MICROPOLIS USERS GROUP

MUG Newsletter #32 - March 1983

Published Monthly by the MUG
Subscription rates:
North America; $18/year
Elsewhere; $25/year - airmailed

```
*****************************************************
*                                                   *
*       VECTOR GRAPHIC SPECIAL INTEREST GROUP       *
*                                                   *
*****************************************************
```

THE JOYS OF BUSINESS MICRO-COMPUTING

by Charles T. Goetz, DBI Industries, Inc.
80 Allen Rd, S. Burlington VT 05401
802-863-6873

Let me tell you my sad story.

Small businessmen have to be innovative--right? Right. So better than five years ago we carefully speced out a nice little micro system to handle our accounting and general office work. We bought a Xitan box (S-100), good high quality printer and two Micropolis MOD 1053-II drives. By upgrading our memory board we had 64K of ROM and 1.2 megs of storage on line, a good four years before the current revolution in micros hit. We never did use the Micropolis operating system, electing CP/M because we felt that there would be more programs available for that system.

Pretty good so far. Since we are electronics oriented there was always some guy in-house who could write code and we developed a decent set of programs which have served us well. Still doing pretty well.

Of course Xitan went down the pipe very shortly after our purchase. But we had our local dealer for support. . . . .

"Ahh -- what do you mean his phone has been disconnected?

Well, no big deal. Somebody in-house could always make little adjustments here and there in the hardware and software when needed. And that really was not very often. Meanwhile, back at the ranch, we are building all sorts of records and adding a few bytes here and a few bytes there into our data file. Like good little fellows, we are getting our printouts for back-up and, after all, I got enough things around here that don't work so why should I worry about something that is working?

Gradually these "in-house guys who can write a little code" get ideas about upgrading their lot in life. Electronic technicians suddenly become computer consultants and one day there was nobody around to answer the question, "what do I do--the computer is saying BAD SECTOR?"

What do you do? You learn in one hell of a hurry. You learn what MTBF is all about. You learn that CPU is not CP/M's sister. You learn that the local computer store does not sell parts for Xitan. (They look at you like you had a disease). Late one Thursday evening you begin to wonder what would happen if that thing smoked before the payroll tomorrow morning?

So, like a good little businessman, you buy yourself some spares. You see, there is a difference between relying on a computer for your business operation and for some red eye hobby-type fun nights and weekends.

Every good little manufacturer is suppose to expand. If you do it right, you first make sure that you have damn good control over your existing operations. Being mightily impressed with the power of the computer and the descending prices of new equipment we started to think pretty heavy about a second system. Number one, we wanted to back-up our existing system in case of a failure and number two, we wanted to start to put our manufacturing operations on computer.

Lots of nifty new systems out there with megabytes of memory and multi-megabytes of storage. What a grand new world! So why not invest in one of these systems and, if your Xitan box smokes on Thursday night, do your payroll on the new system.

Oh yea? "You got a what kind of system?" "A what kind of disc format?" "Well, that's no big problem. We'll just write you a new program to transfer all your files over to our system and you won't have to worry about your old stuff."

Doesn't sound right. So you get on the phone and find out that there is a guy named Buzz Rudow down in Alabama who knows a little bit about the Micropolis format and this fine fellow says, "Hey, Vector Graphic uses the same format. Why don't you look into their line?"

Wow! Vector has just come out with a brand new, hot model--the Vector 4. You can get it with two double density 77TPI, double sided disc drives. Since we run, in effect, four drives for our programs (A, B, C and D) 2 times 2 equals 4--right? You even go for a demo at your nearest dealer (200 miles). The dealer does a Config F and the thing asks you how you want to configure drives A, B, C and D--double or single sided?

Bingo! Besides the back-up for your Xitan you have just about unlimited expandability because this Vector 4 is designed for a networking system which would be ideal for inventory control, shop floor control, scheduling, purchasing, etc., etc., etc. I must be dreaming.

Purchasing goes out for bids, we get a nice price and eventually the thing gets here, a little bit worse for wear, but never-the-less our dream machine is here.

