program should eventually reduce the work load in an office. This is accomplished by streamlining the handling and processing of all the pertinent patient information, such as billing, insurance, and scheduling. As a by product of maintaining this information a multitude of reports can be generated: analysis of service reports, producer reports, and delinquency reports, to name a few. These reports are important in bettering the management of an office.

The files

All of the data of the dental management system is stored in a collection of files that are maintained by the programs making up the system. From looking at and reviewing these files in a system the user can get a good idea of what information must be entered into the system, how that information is stored and what reports can be generated from the data. Below is a list of the major files maintained by a dental management system.

☐ Patient file-common data each patient.

☐ Dependent file-information on a responsible party's dependents.

☐ Insurance file-insurance information on various patients.

☐ Code file-codes and standard fees for particular procedures.

☐ Diagnostic code file-descriptions of diagnostic codes.

☐ Doctor file-information on each doctor in the practice.

☐ Analysis file-information required to generate analysis of services reports (i.e. time, service, amount, doctor, etc).

☐ Billing file-information on how the billing forms are to be generated.

☐ Insurance form file-once an insurance form has been laid out it is stored in this file and then referred to as a number.

☐ Recall file-contains the recall information on various patients.

The programs

The files themselves are important, but even more important are the programs to access and easily maintain these files. These programs must be well written with the end user in mind. They should allow a smooth flow through the system; from when the system is turned on to the particular function the user wants to perform.

Procedure (Data entry, update and retrieval)

Entering Initial data:

As with any program such as this the initial glamour of the system is tarnished by the fact the the data base must first be entered. This includes entering the chart of service codes and standard fees, doctor data, billing forms, the insurance forms, and finally entering all the patient data into the computer. This is a very time consuming task but when completed, it is well worth the effort.

An important difference between the two packages, which is important to all dentists, is in the area of printing

insurance forms. UNIVAIR has the standard ADA insurance form pre-programmed into the system. If it is necessary to run any other insurance forms you must contact your representative, (however multiple forms can be recalled from the system). With the PAS-3 system the insurance forms that are needed are laid out, programmed into the system, and then referred to a number (standard ADA insurance form is form number 1, Blue Cross-form number 2, etc.). With this system any number of insurance forms may be entered into the system by the end user and then simply recalled by a number. Handling insurance forms this way is time consuming in the beginning, but in the long run it saves time, and makes the system much more versatile.

Daily Data entry

The simplicity of data entry is very important with any system, but it must also have data screening and error catching routines to maintain the integrity of the data base. The major difference between the PAS-3 and the UNIVAIR systems is in the way that the patient accounts are set up. In the PAS-3 system each account number is set up with a responsible party. Any dependents the responsible party might have are tagged onto this number. This technique is quick and efficient, because in most cases when the patient is first entered into the system, the billing and insurance information is entered once for the responsible party, with any dependent also using this information. However this system does have a disadvantage in that all charges and payments to a responsible party's account are put in one balance forward account, not credited or debited to each dependent. This can cause problems if, for insurance reasons, an office has to keep track of what payments are for which dependent. UNIVAIR handles each individual in the system with a separate account number. This solves the problem of what payment gets credited to which account. However it also creates considerable overhead, on having to maintain separate account numbers, addresses, insurance information, etc. on each individual.

Maintaining data

The system must also have a way to maintain the data base once it is entered. This function is provided through a series of utility programs which are used to change and update information once it has been entered into the computer (delete, add, or change patients, service codes, diagnostic codes, etc.). With the UNIVAIR system, if a patient's record must be updated and their account number is not known, a separate search program can be run to find it by keying in the name and the edit program is run using the account number for a reference. With the PAS-3 system, any patient record may be called up for reviewing or editing by either the patient name or number. This may seem trivial, but in the long run could save time from jumping from one program to another.

The reports (Generating reports)

Below is a summary of reports available with each of the systems.

PAS-3 System

- ☐ Daily
  - Charges & Receipts
  - New Accounts
  - Producer report
- ☐ Monthly
  - Delinquency Reports
  - Account receivable aging
  - Insurance Billing
  - Regular Billing
- ☐ Special Reports
  - Analysis of services
  - Dump of all patients on disk
  - Query Program

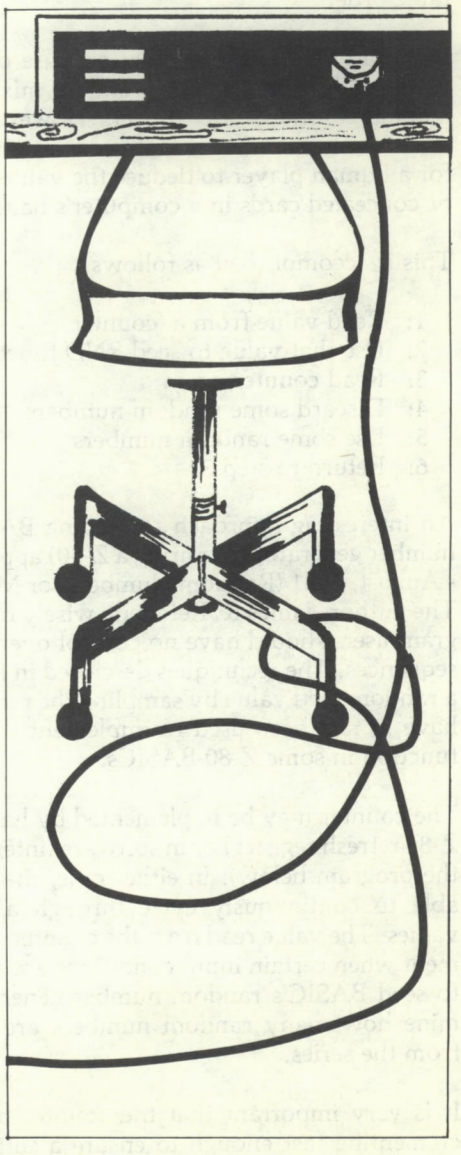
UNIVAIR

- ☐ Patient master-numeric or alphabetic
- ☐ Patient number listing
- ☐ Patient scheduling
- ☐ Ticket register-by clinic or doctor
- ☐ Payment register-by clinic or doctor
- ☐ Print monthly statement
- ☐ Print mailing labels
- ☐ ADA procedure report-by clinic or doctor
- ☐ Trial balance-detailed or summary
- ☐ Aged trial balance
- ☐ Print insurance forms

The reports available with the PAS-3 and the UNIVAIR systems are similar. The processing of a report in either system for the most part consists of selecting a report number and then hitting the appropriate key.

Overview

In general I would say both of these packages are well written. I have spoken with end users of both systems and from all indications, the programs work well. For all practical purposes, they both supply the same end product; a series of functions and reports that aid the dentist in the management of his office. Each system goes about this process a little bit differently. My advice to anyone interested in purchasing a dental management program is to first get the manuals of any software packages being considered and go over them in detail. The quality of the manual might give you an indication of the quality of the software, and by comparing the manuals you will get a better idea of which system is particularly suited to your needs and present office procedure.



Better Random Numbers

by Bill Burton

Editor's note: This article was received before our last issue was mailed. It is for this reason only that the author has not acknowledged the random number seeding program, submitted by Bob Kowitt, *Lifelines*, September, '81.

Many computer simulations of actual random events fail because they rely solely on sequential calls to pseudo-random number tables. Simulated results produced by this method are not truly random and can eventually be predicted.

I will describe by which a simulated deck of cards can be shuffled adequately in BASIC. I chose card shuffling because it requires that a reasonably large sequence of values (52 for a single deck) be rearranged in unpredictable order.

It should be understood that any use of pseudo-random values will produce a less random mix than a thorough manual shuffle of a real deck. However, this limitation may be minimized so that it becomes virtually impossible for a human player to deduce the values of undealt cards or concealed cards in a computer's hand.

This is accomplished as follows:

1: Read value from a 'counter'
2: Use that value to seed RND function.
3: Read counter again
4: Discard some random numbers
5: Use some random numbers
6: Return to step 3:

An interesting approach to seeding BASIC-80's random number generator (requiring a Z-80) appeared in **Lifelines** - August, 1981 (Random Numbers for Microsoft BASIC). The author, James R. Reinders, wisely observed that program users should have no control over random number sequences. The techniques described in that article derive a random seed value by sampling the refresh register, and have in fact been used to implement the RANDOMIZE function in some Z-80 BASICs.

The counter may be implemented by hardware (as in the Z-80 refresh register) or in software (integer variable 'T' in the program below). In either case, the counter must be able to continuously cycle through a preset range of values. The value read from the counter at the precise moment when certain input conditions are met is used either to seed BASIC's random number generator or to determine how many random numbers are to be discarded from the series.

It is very important that the counter be capable of incrementing fast enough to ensure a sufficient variety of possible values. Hardware counters or those implemented in machine language will easily satisfy this condition. If the counter is a BASIC variable, it should begin to increment whenever a new screen is displayed. The few seconds required to read the screen will allow enough delay for the counter to cycle to an unpredictable value.

The following program was excerpted from a poker simulation which I am writing. It shuffles a single deck and deals two five card [draw poker] hands.

```
100 '***** DEMO PROGRAM - WRITTEN FOR BASIC-80
110   DEFINT A, I, J, K, L, T
120   DIM A(52), V$(13), S$(4), H$(10), C$(10)
130   O$ = " OF " : T=0
140   FALSE%=T : TRUE%=NOT FALSE%
150     FOR I=1 TO 52
160       A(I)=I
170     NEXT I
180       FOR I=1 TO 13
190         READ V$(I)
200       NEXT I
210         FOR I=1 TO 4
220           READ S$(I)
230         NEXT I
240 '
250 '*****  SEED RND FUNCTION
260   PRINT "ENTER 'B' TO BEGIN    ";
270     WHILE E$<>"B" AND E$<>"b"
280       E$=INKEY$ : T=T+3
290         IF T>256 THEN T=T-256
300     WEND
310   RANDOMIZE(T)
320 '
330 '***** MAIN PROGRAM DRIVER
340     WHILE TRUE%
350       GOSUB 410
360       GOSUB 540
370       GOSUB 620
380     WEND
390 '
400 '***** DISCARD SOME NUMBERS
410   PRINT : PRINT : E$=""
420   PRINT "ENTER 'D' TO DEAL     ";
430     WHILE E$<>"D" AND E$<>"d"
440       E$=INKEY$ : T=T+3
450         IF T>128 THEN T=T-128
460     WEND
470       FOR I=1 TO T
480         X=RND
490       NEXT I
500   PRINT : PRINT : PRINT : PRINT
510   RETURN
520 '
530 '***** SINGLE DECK SHUFFLE
540   FOR I=1 TO 52
550     J=INT(RND*52)+1
560       IF I=J THEN 550
570     SWAP A(I), A(J)
580   NEXT I
590   RETURN
600 '
610 '***** DEAL TWO FIVE CARD HANDS
```

```
620   L=0 : PRINT "PLAYER'S HAND";
630   PRINT TAB(30) "COMPUTER'S HAND"
640   PRINT
650     FOR I=1 TO 9 STEP 2
660       L=L+1
670       C$(L)="" : H$(L)=""
680       K1=A(I) : K4=A(I+1)
690       K2=K1 MOD 13 + 1 : K3=K1 MOD 4 + 1
700       K5=K4 MOD 13 + 1 : K6=K4 MOD 4 + 1
710       C$(L)=C$(L)+V$(K2)+O$+S$(K3)
720       H$(L)=H$(L)+V$(K5)+O$+S$(K6)
730       PRINT C$(L);
740       PRINT TAB(30) H$(L)
750     NEXT I
760   RETURN
770 '
780 '***** DATA AREA
790 DATA "ACE  ", "DEUCE", "THREE", "FOUR "
800 DATA "FIVE ", "SIX  ", "SEVEN", "EIGHT"
810 DATA "NINE ", "TEN  ", "JACK ", "QUEEN"
820 DATA "KING ", "HEARTS", "DIAMONDS"
830 DATA "SPADES", "CLUBS"
840 END
```