The first thing you find out is that there is no drive C or drive D. Drive A is either single sided or double sided--the underside is not drive B or even drive C or drive D. Sure, if your accounting equipment is down, you can bring the discs over and transfer files to a double sided disc. Only trouble with that is that you are going to have to rewrite your programs and that means a lot more than just redesignating drives, or drive addresses within the programs. A four disc interaction doesn't transfer that easily. Take my word for it.

A month or so later some of your manuals start to show up (they cost extra) and you find out that you've got a couple of vacant S-100 slots inside the Vector 4. Being the proud owner of a back-up Micropolis disc driver board and spare drives, there should not be a great problem in running the thing with four single sided drives.

So, since you work late nights on this thing anyhow why not call Vector in California and speak with an applications engineer? Well, pretty quick you find out that Vector in California is really an answering machine that says, "I am sorry sir. You will have to go back to your dealer."

So, we go back to our dealer, who is really a nice, accommodating guy. He goes back to Vector (he has the secret code that gets him past the answering machine) and he gets back to you to tell you that the S-100 buss in the Vector 4 is "modified". This means, according to them, that configuring for four single sided drives is nie on to impossible. (I don't believe it).

O.K. So what about all the external ports on this magnificent machine? Can't we just designate an I/O port as drives C and D? "Ahh-Umm. The networking programs are not yet available from Vector." Etc., etc., etc.

Oh, I also might add that, although both of my systems are CP/M, my Vector programs will not work on my Xitan.

O.K., I've bored you long enough with my problems. Now a proposition.

As a small business owner, not particularly sophisticated in either computer hardware or software, I can see that the explosive growth in micros is going to have a very significant effect on small businesses. Lots of people are going to jump into micros as a cure-all for business information systems. I think that they are going to get burned. I believe that user's groups are bound to become an absolute necessity. The swapping of information for mutual benefit could save an awful lot of trauma.

The setting up of a Vector Graphic special interest group by MUG was really a welcome announcement. For one, I would be happy to share what little information I have, and will learn, with anybody who wants it. We do have some expertise in dealing with the Micropolis hardware and I would also be willing to share this. The Vector 4 is a new piece of equipment and there probably are not a lot of users out there yet. (Remember the pioneer? He was the guy who got the arrow in the back.)

So if anybody out there needs some answers from me or can give some answers to me, I would be delighted to hear from them.

....

```
***************************************************
*                                                 *
*          BASIC/Z SPECIAL INTEREST GROUP         *
*                                                 *
***************************************************
```

BASIC/Z CORNER #2

bySteven Guralnick
375 South Mayfair Avenue, Suite 205
Daly City, CA 94015

I am back.

I mentioned in the last article that Jerry Lenz and I put together a Client Ledger Program, for this law office. I purchased BASIC/Z to convert that program from Micropolis BASIC to BASIC/z. Since the conversion was a total success and since I then used BASIC/Z to go on from there, I think it would be a helpful tutorial to talk about my use of BASIC/Z for that particular package and I will start with the conversion process. In this, I am assuming that you have some basic familiarity, (forgive the pun) with the operation of BASIC/Z, i.e., that you have turned it on, have used the editor and have used the compiler. If you do not yet own BASIC/Z, let me assure you that learning the routines is quite straight forward. The editor is almost identical to the editor in Micropolis BASIC, only it is better.

The first thing you need do to convert from Micropolis BASIC to BASIC/Z is to be aware that there are a few special syntactical requirements in BASIC/Z. What it really comes down to is you should have written your original code in a proper way. For example, in Micropolis BASIC you can get away with: IF A=B THEN100. BASIC/Z insists on a space between "THEN" and "100". TAB statements require semicolons in BASIC/Z, and so forth. You have two choices: you can tidy up this type of thing while you are still in Micropolis BASIC or you can transfer over and then tidy up with the BASIC/Z editor. Frankly, it is a little easier to do it later, because you have a search and replace function. So, if you have a load of the same type of error, you can usually construct a search and replace routine that will get rid it. That is the way I did it when I was converting the Client Ledger Program. If you are looking for thirty-five TAB statements, it is a lot easier to tell the editor to go list them for you than having to scan your code one line at a time, looking for them.

The first step of the conversion process is to set up a blank, formatted disk, with your CP/M operating system on it. To that disk, you should transfer the programs called TR/III.BZO, BASIC/Z.COM, BASIC/Z.OVL and RUN/Z.COM. Place that disk in the drive that you boot on, usually A:. The reason for this is that the translator program (TR/III) reboots the system every time it finishes and you will always get the CP/M A> when you are finished. Next, put into another drive a copy of your Micropolis BASIC source code or codes. Now, enter "RUN/Z TR/III". The TR/III program will sign on and ask you the name of your MDOS file. Enter the name, and be sure and put the drive designator in front of it. Then the program will prompt you for the name of your new CP/M file. Enter the drive designator and you should use the same file name BUT you MUST use a file name extender of ".ASC", if this is a source code you are transferring and not a data file. The reason for this is this: When you try to load it later into the BASIC/Z editor, it will only load ASCII files (which your newly transferred file is) if you indicate that you are loading a file with an extender of ASC and the file itself has an extender of ASC. I spent a whole evening of wasted time on this, until Buzz bailed me out, so take a lesson from my book. If your MDOS file is a data file, then you can use the same name in CP/M as you would in MDOS, provided that it does not exceed eight characters.

When you have entered the correct CP/M file name, press <CR> and go get a drink of water. Depending on the size of the file it should take a little while.

When the TR/III program has completed, you will be returned to CP/M. Then, enter "BASIC/Z". When the editor signs on, enter "LOAD" followed by the file name and the ASC extender. The editor should then indicate to you that the file is loaded. If you are having any problems with this, I will be happy to be of any help I can. Start with a small MDOS file so you don't have to wait a long time for the translation.

Once the file is loaded into the BASIC/Z editor, immediately save it, by entering "SAVE". Then, I recommend you reload the file by simply entering "LOAD" and the file name, only this time no extender is necessary. Your ASCII file is still on disk but now you will be working with a token file and now it will LOAD and RESAVE much more rapidly. It has a ".BZS" extender.

When the file is loaded into your editor, do a listing and let it run by your screen. If you have some syntax problems, the editor will place an apostrophe immediately to the right of the offending line number. You can search for the apostrophe, using the search routine, or you can just scroll your text and look at them one at a time. Here are a few of the typical apostrophe situations: ASSIGN must be changed, either to LPRINTER or CONSOLE; WIDTH must be converted into a PAGESIZE statement.

One of the errors that you will not see with an apostrophe is that your OPEN statements must have a RECLEN option included if you are opening a random access file. When TR/III transfers your data files, it arbitrarily sets the record length to 250 so use RECLEN 250 for all such OPEN statements, for the time being. Because this is CP/M, you are not stuck with such a large record size, if in fact your record size is much smaller. Later on, after everything is running well, you can write some converter programs that change the record lengths down to something more reasonable and then you will have to go back and change those RECLEN statements. If you are advanced enough, you can write the converter programs first and save yourself an extra step later.

Remember to remove the apostrophes each time you correct the error. If you remove the apostrophes and the error is not corrected, BASIC/Z will flash an error mark and point to the location of the problem. This can be very handy if you are not sure where the error is; just remove the apostrophe and press <CR>. Then you will see the location of the error. To get out of a situation without correcting the error, press "Q" and you will be returned back to the original status of that line.

When you have got all the errors out, and have re-saved the program, compile and then run it. I will quit here on this phase of BASIC/Z and look for some comments or questions.

In looking over the February MUG letter, I noticed two articles which tend to affect what I do so I will use this column to make some comments. One subscriber hates CP/M and thinks that MDOS is far superior. It may be; I am not an expert on MDOS, but CP/M is the choice for thousands of programmers and this whole column has come about because our new computers require CP/M. There is no question that that operating system is the standard of the microprocessing industry and I think it's beating one's head against the wall to pretend that it is not.