The routines of principal interest do the following:

1:Seed RND function (lines 270-310)
:Discard some random values (lines 430-490)
3:Shuffle the deck (lines 540-580)

The first two of these use BASIC-80's INKEY$ function to test key
entry. The value of variable T continues to cycle until a specified key ('B' for seed value and 'D' for new deal) is pressed. CBASIC users may use the CONCHAR% function to the same effect. Note: In a fully implemented simulation, a reasonable delay (perhaps to read a screen) should be forced immediately before lines 260 and 340. Thereafter, the time required to read the hands displayed on the screen will provide enough delay.

Acceptable ranges for the variable T are dictated by lines 290 and 450. These values work quite well with the BASIC-80 interpreter running on a 4Mhz Z-80 machine. These lines may be adjusted as needed for other environments. One note of caution is in order; the BASIC-80 interpreter produces extremely long pseudo-random series (I have tested beyond a million without repetition). Other BASICs may require additional code to reseed the random number generator if there is any reasonable chance of exhausting the series during play.

In general, the most efficient simulations of single deck shuffles share these common traits:

1: The deck is represented by a single numeric array.
2: The position of each card must change at least once.
3: The position of any card may change more than once.

Any computer shuffling technique must reorder an imaginary deck but this is not the only requirement. When an actual deck is shuffled, any card might reappear in its initial position. Some commonly used shuffling algorithms mistakenly include checks which prevent this possibility.

Imagine the cards in a standard deck occupying 'positions' 1-52; (positions 1 and 52 are the top and bottom of the deck respectively). One possible arrangement of the topmost three cards might be:

1: SIX OF DIAMONDS
2: FOUR OF CLUBS
3: ACE OF CLUBS

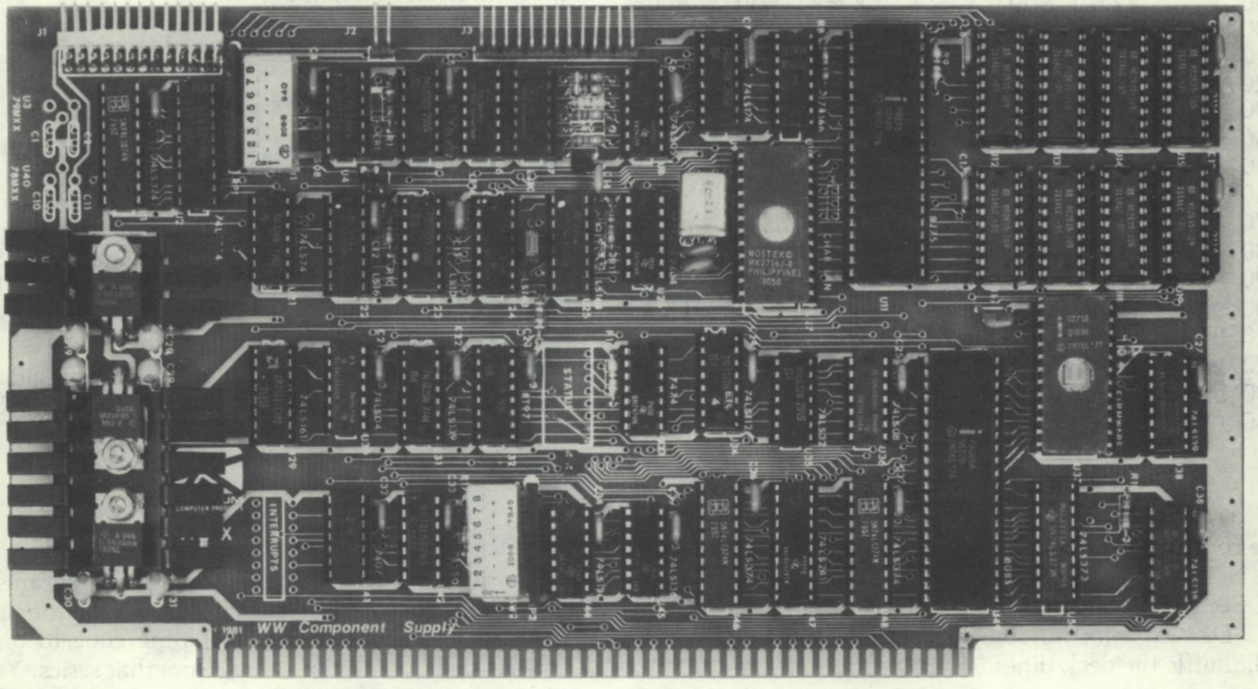
A proper computer shuffle should reflect the one chance in fifty two that the Six of Diamonds will return to position 1. A realistic card mix should also allow each of the top three cards to be shuffled to their original order in positions 1-3. (This should happen only very rarely, the odds against are more than 13000 to 1).

The algorithm in lines 540-580 is simple and satisfies the requirements for a realistic shuffle. Users of BASICs without the SWAP command may recode line 570 as follows:

```
570 L=A(I) : A(I)=A(J) : A(J)=L
```

In conclusion, I would suggest that any programs which simulate random events using pseudo-random number series will be improved by including code to discard a time-related quantity of values from that series. You may wish to enter the above program to verify its efficiency (under 1 second at 4Mhz to shuffle and deal two hands).

INTELLIGENT VIDEO I/O FOR S-100 BUS



VIO-X

The VIO-X Video I/O Interface for the S-100 bus provides features equal to most intelligent terminals both efficiently and economically. It allows the use of standard keyboards and CRT monitors in conjunction with existing hardware and software. It will operate with no additional overhead in S-100 systems regardless of processor or system speed.

Through the use of the Intel 8275 CRT controller with an onboard 8085 processor and 4k memory, the VIO-X interface operates independently of the host system and communicates via two ports, thus eliminating the need for host memory space. The screen display rate is effectively 80,000 baud.

The VIO-X1 provides an 80 character by 25 line format (24 lines plus status line) using a 5 × 7 character set in a 7 × 10 dot matrix to display the full upper and lower case ASCII alphanumeric 96 printable character set (including true descenders) with 32 special characters for escape and control characters. An optional 2732 character generator is available which allows an alternate 7 × 10 contiguous graphics character set.

The VIO-X2 also offers an 80 character by 25 line format but uses a 7 × 7 character set in a 9 × 10 dot matrix allowing high-resolution characters to be used. This model also includes expanded firmware for block mode editing and light pen location. Contiguous graphics characters are not supported.

Both models support a full set of control characters and escape sequences, including controls for video attributes, cursor location and positioning, cursor toggle, and scroll speed. An onboard Real Time Clock (RTC) is displayed in the status line and may be read or set from the host system. A checksum test is performed on power-up on the firmware EPROM.

Video attributes provided by the 8275 in the VIO-X include:

- FLASH CHARACTER
- INVERSE CHARACTER
- UNDERLINE CHARACTER or
- ALT. CHARACTER SET
- DIM CHARACTER

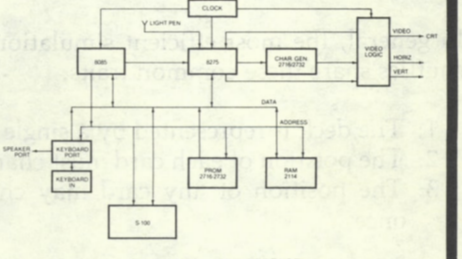
The above functions may be toggled together or separately.

The board may be addressed at any port pair in the IEEE 696 (S-100) host system. Status and data ports may be swapped if necessary. Inputs are provided for parallel keyboard and for light pen as well as an output for audio signalling. The interrupt structure is completely compatible with Digital Research's MP/M ®

Additional features include:

- HIGH SPEED OPERATION
- PORT MAPPED IEEE S-100 INTERFACE
- FORWARD/REVERSE SCROLL or
- PROTECTED SCREEN FIELDS
- CONVERSATIONAL or BLOCK MODE (opt)
- INTERRUPT OPERATION
- CUSTOM CHARACTER SET
- CONTROL CHARACTERS
- ESCAPE CHARACTER COMMANDS
- INTELLIGENT TERMINAL EMULATION
- TWO PAGE SCREEN MEMORY

VIO-X1 – 80 × 25 5 ×7 A & T **$295.00**
Conversational Mode
VIO-X2 – 80 × 25 7 × 7 A & T **$345.00**
Conversational & Block Modes



VIO-X S-100 I/O INTERFACE

FULCRUM™ COMPUTER PRODUCTS

Distributed by:
WW COMPONENT SUPPLY INC. 1771 JUNCTION AVENUE • SAN JOSE, CA 95112 • (408) 295-7171

CPMUG Volumes 53 and 54: Catalogues and Abstracts

Volume 53

DESCRIPTION: BDS-C Users group 'original' Adventure disk. This is the 'original' Crowther/Woods adventure, converted into BDS-C by Jay R. Jaeger, and updated for V 1.43 by L.C Calhoun. We thank Robert Ward, and the BDS-C Users Group for this entire volume contribution.

NUMBER	SIZE	NAME	COMMENTS
		-CATALOG.053	CONTENTS OF CP/M VOL. 053
53.1	1K	ADVCOMP.SUB	These files are
53.2	5K	ADVENT.C	fully explained
53.3	10K	ADVENT.COM	in"CATALOG.DOC"
53.4	5K	ADVENT.CRL	and appear here
53.5	6K	ADVENT.H	merely to get
53.6	18K	ADVENT1.DAT	"official numbers".
53.7	7K	ADVENT2.DAT	
53.8	7K	ADVENT3.DAT	
53.9	4K	ADVENT4.DAT	
53.10	6K	ADVENT5.DAT	
53.11	18K	ADVENT6.DAT	
53.12	1K	ADVLINK.SUB	
53.13	2K	CATALOG.DOC	
53.14	7K	DATABASE.C	
53.15	5K	DATABASE.CRL	
53.16	1K	EADVENT.C	
53.17	24K	EADVENT.COM	
53.18	1K	EADVENT.CRL	
53.19	3K	ENGLISH.C	
53.20	2K	ENGLISH.CRL	
53.21	3K	ENVIRON.DOC	
53.22	4K	ITVERB.C	
53.23	3K	ITVERB.CRL	
53.24	1K	MKADVENT.SUB	
53.25	1K	MKEADVEN.SUB	
53.26	1K	MKSAVEAD.SUB	
53.27	1K	READ.ME	
53.28	1K	SAVEADV.C	
53.29	7K	SAVEADV.COM	
53.30	2K	SAVEADV.CRL	
53.31	14K	TURN.C	
53.32	10K	TURN.CRL	
53.33	13K	VERB.C	
53.34	8K	VERB.CRL	
		CRCK.COM	Program to check CRC of each file.
		CRCKLIST.CRC	List of CRC's for all files.

If you suspect that your copy of this disk may be bad, run CRCK *.* and compare the CRC's against the ones in CRCLIST.CRC. If you want to create a new file, use CRCK *.* F but be aware it will overwrite CRCKLIST.CRC.

Volume 54 □□□□□□□□□□□□□□□□□□□□

Description: Xitan Disk Basic:
1. Games
2. CAI programs

Many of the games found here were contributed to the CPMUG by William P. Ruf of Kansas. The files were reviewed and abstracted by Jim Kennedy of CACHE (the Chicago Area Computer Hobbyist Exchange), who also made some corrections to some programs, or, in some cases, pointed out the known bugs in some of these programs for users to de-bug. Jim also contributed his own 'typing drill' program, TDRILL.BAS. These programs are written (or modified) for Xitan Disk Basic, but some will run under MBASIC without modification, and some require only slight modification.

Jim Mills, CPMUG reviewer

NUMBER	SIZE	NAME	COMMENTS
		-CATALOG.054	CONTENTS OF CP/M VOL. 54.
		ABSTRACT.054	Abstracts of programs.
		CRCK.COM	CRC filecheck program.
		CRCKLIST.054	List of CRC's of files.
54.1	3K	1CHECK.BAS	Solitaire checker puzzle.
54.2	2K	ALFABET2.BAS	Interactive alphabetizing, de-bugged and expanded.
54.3	1K	ALFABETI.BAS	Original bugged version.
54.4	2K	ARITH.BAS	Simple addition and subtraction for elementary students.
54.5	4K	BIOCAL.BAS	Biorythmic calendar (Bugs).
54.6	8K	BLKJAC.BAS	Blackjack (21) game.
54.7	3K	BOMBER.BAS	WW2 Bomber game.
54.8	2K	BOUNCE.BAS	Plots a bouncing ball.
54.9	7K	BUG.BAS	"Draw Bugs faster than your computer' game.
54.10	2K	BULCOW.BAS	Buggy program, number guessing game.
54.11	7K	BUNNY.BAS	Draws a "bunny" on your CRT.
54.12	3K	BUZZWD.BAS	Selects a list of "buzzwords".
54.13	4K	CHASE.BAS	High Voltage survival game.
54.14	6K	CHASE2.BAS	Not related to above game. Construct and maneuver in a maze.
54.15	3K	CHOMP.BAS	Eat pieces of a cookie, last piece loses (NIM?).
54.16	3K	CRAPS.BAS	Standard Nevada table rules.
54.17	5K	CUBE.BAS	Get thru the cube & win a bet.
54.18	3K	DEFUSE.BAS	Find and defuse the bomb before...
54.19	1K	DIAMND.BAS	Fills screen with diamond shapes that spell DEC.
54.20	3K	DRAW.BAS	Buggy program -- see abstract.
54.21	2K	DRINKS.BAS	"How to mix drinks", see abstract.
54.22	3K	FISHING.BAS	Catch fish in a lake, avoid hazards, a mini-adventure game.
54.23	9K	FOOTBL.BAS	Standard professional rules, except no penalties.
54.24	2K	FRACT.BAS	Fraction mathematics.
54.25	1K	GRAFIT.BAS	Some kind of student plotting program.
54.26	2K	GUNNER.BAS	Fire a field artillery weapon, bugs.
54.27	6K	HOCKEY.BAS	For hockey fans.
54.28	4K	HORSES.BAS	Place your bets on the horse races.

Abstracts

Volume 53

CPMUG is not providing abstracts for this disk; the description with the catalogue should be sufficient.

Volume 54

This volume contains programs intended to be used with TDL-XITAN DISK BASIC. Many of these programs are modified versions of public domain programs which have been distributed by DECUS, the DIGITAL EQUIPMENT COMPUTER USERS SOCIETY. DECUS requires programs submitted to it for distribution be in the public domain. Many of these programs were originally written by high school students and other individuals who wrote them and submitted them to a users group so that they could be shared with others.

Back in 1971, a gentleman who worked for DIGITAL, helped in a project to collect donated programs, and published them in a book called 101 BASIC COMPUTER GAMES. His name is David H. Ahl. Mr. Ahl left DEC in 1974, and asked for the rights to print the book independently. They agreed as long as the name was changed. He revised many of the programs, added some, removed some, and published a book he called BASIC COMPUTER GAMES, MICROCOMPUTER EDITION. He copyrighted the book. Meanwhile, DEC gave the original programs to DECUS, who distributed them on DECtape and Magtape to the computer community, essentially putting them in the public domain. It is this group of programs, (and NOT the revisions copyrighted by *Creative Computing*) that have been translated into TDL-XITAN BASIC and submitted to CPMUG for this set of disks. Many of these programs were received by The CP/M Users Group in XITAN internal code. They have been translated back into ASCII for distribution, so it is possible that some of them will work on other BASICS, such as Microsoft BASIC. However, they have been checked out only with TDL-XITAN Disk BASIC for these abstracts. Some of these programs do have bugs, some of which are noted in these abstracts. The bugged programs are included so that some CP/M users who like challenges will have the opportunity to try their skills on finding and correcting these bugs.

Jim Kennedy

Editor's Note: The actual abstracts for this volume are somewhat more detailed than the descriptions below. But we think these outlines will give you a good overview of the games on this volume.

1CHECK.BAS
Solitaire checker puzzle by David Ahl, p. 163-164 of 101 BASIC Computer Games (published by Digital Equipment Corporation). Works fine, but it would be nice if the number grid were repeated at each move.

ALFABETI.BAS
Author unknown. A nice little interactive alphabetizing program, which works OK on the first run, but hangs up with a Re-Dimensioned array @ line 70 error when you try to enter a second list of items. SEE ALFABET2.BAS.

ALFABET2.BAS
This is the ALFABETI.BAS program revised by Jim Kennedy, to correct problems in the original, and give the user more detailed instructions. The program interactively accepts a list of words, names, etc., and prints an alphabetized list. Note: use all caps or all small letters in your items. Otherwise, items in caps will be listed before lower-case items in the alphabetized list.

ARITH.BAS
Presents simple addition problems. You give the answer, it tells you if you are right or wrong. If wrong, you get to try a second time. If still wrong, you are given the answer and then given a new problem. Good for elementary students for drill and practice in addition, and as an illustration of some CAI programming techniques.

BIOCAL.BAS
Biorhythmic calendar. Will NOT run on my XITAN BASIC. % after end numbers is OK on some Microsoft and Digital BASICs, but not on XITAN. Also, I think there should be more code after line 1430. Perhaps someone can find the source of this program and fix it up.

BLKJAC.BAS
P. 39-41 of 101 BASIC Computer Games. This version of Blackjack is the one written and modified by DIGITAL personnel, originally for RSTS-11. It runs well on XITAN DISK BASIC. It is very comprehensive and fun to play. (It takes a long time to load as an ASCII file.)

BOMBER.BAS
P. 45-46 in 101 BASIC Computer Games. Originally written by David Sherman of Curtis Junior High School, Sudbury, MA., and later modified by DIGITAL personnel. It has been since additionally modified to run on XITAN BASIC. You can make some decisions, but your fate is largely up to the random number generator. It runs well, and was exciting for the children in the family for the first few times they ran it.

BOUNCE.BAS
This program plots a bouncing ball. Written by Val Skalabrin, and found on P. 47 of 101 BASIC Computer Games. The program takes a long time to print out if you choose figures requiring a tall plot. Try .1 sec, 25 FPS velocity and a coefficient of .9 to start. These figures will keep the plot on a 25 line TDL-VDB screen.

BUG.BAS
P. 52-54 of 101 BASIC Computer Games. The object of this game is to finish your drawing of a bug before the computer finishes its drawing. Written in the early 70s by a 7th grade student, Brian Leibowitz. The computer rolls the dice each turn and the operator needs only to type yes (or return) or no to the question concerning the bug pictures.

BULCOW.BAS
P. 55-56 of 101 BASIC Computer Games. This program does NOT work properly in XITAN BASIC.

BUNNY.BAS
By Goodyear Atomic Co., Piketon, Ohio. Submitted to DECUS 30 July 1973. This program sends a picture of a bunny head to the printer. It has obviously been re-written for XITAN

⇒

DISK BASIC, and it works well.

BUZZWD.BAS
P. 63-64 of 101 BASIC Computer Games. Written by Tom Kloos of the Oregon Museum of Science and Industry. It prepares sets of "buzzwords" by selecting words from 3 lists and putting them together. Runs OK.

CHASE.BAS
"You are within the walls of a high voltage maze—your only chance for survival is to maneuver each intercepter into a high voltage area."

CHASE2.BAS
This is a chase program written by Michael P. Ruf on 12/16/78. It asks the player for the width, the length, and the density of a maze. It then tries to construct a maze (which comes out on my screen as a column of symbols), and then asks for players' moves— to be entered as numbers. Try this program on your system...it may work with your terminal configuration.

CHOMP.BAS
P. 78-79 of 101 BASIC Computer Games. Submitted to DIGITAL by Peter Sessions of People's Computer Company, based on the game of CHOMP (Scientific American, Jan. 1973).

CRAPS.BAS
P. 83-84 of 101 BASIC Computer Games. Modified to work on XITAN BASIC. Original author unknown. his version is based on the standard Nevada craps table rules. Fun to play.

CUBE.BAS
P. 85-86 of 101 BASIC Computer Games. Written by Jerimac Ratliff of Ft. Worth, Texas, and converted to RSTS/E by David Ahl. You progress along a cube from coordinate 1,1,1 to 3,3,3. You may be zapped by a land mine along the way, but if you make it you win a wager you made when you started and are richer. You have a chance of quitting and keeping your winnings, or wagering all or part of your money on your next trip. This game is good for teaching the meaning of a 3 dimension coordinate system to youngsters.

DRAW.BAS1
I CAN'T LOAD THIS ONE WITH XITAN BASIC. I GET A MISSING STATEMENT NUMBER ERROR.

DEFUSE.BAS
Written by Tom Karzes, Curtis Jr. High School, Sudbury MA. and modified by Dave Ahl, DIGITAL. (NOT in 101 BASIC Computer Games). You are in a large building, 100 rooms long, 100 rooms wide, and 100 rooms high and are looking for a bomb. You have a bomb strength meter to guide you to the bomb before it goes off. Everytime I played the game I got blown up.

DIAMOND.BAS
P. 87-88 of 101 BASIC Computer Games. Fills the screen (or an 8 ½ X ll piece of paper if you change some print statements to PRINT#2,) with diamond shapes each containing the letters DEC!!! You can control the size of the diamond shapes. This sort of program could easily be modified to produce other patterns.

DRINKS.BAS
"This program prepares drink recipes guaranteed to make your next party a 'smashing' success." The program runs as is, and would be fun to show off at a party—but don't let anybody take it seriously.

FISHING.BAS
You are at a dock at the northwest corner of a square 8X8 unit lake. You are to move your boat through the lake and return to the dock with your catch by responding to the move question with a compass direction N,S,E,W, or F for staying in a Fixed position. If you type B, the game will begin again. You must return to the dock within 6 (computer) hours or half your catch will spoil. If you hit the shore of the lake, you will be grounded and sunk. There are other hazzards to make the game exciting.

FOOTBL.BAS
P. 101-103 in 101 BASIC Computer Games. This simulation of the game of football uses standard professional rules except that there are no penalties. This game is fun to play once you have memorized the numbers representing the different plays.

FRACT.BAS
Author: Michael Ruf. Written August 29, 1979. This seems to be a well written program dealing with fraction end math. However, it uses a WAIT statement that causes my system to hang up. It may work with a serial terminal accessing the correct port(s), or perhaps someone familiar with the WAIT command could modify it for his system.

GRAFIT.BAS
It runs, but I don't know its use—any ideas?

GUNNER.BAS
Written by Tom Kloss of the Oregon Museum of Science and Industry, and modified by David Ahl, DIGITAL. "This computer demonstration stimulates the results of firing a field artillery weapon." The game works as a game, but the formula must be wrong for computing the trajectory, because some results are not realistic.

HOCKEY.BAS
P. 130-132 in 101 BASIC Computer Games. Written by Charles Buttrey of Eaglebrook School, Deerfield, MA. and submitted to DIGITAL by Mrs. Kingsley Norris. Converted from Brand X to DIGITAL RSTS/E by David Ahl. Instructions are easier than those of the football game, but the play does not seem as interesting—unless you are a real hockey fan.

HORSES.BAS
P. 133-134 of 101 BASIC Computer Games. Author unknown. From DIGITAL. You are given the odds for each horse and you place your bets for win, place or show. After the race had been run, you are told your winnings (or losses) and invited to press your luck further.

INTEREST.BAS
A simple program to calculate simple and compound interest over a 10 year period. Works fine. Author unknown.

KING.BAS
P. 138-140 of 101 BASIC Computer Games. Author: James A. Storer, Lexington High School, modified by Dave Ahl, DEC. Available from DECUS, where it is called "Pollution Game". One of the more comprehensive, difficult and interesting land and resource management games. It runs fine on XITAN BASIC and is a lot of fun to play. This copy often terminates play after 1 year, but that is probably because I am not a very

good King. (Or the program could have a small bug).

LITQZ.BAS
P. 150 in 101 BASIC Computer Games. A simple CAI type program which presents four multiple choice questions from children's literature —illustrates simple CAI techniques in BASIC. Questions could be changed and program expanded for other instructional objectives.

MATH.BAS
Author not known. Program won't run on my XITAN system with its TDL VDB board. The WAIT at line 504 hangs it up. Perhaps the program would work with a system having a serial terminal using ports 72 and 73.

MUGWMP.BAS
P. 156-157 in 101 BASIC Computer Games. Originally written by the students of Bud Valente of Project SOLO at the University of Pittsburgh, this program was slightly modified by Bob Albrecht of People's Computer Company and converted to DEC's RSTS/E by David Ahl when he worked for Digital. The object of the game is to locate four mugwmps hiding on various squares of a 10 x 10 grid. Good practice in triangulation techniques.

PICTUR.BAS
Not in 101 BASIC Computer Games, but I have seen it in the DECUS library. It asks for your name and where you want your picture. You can have it displayed on the terminal by typing KB: or on the printer by typing LP: The program shows a technique for switching output from the video screen to the printer without having to re-write or edit your print statements.

POET.BAS
P. 171 of 101 BASIC Computer Games. Original author unknown. Modified and reworked by Jim Bailey, Peggy Ewing, and Dave Ahl of DIGITAL. This program produces random verse made of phrases suggestive of Edgar Allen Poe.

POKER.BAS
P. 172-174 in 101 BASIC Computer Games. Written by A. Christopher Hall, submitted to DECUS by A. E. Sapega. You play draw poker with the computer as your opponent.

PRIME.BAS
"This program prints the prime numbers from 1 to 10,000" (if you can wait that long).

QUBIC.BAS
P. 175-177 Original author unknown.Was on a GE timesharing system in 1968. Now in DECUS library. Qubic is the game of tic tac toe in a 4X4X4 cube. Considerably more difficult than standard two dimensional tic tac toe. This version has been improved over the one in 101 BASIC Computer Games by the addition of a provision to display the board on command. Be patient¢the computer sometimes takes many seconds to determine its move.

REVRSE.BAS
P. 180-181 in 101 BASIC Computer Games. Written by Bob Albrecht, People's Computer Co. The game requires you to arrange a list of numbers in numerical order from left to right by reversing numbers.

ROCKET.BAS
P. 184 of 101 BASIC Computer Games. Written by Jim Storer, Lexington H.S. Converted from Focal to EDUSYSTEM 30 BASIC by David Ahl, DIGITAL. This version has been considerably further modified to work with XITAN BASIC. Fun to play, but the lb. fuel remaining indicator seems to be set too high, causing you to run out of fuel and crash when you think you have enough fuel to make it.

ROCKT1.BAS
P. 185 in 101 BASIC Computer Games. Written by Eric Peters of DIGITAL. One of many variations of the lunar lander idea. This one plots a graph of your descent.

SNOOPY.BAS
This "SNOOPY" allows you to enter your name, and it then prints the Snoopy and puts your name under it.

SPORTS.BAS
Simple CAI program to give you multiple choice questions on sports. If you guess the right answer, you are congratulated. If you guess wrong, you are given the right answer.

STARS.BAS
By Bob Albrecht of People's Computer Co. A number guessing game.

STOCK.BAS
P. 209-211 in 101 BASIC Computer Games. A stock market simulation game revised 8/18/70 by D. Pessel, L. Braun, and C. Losik, Huntington Computer Project, SUNY. Lets you lose money on the stock market without having to pay out any money—other than the small fortune you spend in upgrading and maintaining your micro.

TAKEAWAY.BAS
By Michael Ruf and Rick Mack. The game seems to be well written, but some of the coding, probably intended to do something with an intelligent terminal, gives confusing output on my TDL-VDB. The game has several players alternately taking away asterisks from an asterisk collection... something like the game of CHOMP.

TDRILL.BAS
Author: Jim Kennedy. Date written: July 14, 1981 This is a simple typing drill program designed to give a beginning typist extra drill and practice. Rather than generate the letters and words with a random letter generator, as is done in some other typing drill programs, this one uses data lists to supply the letter groups. This gives more control over which letters are used in the early lessons when only a few keys have been introduced. The student may correct a mistake by backspacing if it is noticed before the CR has been hit, and the mistake is not noted. However, if a line is not perfect when the CR is hit, the student will be told so, and will be asked to try again. At the end of each 12 line drill (6 lines X 2), the student will be told how many lines were typed to get the dozen correct. Another drill can then be picked, or the student can terminate the program, and receive a count of the number of lines right and the total number typed. This program will be used in the future with a different data list to continue this lesson series, starting with lesson 13.

TENNIS.BAS
A Mult. choice quiz on tennis, with a question technique similar to SPORTS.BAS.

TEXT.BAS
An introductory information program about some of these (and other) programs. ⇒

TICTAC.BAS
This is an expanded tic tac toe game played on a 9 X 9 grid. You enter your moves as grid coordinates and the computer (often "thinking" for over a minute) prints the position of your move and its move. You have to get 5 across, down, or diagonal to win. For those with the patience to wait between moves, this game could be an enjoyable challenge. It is also good for grid coordinate practice.

TRAP.BAS
P. 224-225 in 101 BASIC Computer Games. Written by Steve Ullman, and modified by Bob Albrecht of People's Computer Company. Another "guess the mystery number" game.

TTTOE.BAS
Tic Tac Toe game. Usually plays well, but periodically gives error message: Subscript out of range on line 84, usually when no one is going to win. This one too, may benefit from some user de-bugging and feedback.

TVPLOT.BAS
Originally written in FOCAL by Mary Cole and converted to BASIC-PLUS by Dave Ahl. This program produces various funny TV plots. Good for a laugh or two.

TYPING.BAS
Author not mentioned, but it could have been written by Michael Ruf. It uses the WAIT on line 80 that hangs up my system. It may work with a serial terminal that uses ports 72(status) and 73. It is intended as a typing drill program, with the character strings generated by a random generator. Rather than correct this program, I wrote my own typing drill program. (see TDRILL.BAS).

WEKDAY.BAS
P. 234-235 in 101 BASIC Computer Games. Adapted from a GE timesharing program by Tom Kloos of the Oregon Museum of Science and Industry and further modified for XITAN Disk Basic.

WISHES.BAS
A silly wish poem writing program. It is in a nice conversational tone for primary school children. This program has good possibilities for expansion into something interesting for slightly older children.

Ordering From CPMUG

CP/M Users Group library diskettes are available from The CP/M Users Group, 1651 Third Avenue, New York, New York 10028. As of this date, the Library contains 54 volumes of software available on 8" IBM single-density CP/M diskettes, or on North Star diskettes readable by users of double-density CP/M 1.4, double-density CP/M 2.2, quad capacity CP/M 2.2.

The complete CPMUG catalogue is available for $6 prepaid to the U.S., Canada and Mexico. The cost to all other countries is $11 prepaid. Members receiving the material are reminded that software contributions are necessary if the exchange program is to prosper. Software contributions are gladly received for inclusion into the Library with the understanding that the contributor is authorized to make the material available to others for their individual non-commercial use.

Please write for more information to CPMUG at the address above.

A Report On CP/M 2.25A For The TRS-80 Model II

The ADM-31 screen emulator does not implement the print screen function (1B,50 Hex). This function would output the screen data to the AMX or printer port of the terminal. Since this auxiliary connector is not present there is no code to address it. The Control P function of the CP/M is normally used to echo console data to the list device.

In this version certain video handling features are implemented. The screen driver in this CP/M is designed to emulate two standard terminals, the ADM-3A and also, with few exceptions, the ADM-31 of the Data Products Division of Lear Siegler, Inc. The ADM-3A is supplied because many software products are delivered pre-configured for this popular screen. The ADM-31 has many more features, and if an application is to be configured for use with this CP/M version, the codes for this screen are preferable. The two terminals have very different commands, and so the two emulations have been made co-resident without their interfering with one another.

The following table shows screen command codes supported for the ADM-31:

Function	Decimal	Hex
Line Feed	10	0A
Reverse Line Feed	11	0B
Non-destructive Forward Space	12	0C
Carriage Return	13	0D
Erase to End of Line	23	17
Clear Screen	26	1A
Home Cursor	30	1E

The following command codes must be preceded by an escape character, ‹ESC›, hex value 1B:

Function	Decimal	Hex
Clear Screen	58	3A
Set Inverse Video	41	29
Reset Inverse Video	40	28
Erase to End of Screen	121	79
Erase to End of Line	84	54
Insert Line	69	45
Delete Line	82	52
Insert Character	81	51
Delete Character	87	57
Position Cursor	61	3D
Print Graphics		

new products new products n

new products new produ

new products new p

new products new

Apartment Management System
by Cornwall Computer Systems, Inc.

This system is intended to make available management reports providing status information on different aspects of apartment house management. It records payments from tenants and charges to tenants for amounts due, prints form letters with or without labels, and can accommodate one or more apartment house complexes with varying numbers of units in each.

Four data files are used for each apartment house. A master file contains information on the whole project and only changes when an apartment is vacant or a new tenant rents. A tenant payment history file maintains information on tenants' payments and allows space for 24 months for each tenant. Financial totals are kept in a year-to-date financial history file which precedes each month's totals with twelve previous months' records. Finally, a fourth file maintains information on former tenants.

Reports can be generated on screen or from the printer, in single or multiple copies. They include: past due and due rents to date, a list of vacant apartments, a list of tenants with leases expiring each month, a list of tenants without signed leases at the end of the month, a list of tenants on dispossess at the end of the month, maintenance data on the condition of each apartment, a report on all amounts due at the end of each month in an entire project, a tenant payment history file by project or individual, a former tenant file, status report on all apartments, total financial history for a year listed by month for a whole project, bank deposit reports.

Apartment Management requires two 250K drives, a printer, and 48K of available memory.

GLector IV
by Micro-Ap Information Management Systems

This is a small business-oriented general ledger option for Selector IV, designed for use by non-accounting personnel. The operator employs a set of user-defined transaction codes which are interpreted by GLector IV; then the appropriate accounts are debited or credited. Trial balances are performed when transactions are entered or updated, and single offset accounts are automatically entered. This general ledger option has a 24-month data storage ability.

GLector IV generates a Balance Sheet for any month, containing current and last year's balances. It will create a P and L afor any period of the current fiscal year showing last year's period, percentage of sales, year-to-date, and percentage change for the period.

GLector IV requires Selector IV and is priced at $450.

Microcache
by Microcosm

This RAM memory extension is controlled by software, which uses the extra RAM as an intelligent buffer between disk drives and normal memory. Disk records required most often by an application are automatically stored in the Microcache buffer and can be transferred at high speed into user memory. A maximum of 256K bytes of memory can be added. It features selective record lock-in and selective disk lock-out and has an autoload facility.

It requires a CP/M or Apple II (with Apple DOS) machine, a Z80 processor; if Cis used, it must be version 2.0 or later (or MP/M or CP/NET).

PANEL
by Roundhill Computer Systems Ltd.

This screen panel design aid is for use with PL/I-80 and CP/M or MP/M. It permits the setting up and modification of screen designs at a terminal and then generates PL/I-80 statements which permit application progra. write to and read from the screen.

PANEL supports multiple- and single-line fields; character and line insert, reverse video, delete within fields, and full cursor movement are convenience features for the end user targeted by the application program. The screen editor allows the design of help screens, menus, and data entry screens. Sample screen layouts can be generated to aid in the programmer's application system documentation.

PANEL requires a 44K TPA, LINK-80, and a 24 by 80 terminal with cursor addressing. It can generate designs for smaller terminals.

The Programmer's Apprentice
by The Software Group

This is a program development tool for CP/M based microcomputer systems. The Apprentice is intended for use by beginners and more advanced programmers. It uses a macro-like language to define standard routine which its code generator utilizes to create MBASIC source code programs. It is designed to create debugged programs in MBASIC for screen prompted data input, data base management, file maintenance and report generation.

A screen editor permits the user to create a report template on screen. Definitions modules allow each file's attributes to be defined exactly.

Apprentice is 8080/Z80 compatible, requires 56K of RAM, 256K of floppy or hard disk storage, serial cursor addressable CRT display terminal, and Microsoft's BASIC-80 compiler.

Silicon Disk System
by Microcosm

This product convinces CP/M that additional standard RAM Memory (up to 8MB's) is really a disk drive. Thus, the user can access this 'drive' as if it was a real disk drive, but much faster.

The Silicon Disk System is designed for implementation with databases, business packages, sorts, and other packages. It is intended to make the

segmentation and overlaying of large programs practical, and to permit the handling of large data arrays. In addition, the manufacturer recommends it for use as a speed buffer for printer spooling, communication, machine control systems, graphics, etc. It features built-in memory diagnostics, the possibility of re-assigning logical to physical disk drive letters, Z80 code, autoload and autostart facility.

New Publications

File Processing With COBOL
by Donald H. Beil
This book is designed to aid in solving business applications problems using structured COBOL; it describes techniques for file creation, merging, sorting, reporting, and updating sequential files. Further instructions are given on updating files and records and effective ways of setting up these processes.

How to Copyright Computer Software
bySofprotex
This is a government publication written in order to clear up confusion about copyright protection for software. It explains some recent rulings on patents and other government activities and regulations pertinent to software development.

CBASIC
Version 2.8

These additions and bug fixes are included in this update:
1-CRUN2 has a new error message, AE; it indicates that an array element was referenced *prior* to executing a DIM statement for that array.
2-The compiler prints messages when a reference to a function is encountered prior to the function being defined. The message provides the name of the function in question.
3-The compiler no longer hangs when a WHILE statement with an invalid or missing expression is detected.
4-XREF now processes more than 255 variables. Previously it hung and would not reboot when more than 255 variables were encountered.

CIS COBOL
Version 4.4

This new update includes improvements to the compiler and run time system.

Compile time command line option can now be selected by the user when loading the compiler. GSA flags can optionally be set at any level and reports will be included in the listing file for features higher than the selected level. ANSI-required source lines can be omitted without detriment, unless the FLAG directive is included to specify ANSI COBOL.

Segments can now be compiled in any order. It is no longer necessary for permanent segments to precede independent ones.

An option parameter ? has been added to the Run-Time Command line; it can be included after the RUN command and before the optional directive parameters that can precede the file name. By default there is a compatibility check between the run-time system and the intermediate code (generated by the compiler) being loaded. The ? parameter inhibits this version check. If the check detects incompatibility, the run is ended with the display of error message 165.

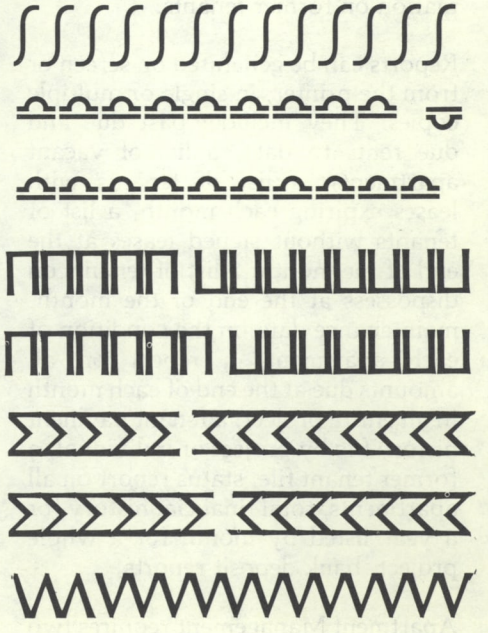
It is generally safe to inhibit the check when Version 4.3 intermediate code is to be run. If unexpected run-time errors result, ascertain that they are not caused by incompatibility or the inclusion of other than 4.3 or 4.4 code.

Users with MP/M Version 1 may experience difficulty at Compile/Run time, a FATAL I/O error may occur depending on the release of MP/M. The following patch to CIS COBOL run time system and compiler will negate this problem:

```
DDT RUNA.COM
DDT VERS N.N
NEXT PC
8900 0100
-S322F
322F 0E 00
3230 0C 00
3231 CD 21
3232 A4 21
3233 42 01
3234 23 .
-G0
A>SAVE 136 RUNA.COM

DDT COBOL.COM
DDT VERS N.N
NEXT PC
8900 0100
-S2148
2148 0E 00
2149 0C 00
214A CD 21
214B BD 21
214C 31 01
214D 23 .
-G0
A>SAVE 136 COBOL.COM
```

Pascal/Z
Version 4.0

This version includes a new feature called SWAT (Software Analysis Tool), an interactive symbolic Pascal debugger; overlay capabilities have also been added. The assembler and linker now can accept up to eight significant characters.



Some Hints On T/MAKER II

A number of people would like to know how to total a row of figures and then use that total to compute each row as a % of that total.

Here is a T/MAKER mask which does it easily.

Note the following features:

1. "=" is at the top of the column
2. 'Store and Fetch' is used
3. 'Compute C' is used (Double Compute)

Try it It works, and Oh, so simple. (Or you can read page 54 in the manual.)

compute c clean

Example of Double compute to allow total of a column to be used as a constant in the further calculation of that column

```
ex                    99,999 ,,,,,, 999.99
zv

jc1                     stA
          STATES      CENSUS          PERCENT
=

ac2                             ftA
ac3                     +       /       %
+         NY            575
+         MASS          422
+         CONN          334
+         NH            229
+         VT            167
```

Figure 1

clean

This illustrates how the mask looks after "COMPUTE C"

```
ex                    99,999 ,,,,,, 999.99
zv

jc1                     stA
          STATES      CENSUS          PERCENT
=                       1,727           100.00

ac2                             ftA
ac3                     +       /       %
+         NY            575             33.29
+         MASS          422             24.44
+         CONN          334             19.34
+         NH            229             13.26
+         VT            167              9.67
```

Figure 2

This illustrates how the table looks after "CLEAN"

STATES	CENSUS	PERCENT
	1,727	100.00
NY	575	33.29
MASS	422	24.44
CONN	334	19.34
NH	229	13.26
VT	167	9.67

Figure 3

Tips and Techniques

Michael J. Karas has sent us a useful tip on speeding up multiple PIP submit files:

"A very common usage of the CP/M SUBMIT utility is to make a new system diskette containing just the files that you need to get started on that new software project. Typically you would start with an existing system diskette in Drive A: that had "everything" on it. A submit file of the following format is used to move the necessary files over to a freshly "formatted" and "sysgened" diskette in Drive B:. (The name of the submit file is arbitrary but could be 'COPYSYS.SUB'.)

Typical format:

```
PIP B:=A:ED.COM[V]
PIP B:=A:FORMAT.COM[V]
PIP B:=A:STAT.COM[V]
```

and so on...

Each time the submit file reenters CP/M for the next command line the PIP utility program is reloaded to perform the next file copy operation. Since the PIP.COM object file is 8K bytes, a considerable amount of time is spent just reading the PIP file back in each time. If you would like to see this operation execute a little faster try the following stunt:

1. With your "everything" disk in Drive A:, type the command:
   A»SAVE 0 @.COM«cr»
   This forms a zero length file that when "loaded" by the CP/M CCP causes the previously loaded transient program to be executed over again.
2. Retype the above typical system copy submit file like this:
   New format:

```
PIP B:=A:ED.COM[V]
@B:=A:FORMAT.COM[V]
@B:=A:STAT.COM[V]
etc...
```

Now PIP.COM is loaded only once for the entire submit file execution. If the files being placed on to the new system diskette in Drive B: are all about 8K to 12K bytes in size, then you may expect an appropriate execution performance factor improvement of 15 to 20%. If all files are very long (25 to 30K), then the speed gain would be minimal. The concept works well for the system disk process due to the fact that most of the system files being copied tend to be quite short but very numerous."

Here is another tip sent to us by Lennart Svensson:

"How many times have you tried to execute PIP after you have changed diskettes? Here is the solution.

Patch the program with these programs:

```
BUFF    EQU  1ECBH
        ORG  11OH
EXTRA   LD   HL,BUFF+1
        LD   A,1
        CP   (HL)
        RET  NZ
        INC  HL
        CP   (HL)
        RET  NZ
        LD   HL,53CH
        EX   (SP),HL
        CALL 82EH
        LD   C,13
        JP   5

        ORG  96FH
        LD   HL,BUFF
        LD   (HL),128
        EX   DE,HL
        LD   C,10
        CALL 5
        JP   EXTRA
```

After patching you key in ⇑A and nothing else; the disk drives are reset."

Change of Address

Please notify us immediately if you move. Use the form below. In the section marked "Old Address", affix your Lifelines mailing label—or write out your old address exactly as it appears on your label. This will help the Lifelines Circulation Department to expedite your request.

New Address:

NAME

COMPANY

STREET ADDRESS

CITY STATE

ZIP CODE

Old Address:

NAME

COMPANY

STREET ADDRESS

CITY STATE

ZIP CODE

bugsbugsbugsbugsbugsbug
bugsbugsbugsbugsbugs
bugsbugsbugsbug
bugsbugsbugsbu

DataStar
Version 1.101

Formlist does not print exclamation points—it also doesn't finish with a crlf.

WordStar
For the Apple

This version in a 44K system cannot handle a block larger than 20 characters.

WordStar
Version 2.26P (ETX/ACK)

Port driver values for this version should be:

	For CCS	For Altos
DATA:	0e09f	1e
STATUS:	0e09e	1f
XMIT :	01	04
REC:	02	01

S-BASIC
Version 5.4

These bugs have been reported in this version of SBASIC:

1-The EXP function may return an inaccurate result for increasingly large arguments.
2-A^X–This is inaccurate if X› 32767. X is separated into its integer part and its fraction part. The integer part is stored in a word INTEGER.
3-DIM–Array names are limited to 32 characters.
4-VAR–Local variables between BEGIN and END statements may change value at run-time. This is because the compiler allocates local variables from different blocks onto the same memory to save space. There is a bug in the compiler causing misalignment of the variables.
5-CLOSE–When closing a sequential file, CLOSE may lose track of the channel number. Causes a run-time error message.
6-VAR/COM/BASED–As names are entered they are fed into a temporary buffer. The length of this buffer is 128 characters. Therefore the sum of the characters of all the names in a given VAR/COM/BASED may not exceed 128.
6-ON ERROR–ON ERROR OFF generates a syntax error. Zero the following byte in S-BASIC.COM. The byte to nop out is 992H, it was a 09; change it to a 00. This can be done with the "S" command of DDT.
7-SERIAL FILES–Once a serial file channel (either ASCII or binary) is used for a read or write operation, that channel becomes a read or write only channel. The first operation sets the channel to read only or write only operation. OPEN and CLOSE will not reset this status.

MMU Announced

Lifeboat Associates has announced the forthcoming release of a series of MMU peripheral boards for Z80 based microcomputers. The MMU is designed to permit almost any host system based upon a Z80 processor to operate SB-80™ and other CP/M compatible operating systems. This added facility is offered without altering the capability of the host system to use its own original native software as and when needed. The MMU will provide, under software control, the correct environment to support these popular disk operating systems. This environment consists of the required memory allocation and a suitable handling to respond to Non-Maskable Interrupts.

The unit is available optionally with 16K bytes of memory which can be used to supplement the machine's existing memory. The unit is designed to map the host system memory and its own optional memory in order to create the desired architecture for the disk operating system.

The installation of the MMU in most systems is a simple plug-in addition to the host machine without any soldering or tampering with the circuit board required.

The unit is intended initially as an OEM device for manufacturers of microcomputers. Lifeboat Associates is able to supply both the MMU hardware and the disk operating system configured for the MMU and the peripherals of the host machine.

Operating Systems

Description	Version
CP/M for:	
Apple II w/Microsoft BASIC	**2.20B**
Datapoint 1550/2150 DD/SS	2.2
Datapoint 1550/2150 DD/DS	2.2
Datapoint 1550/2150 DD/SS w/CYN	**2.2**
Datapoint 1550/2150 DD/DS w/CYN	**2.2**
Durango F-85	2.23
Heath H8 w/H17 Disk	1.43
Heath/Zenith H89	2.2
ICOM 3812	1.42
ICOM 3712 w/Altair Console	1.42
ICOM 3712 w/IMSAI Console	1.42
ICOM Microfloppy (2411)	1.41
ICOM 4511/Pertec D3000 Hard Disk	2.22
Intel MDS Single Density	2.2
Intel MDS 800/230 Double Density	2.2
MITS Altair FD400, 510, 3202 Disk	1.41
MITS Altair FD400, 510, 3202 Disk	**2.2**
Micropolis Mod I - All Consoles	1.411
Micropolis Mod II - All Consoles	1.411
Micropolis Mod I	2.20B
Micropolis Mod II	2.20B
Compal/Micropolis Mod II	1.4
Exidy Sorcerer/Micropolis Mod I	1.42
Exidy Sorcerer/Micropolis Mod II	1.42
Vector MZ/Micropolis Mod II	1.411
Versatile 3B/Micropolis Mod I	1.411
Versatile 4/Micropolis Mod II	1.411
Horizon North Star SD	1.41
Mostek MDX STD Bus	2.2
Ohio Scientific C3	2.24
Ohio Scientific C3-B/74	2.24B
Ohio Scientific C3-C'(Prime)/36	2.24B
Ohio Scientific C3-D/10	**2.24A**
Sol North Star SD	1.41
North Star SD IMSAI SIO Console	1.41
North Star SD MITS SIO Console	1.41
North Star SD	2.23A
North Star DD	1.45
North Star DD/QD	2.23A
Processor Technology Helios II	1.41
by Lifeboat/TRS-80 5 1/4"(Mod I)	1.41
by Lifeboat/TRS-80 Mod II	2.25A

Hard Disk Modules

Description	Version
Corvus Module	2.1
APPLE-Corvus Module	2.1A
KONAN Phoenix Drive	1.8
Micropolis Microdisk	1.92
Pertec D3000/iCOM 4511	1.6
Tarbell Module	1.5
OSI CD-74 for OSI C3-B	1.2
OSI CD-36 for OSI C3-C'	1.2
SA-100A for OSI C3-D	1.2

VERSION LIST

September 16, 1981

The listed software is available from the authors, computer stores, distributors, and publishers.

New products and **new versions** are listed in boldface.

S Standard Version
M Modified Version
OS Operating System
P Processor
MR Memory Required

Product	S	M	OS	P	MR	
ACCESS-80	1.0		CP/M	8080/Z80	54K	
Accounts Payable/Cybernetics	3.1		CP/M	Z80	64K	Needs RM/COBOL
Accounts Payable/MC	1.0		CP/M	8080/Z80	56K	For CP/M 2.2
Accounts Payable/Structured Sys	1.3B		CP/M	8080	52K	w/It Works run time pkg.
Accounts Payable/Peachtree	07-13-80		CP/M		48K	Needs BASIC-80 4.51
Accounts Receivable/Cybernetics	3.1		CP/M	Z80	64K	Needs RM COBOL
Accounts Receivable/MC	1.0		CP/M	8080/Z80	56K	CP/M 2.2
Accounts Receivable/Peachtree	07-13-80		CP/M	8080	48K	Needs BASIC-80 4.51
Accounts Receivable/Structured Sys	1.4C		CP/M	8080	56K	w/It Works run time pkg.
Address Mngemt. Sys	1.0		CP/M	8080		Requires 2 drives
ALDS TRSDOS		3.40	TRSDOS	8080	32K	TRSDOS Macro-80
ALGOL 60	4.8C		CP/M	8080	24K	
ANALYST	2.0		CP/M	8080	52K	Needs CBASIC2,QSORT/VSORT
APL/V80	3.2		CP/M	Z80	48K	Needs APL terminal
Apartment Management (Cornwall)	1.0	1.0	CP/M	8080		Needs CBASIC2
ASM/XITAN	3.11		CP/M	Z80		
Automated Patient History	1.2		CP/M	8080	48K	
BASIC Compiler	5.3	5.3	CP/M	8080	48K	
BASIC-80 Interpreter	5.21	5.21	CP/M	8080	40K	W/Vers. 4.51,5.21
BASIC Utility Disk	2.0	2.0	CP/M	8080	48K	
BOSS Financial Accounting System	1.06		CP/M	8080	48K	Needs 2/-drives w/min 200k each, & 132-col. printer
BOSS Demo	1.06		CP/M	8080	48K	
BSTAM Communication System	4.5	4.5	CP/M	8080	32K	
BDS C Compiler	1.44	1.44T	CP/M	8080	32K	W/'C' book
Whitesmiths' C Compiler	2.0		CP/M	8080	60K	
BSTMS	1.2	1.2	CP/M	8080	24K	
BUG / uBUG Debuggers	2.03		CP/M	Z80		
CBASIC Compiler	2.08		CP/M	8080	32K	
CBS Applications Builder	1.3		CP/M	8080	48K	Needs no support language
CIS COBOL Compiler	4.4,1		CP/M	8080	48K	
CIS COBOL Compact	3.46	3.46	CP/M	8080	32K	
FORMS 1 CIS COBOL Form Generator	1.06	1.06	CP/M	8080		
FORMS 2 CIS COBOL Form Generator	1.1,6a	1.16	CP/M	8080		
Interface for Mits Q70 Printer			CP/M			CP/M 1.41 or 2.XX
COBOL-80 Compiler	4.01	4.01	CP/M	8080	48K	
COBOL-80 PLUS M/SORT	4.01		CP/M	8080	48K	
CONDOR	1.10		CP/M	8080	48K	
CREAM (Real Estate Acct'ng)	2.3		CP/M	8080	64K	CBASIC needed
Crosstalk	1.4		CP/M	Z80		
DATASTAR Information Manager	1.101		CP/M	8080	48K	
Datebook	2.03		CP/M	8080	48K	Needs 80 24 terminal
DBASE-II	2.02A		CP/M	8080	48K	
DBASE-II Demo	2.02A		CP/M	8080	48K	
Dental Management System	8.7A		CP/M	8080	48K	Needs CBASIC
DESPOOL Print Spooler	1.1A		CP/M	8080		
DISILOG Z80 Disassembler	4.0	4.0	CP/M	Z80		Zilog mnemonics
DISTEL Z80/8080 Disassembler	4.0		CP/M	8080/Z80		Intel mnemonics,TDL extensions
EDIT Text Editor	2.06		CP/M	Z80		
EDIT-80 Text Editor	2.02	2.02	CP/M	8080		
ESQ-1	2.1		CP/M	8080		Needs CBASIC2
FABS	2.4A		CP/M	8080	32K	
FILETRAN	1.20		CP/M		32K	1-way TR -80 Mod I,TRSDOS to Mod I CP/M
FILETRAN	1.4		TRSDOS		32K	2-way TRS-80 Mod I,TRSDOS & Stnd. CP/M
FILETRAN	1.5		CP/M		32K	1-way TR -80 Mod II,TRSDOS to Mod II CP/M
Financial Modeling System	2.0		CP/M		48K	
Floating Point FORTH	2		CP/M	8080/Z80	28K	
Floating Point FORTH	3		CP/M	8080/Z80	28K	
FORTRAN-80 Compiler	3.43	3.43	CP/M	8080	36K	
FORTRAN Package	3.40		TRSDOS			
FPL 56K Vers.	**2.51**		**CP/M**	**8080**	**56K**	
FPL 48K Vers.	2.5		CP/M	8080	48K	
General Ledger/Cybernetics	1.3C		CP/M	Z80	48K	Needs RM/COBOL
General Ledger/MC	1.0		CP/M	8080/Z80	56K	CPM 2.2 or MPM
General Ledger/Peachtree	07-13-80		CP/M	8080	48K	Needs BASIC-80 4.51
General Ledger/Structured Sys	1.4C		CP/M	8080	52K	W/It Works Package
General Ledger II/CPaids	1.1		CP/M	8080	48K	Needs BASIC-80 4.51
GLECTOR Accounting System	2.02		CP/M	8080	56K	Use w/ BASIC2,Selector III
GLECTOR IV			**CP/M**	**8080**		**Needs Selector IV**
HDBS	1.05		CP/M	+	52K	
IBM/CPM	1.1		CP/M	8080		
Integrated Acctg Sys/Gen'l Ledger			CP/M	8080	48K	Needed for pkgs below
Integrated Acctg Sys/Accts Pyble			CP/M	8080	48K	
Integrated Acctg Sys/Accts Recvble			CP/M	8080	48K	
Integrated Acctg Sys/Payroll			CP/M	8080	48K	
Interchange			CP/M	Z80	32K	
Inventory/MicroConsultants	5.3		CP/M	8080/Z80	56K	Needs CP/M 2.2
Inventory/Peachtree	07-13-80		CP/M	8080	48K	Needs BASIC-80 4.51
Inventory/Structured Sys	1.0C		CP/M	8080	52K	W/It Works Package
Job Cost Control System/MC	1.0		CP/M	8080/Z80	56K	Requires CP/M 2.2
JRT Pascal System	1.4		CP/M	8080	56K	
LETTERIGHT Text Editor	1.1B		CP/M	8080	52K	
LEVEL 3 BASIC / G2			TRSDOS			Cassettes
LINKER			CP/M	Z80		
MAC	2.0A		CP/M	8080	20K	
MACRO-80 Macro Assembler Package	3.43	3.43	CP/M	8080/Z80		
Magic Wand	1.11		CP/M	8080	32K	
MAGSAM III	4.2		CP/M	8080	32K	
MAGSAM IV	1.O		CP/M	8080	32K	
MAILING ADDRESS Mail List System	07-13-80		CP/M	8080	48K	
Mail-Merge	3.0		CP/M	8080		
Master Tax	1.0-80		CP/M	8080	48K	Has 1980 tax forms
Matchmaker			CP/M	8080	32K	
MDBS	1.05		CP/M	+	48K	
MDBS-DRS	1.02		CP/M	+	52K	
MDBS-QRS	1.0		CP/M	+	52K	
MDBS-RTL	1.0		CP/M	+	52K	
MDBS-PKG			CP/M	+	52K	W/all above MDBS products

VERSION LIST

Product	S	M	OS	P	MR	
Microspell	**4.21**		**CP/M**	**8080**	**48K**	**Needs 15 K**
Medical Mngemt.System	**8.7A**		**CP/M**	**8080/Z80**	**48K**	**Needs BASIC-80,5.03 or above**
Microcosm			**CP/M**	**Z80**		**CP/M 2.X or MP/M**
Mince	**2.6**		**CP/M**	**8080**	**48K**	
Mince Demo	**2.6**		**CP/M**	**8080**	**48K**	
Mini-Warehouse Mngmt. Sys.	5.5		CP/M	8080	48K	Needs CBASIC
Money Maestro	1.1		CP/M	8080/Z80	48K	CP/M.14 or 2.2
MP/M Operating System	1.1		MP/M	8080	32K	
MSORT	1.01		CP/M	8080	48K	
Microstat	**2.01**		**CP/M**	**8080**	**48K**	**Needs BASIC-80, 5.03 or above**
Mu LISP-80 Compiler	2.10		CP/M	8080		
Mu SIMP / Mu MATH Package	2.10		CP/M	8080		muMATH 80
NAD Mail List System	3.0D		CP/M	8080	48K	
Nevada COBOL	2.0		CP/M	8080	32K	
Order Entry w/Inventory/Cybernetics			CP/M	Z80		Needs RM/COBOL
Panel	2.2		CP/M		44K	also MP/M
PAS-3 Medical	1.76		CP/M	8080	56K	Needs 13 -col. printer & CBASIC
PAS-3 Dental	1.62		CP/M	8080	56K	Needs 13 -col. printer & CBASIC
PASM Assembler	1.02		CP/M	Z80		
Pascal/M	**4.02**		**CP/M**	**8080**	**56K**	
PASCAL/MT Compiler	3.2		CP/M	8080	32K	
PASCAL/MT+ w/SPP	5.25		CP/M	8080	52K	Also has 32K version & SuperBrain Ver.
PASCAL/Z Compiler	**4.0**		**CP/M**	**8080**	**56K**	
Payroll/Cybernetics, Inc.			CP/M	Z80		Needs RM/COBOL
Payroll/Peachtree	11-7-80		CP/M	8080	48K	Needs BASIC-80 4.51
Payroll/Structured Sys	1.00		CP/M	8080	60K	No longer needs CBASIC
PEARL SD	**3.01**		**CP/M**	**8080**	**56K**	**w/CBASIC2,Ultrasort II**
PLAN80 Financial Package	2.0		CP/M	8080	56K	Z80 & 8080 Microproc. Vers. avail.
PL/I-80	1.3		CP/M	8080	48K	
PLINK Linking Loader	3.25P		CP/M		Z80	
PLINK-II Linking Loader	1.08		CP/M	Z80	48K	
PMATE	**3.02**		**CP/M**	**8080**	**32K**	
PRISM/ADS	**1.0**		**CP/M**	**8080**	**56K**	**Needs CBASIC, 2.06 or later & 180K/drive**
PRISM/IMS	**1.0**		**CP/M**	**8080**	**56K**	**Needs CBASIC, 2.06 or later & 180K/drive**
PRISM/LMS	**1.0**		**CP/M**	**8080**	**56K**	**Needs CBASIC, 2.06 or later & 180K/drive**
POSTMASTER Mail List System	3.4	3.4	CP/M	8080	48K	
Professional Time Acctg	3.11a		CP/M	8080	48K	Needs CBASIC2
Programmer's Apprentice			**CP/M**	**8080/Z80**	**56K**	**needs Basic-80**
Property Manager	07-13-80		CP/M	8080		Needs BASIC-80 4.51
Property Mngemt. Sys.	1.0		CP/M	8080	48K	Needs CBASIC
PSORT	**1.2**		**CP/M**	**8080**		
QSORT Sort Program	2.0		CP/M	8080	48K	
Real Estate Acquisition Programs	2.1		CP/M	8080	56K	Needs CBASIC
Remote	3.01		CP/M	Z80		
Residential Prop. Mngemt. Sys.	1.0		CP/M	Z80	48K	
RM/COBOL Compiler	1.3C		CP/M	8080	48K	w/Cybernetics CP/M 2
RAID	4.7.3A	4.7.3	CP/M	8080	28K	
RAID w/FPP	4.7.3A	4.7.3	CPM	8080		
RECLAIM Disk Verification Program	2.1		CP/M	8080		
SBASIC	5.4		CP/M	8080		
Scribble	1.2		CP/M	8080/Z80	56K	
SELECTOR-III-C2 Data Manager	3.24	3.24	CP/M	8080	48K	
SELECTOR-IV	2.14A		CP/M	8080	52K	Needs CBASIC
Shortax	1.2		CP/M	Z80	48K	TRSDOS,MDOS too, needs BASIC-80 5.0
SID Symbolic Debugger	1.4		CP/M	8080		N/A-Superbrain
SMAL/80 Programming System	3.0		CP/M	8080		For CP/M 1.x
Spellguard	1.0		CP/M	8080/Z80	32K	Needs Word Processing Program
Standard Tax	1.0		CP/M	8080	48K	Needs BASIC-80 4.51
STATPAK	1.2	1.2	CP/M	8080		Needs BASIC-80 4.2 or above
STRING BIT FORTRAN Routines	1.02	1.02	CP/M	8080		
STRING/80 bit FORTRAN Routines	1.22		CP/M	8080		
STRING/80 bit Source	1.22		CP/M	8080		
SUPER SORT I Sort Package	1.5		CP/M	8080		Max. record=4096 bytes
T/MAKER IIData Calculator	**2.3.1**		**CP/M**	**8080**	**48K**	
T/MAKER II DEMO	2.21					
TEX Text Formatter	1.1		CP/M	8080	36K	
TEXTWRITER-III	3.6	3.6	CP/M	8080	32K	
TINY C Interpreter	800102C		CP/M	8080		
TINY C-II Compiler	800201		CP/M	8080		
TRS-80 Customization Disk	1.3		CP/M	8080		
ULTRASORT II	4.1A		CP/M	8080	48K	
Lifeboat Unlock	1.3		CP/M	8080		Use w/BASIC-80 5.2 or above
VISAM	**2.0**		**CP/M**	**8080**	**48K**	
Visicalc	1.37		Apple	8080	32K	
WordIndex	3.0		CP/M	8080	48K	Needs Wordstar
Wordmaster	1.07A		CP/M	8080	40K	
Wordstar	3.0		CP/M	8080	48K	
Wordstar w/MailMerge	3.0		CP/M	8080	48K	
XASM-05	1.04		CP/M	8080	48K	
XASM-09	1.05		CP/M	8080	48K	
XASM-51	1.07		CP/M	8080	48K	
XASM-F8	1.03		CP/M	8080	48K	
XASM-400	1.02		CP/M	8080	48K	
XASM-18 Cross Assembler	1.30		CP/M	8080		
XASM-48 Cross Assembler	1.30		CP/M	8080		
XASM-65 Cross Assembler	1.95		CP/M	8080		
XASM-68 Cross Assembler	1.96		CP/M	8080		
XMACRO-86 Cross Assembler	3.40		CP/M	8080		
XYBASIC Interpreter Extended	2.11		CP/M	8080		
XYBASIC Interpreter Extended CP/M	2.11		CP/M	8080		
XYBASIC Interpreter Extended COMP	2.0		CP/M	8080		
XYBASIC Interpreter Extended ROM	2.1		CP/M	8080		
XYBASIC Interpreter Integer	1.7		CP/M	8080		
XYBASIC Interpreter Integer COMP	2.0		CP/M	8080		
XYBASIC Interpreter Integer ROM	1.7		CP/M	8080		
Z80 Development Package	3.5		CP/M	Z80		N/A-Magnolia,Superbr'n,mod.CP/M
ZDM/ZDMZ Debugger	1.2/2.0		CP/M	Z80		For Micropolis,N'Star,Apple,IBM8"
ZDMZ DeBugger	2.0		CP/M	Z80		See note above.
ZDT Z80 Debugger	1.41	1.41	CP/M	Z80		N/A-Superbr'n,mod.CP/M
ZSID Z80 Debugger	1.4A		CP/M	Z80		N/A-Superbr'n,mod.CP/M

+ These products are available in Z80 or 8080, in the following host languages: BASCOM, COBOL-80, FORTRAN-80, PASCAL/M, PASCAL/Z, CIS-COBOL, CBASIC, PL/I-80, BASIC-80 4.51, and BASIC-80 5.xx.

If you can't find the right program in our new catalog, it probably hasn't been written.

Take a byte out of our SOFTWARE WITH FULL SUPPORT Catalog

As the world's leading publisher of professional software for microcomputers, Lifeboat Associates offers the largest selection of state-of-the-art programs.

And our new catalog has more to offer than ever. Including:

SYSTEM TOOLS such as:

ZAP80—Powerful menu-driven disk utility for CP/M®-compatible 8080/Z80™ operating systems. Allows direct access to disk surface; includes extensive file utility-servicing to access and patch sectors. Permits file comparisons, sector saving, chaining to .COM files, allows word processor-like editing on disk sectors, and more. $175.

LANGUAGES such as:

STIFF UPPER LISP—Extensive, yet compact, implementation of Lisp packs over 120 compiled functions into less than 14K. Includes library of functions written and callable as needed. Features symbolic debugging aids, an editor, on-line help facility, print formatter, spelling correction function, and much more. Includes manual with study guide, program examples, and the book "LISP" by Winston & Horn. $165.

LANGUAGE AND APPLICATION TOOLS such as:

VISAM—Variable Index Sequential Access Methods allows programmers working in PL/I-80™ to create application with file management capabilities rivaling those of main frame systems. Features variable length keys and records, buffer pool management with "least referenced" algorithm, automatic space management and much more. Allows random, sequential (forward and backward), and skip sequential by full or partial key access. Instructions included for implementing hierarchical and relative data access. $250.

WORD PROCESSING SYSTEMS AND AIDS such as:

BENCHMARK—Easy-to-use, menu-driven, word processing system requiring no special control codes. Just read the function you want to perform, enter the appropriate function number, and its done. It's just that simple! $495.

MAILING LIST SYSTEMS such as:

Benchmark Mail List—create customized letters by merging data from this Mail List into Benchmark word processor-created text. $395.

DATA MANAGEMENT SYSTEMS such as:

Prism—State-of-the-art family of data base management tools; available in several versions, all of which include user manual, sample applications and reports:

Prism/LMS—List Management System creates and maintains lists of customers, parts, vendors, employees, subscribers, etc. Includes unique forms generator; prints user-defined mailing labels, envelopes, form letters, etc. Can create forms outside Prism with a text editor. Accesses records by any key, and allows printing of reports in the desired order without sorting. $250.

Prism/IMS—Information Management System, easy-to-use by non-technical personnel and more powerful than LMS. Interactive multi-keyed files may have up to 99 keys. Includes comprehensive Report Management System. Selects record by fields or calculations (i.e.<, >, <=, >=, etc.). Features control breaks for up to 9 levels of subtotals, full calculation capability, print formatting (footers, headers, etc.), and more. $495.

Prism/ADS—Application Development System designed for professionals. Includes described features in LMS and IMS plus complete set of programming aids to reduce time required to create specialized applications. Creates multi-level menu structures with no programming; calls user written programs directly from menus with automatic passing of user data to called programs. Password protection of critical data. Permits creation of custom programs with professional capabilities. Complete screen-display, formatting, and data-entry functions supplied. Includes powerful MAGSAM/E multi-keyed file management system. $795.

NUMERICAL PROBLEM-SOLVING TOOLS such as:

PLAN80—Powerful tool for improving quality, accuracy, and timeliness of plans, forecasts, budgets, and analyses. Define rows and columns, enter data, specify calculations to be performed ("rules") and the program will process the data and report the results either on CRT or hard copy. Features exclusive graphing mode to display results of calculations in form of on-screen graphs. Recalculates data at your command; useful for "what-if" forecasting. Powerful, yet easy to use. $295.

PROFESSIONAL AND OFFICE AIDS such as:

CORNWALL APARTMENT MANAGEMENT—Automates many functions of rental management office. Uses 4 main data files to capture present and former tenant data, payment history, and property financial history. Reports include Vacancy List, Past Due Rents, Lease Expiration, Unsigned Lease, Apartment Status List, and more. Prints labels, rent increase notices, late notices, form letters, etc. $165.

Series 9000 Family Dental/Medical Management-Univair—Utilizes open-item accounting method; maintains full details of bills until paid or cleared. Keeps record of all family members in same file for billing to appropriate party. Maintains files for Patient History, Patient Profile, Insurance Company Codes, Procedure Codes, Diagnosis Codes, Patient Scheduling, AR, GL status, etc. Makes data available on inquiry or report basis. Prints statements, and ADA/AMA insurance forms. Custom forms available. $950. each

Series 9000 Insurance Agency Management-Univair—Have the same data processing capabilities available to large independent agencies! Code up to 999 types of coverage with full description and standard premium rates charged by each carrier. Store carriers with appropriate agency commission for each type of coverage. Maintain agency and producer files with month, year, and last year-to-date commissions earned. Includes files for Client Master, Policy, Claim History, Coverage Code, etc. Report in detail, or summary form, for Agency, Producer, Insurance Company, Client Master, Policy Register, etc. Generates statements and reports for direct-billed accounts, and more! A must for the small to medium insurance agency. $950

Wiremaster—Tool to aid in design, layout,and construction of electronic hardware. Designed primarily for wrapped wire construction techniques; also useful for layout, error checking, and trouble shooting of printed circuit board. Input derived directly from schematic diagram. Outputs a parts list, error list (i.e., wires that go nowhere, etc.), wire list, pin cross reference list, etc. Invaluable. $150.

DISK OPERATING SYSTEMS

CP/M 2.x configured for the following Ohio Scientific systems:

OHIO SCIENTIFIC C3-B/74	$250.00
OHIO SCIENTIFIC C3-C'/36	$250.00
OHIO SCIENTIFIC D/10	$250.00

SPECIAL 50% OFF CP/M 2.2 configured for Heath H89 by Magnolia. Limited quantities available. Regularly $250.00 Now $125.00

WRITE, TWX, TELEX, OR CALL FOR MORE INFORMATION ABOUT THESE AND ALL OF OUR 200 PROGRAMS, INCLUDING TELECOMMUNICATIONS, GENERAL PURPOSE APPLICATIONS, FINANCIAL ACCOUNTING PACKAGES, BOOKS, PERIODICALS, AND ACCESSORIES.

LIFEBOAT ASSOCIATES' SOFTWARE IS AVAILABLE FOR OVER 100 POPULAR 8080/Z80 COMPUTER DISK SYSTEMS, INCLUDING: HEWLETT-PACKARD 125, XEROX 820, AND OSBORNE-1.

LIFEBOAT WORLDWIDE offers you the world's largest library of software. Contact your nearest dealer or Lifeboat:

Lifeboat Associates
1651 Third Ave
New York, N.Y. 10028
Tel: (212) 860-0300
Telex: 640693 (LBSOFT NYK)
TWX: 710-581-2524

Lifeboat Inc.
OK Bldg., 5F
1-2-8, Shiba-Daimon
Minato-ku, Tokyo 105, Japan
Tel: 03-437-3901
Telex: 2423296 (LBJTYO)

Lifeboat Associates, Ltd
PO Box 125
London WC2H 9LU, England
Tel: 01-836-9028
Telex: 893709 (LBSOFTG)

Lifeboat Associates GmbH
Hinterbergstrasse
Postfach 251
6330 Cham, Switzerland
Tel: 042-36 8686
Telex: 865765 (MICO CH)

Intersoft GmbH
Schlossgartenweg 5
D-8045 Ismaning, W. Germany
Tel: 089-966-444
Telex: 5213643 (ISOFD)

Lifeboat Associates, SARL
10, Grande Rue Charles de Gaulle
92600 Asnieres, France
Tel: 1-733-08-04
Telex: 250303 (PUBLIC X PARIS)

Lifeboat Associates
World's foremost software source

Mail coupon to: Lifeboat Associates,
1651 Third Avenue, New York, New York 10028
or call (212) 860-0300.

Please send me a free Lifeboat catalog.

Name_____ Title_____

Company_____

Street_____

City_____ State_____ Zip_____

SOFTWARE WITH FULL SUPPORT

Z80 is a trademark of Zilog, Inc.
CP/M is a registerd trademark of Digital Research, Inc.
PL/I-80 is a trademark of Digital Research, Inc.

LIFELINES®
1651 Third Avenue / New York, N.Y. 10028

NEW YORK N. Y.

FIRST CLASS MAIL
U. S. POSTAGE
PAID
Permit No. 416

FIRST CLASS MAIL