Second, I read with much interest Buzz' article on $29.95 software. I agree with Buzz completely. There is no free lunch in this business. The days when one purchased some small game or check balancer for such a low price have passed, and good riddance to them. We are running with some exotic, powerful and reliable applications software in this office. I know the authors of many of those packages and I know they sweated blood to make them what they are today. Those people are worth being paid for their time and expertise. People like Buzz have to have enough profit incentive to make it worth supporting people like you and me and I want to take this opportunity to mention that Buzz' support for me of BASIC/Z has been nothing short of super.

One of the problems I will be having with this column is the lead time. This is being finished up on February 10 so you won't see it for a month. If you have any bright ideas you would like me to include in the next article, call me if you would, to get it in right away.

See you soon!

....

LETTERS

LIBRARY DISK PROTECTION

Buzz: I want to send some programs in for the library. Any special instructions, such as wrapping the diskettes in aluminum foil, stating originating hardware, instructions on outside for postal handling, magic spells for safe delivery, etc.?

(Response) Just cut two pieces of cardboard and pack the disk(s) between them, with tape around the edges. I don't wrap disks in aluminum foil, though some members do. There haven't been any problems that I know of in Library Disk distribution. My assumption is that disks are MOD II, that is, single-sided, quad-density. If you are submitting a MOD IV (double-sided, quad-density) or a MOD I (single-sided, double-density), its best that you state so on the disk. No particular instuctions need be put on the package - they are probably ignored anyway - but you could write "Magnetic Media - Do Not X-RAY" if you wish.

MICROPOLIS MAINTENANCE AIDS

Can I use the Micropolis Maintenance Aids for their disk drives on the Tanden drive in my Vector VIP?

(Response) The Maintenance Manual and Diagnostic Disk will be of no use at all. I'm not sure about the Alignment Disk. I'll have to check.

Your question brings up a point, however, about the systems having Micropolis drives but no MDOS. The last year or two of System-Bs were delivered just with CP/M. The Diagnostic Disk runs under MDOS. If you have a local service shop, check to see that they can align your drives. If not, you'll have to do it yourself, and you might want to consider getting MDOS installed before the drives go bad. If there is a CP/M version of the Diagnostic Disk, I'd appreciate hearing about it. The Maintenance Manual is useful, no matter what operating system you are running.

CPMUG SOFTWARE

MUG Newsletter readers might want to know about Sheepshead Software, P.O. Box 486, Boonville CA 95415. They sell CPMUG public domain software for the Vector 3 for $10 per volume.

VECTOR ADD-ON DRIVES

I want a second drive, but my dealer wants to charge me $1000. That seems rather high. I have an extra connector on my drive cable and it seems that all that is needed for installation is to plug the second unit in and tell it to respond to requests for drive B. Aren't cheaper drives available? Where?

(Response) Indeed that sounds high for a single drive. If the purchase was for a double-drive with power supplies, then the price isn't too bad. It may be possible to "hang" a drive on your existing data and power cables. Anyone in the group have a solution or a suggestion?

David Vergin, P.O. Box 700, Milton WA 98354

....

MDOS BUG

I have found what I consider to be a 'bug' in MDOS! The @DFINXPOS routine does not do what the Micropolis Manual says. That is, it does not make the record length equal to the value of the index position. It makes it equal to the value of the index position+1. LINEEDIT makes allowance for this in its operation, but it caught me completely by surprise. The @DFINXPOSTEOR routine does the right thing, but not @DFINXPOS.

Maybe Micropolis had a reason for doing it, but not if the manual is to be believed. It makes things very awkward when a subsequent program tries to find the end of file and ends up pointing to a byte of absolutely no importance.

Could anyone tell me if it really is a bug? Any comments would be appreciated.

Pete Gorton, 44, Francis St., Castle Hill 2154, N.S.W. Australia

....

CLASSIFIED

FOR SALE: Perfect Writer word processor. It requires a 56K system with direct cursor addressing. While I could mod my system to get this, then I couldn't run my ROBOT software. I've elected to stay at 48K. Make offer.

Quint Mushik, Rt. 1, Box 408C, Guntersville AL 35976, Phone (205) 586-6061 x240 (days) or (205) 586-7307 (nights)

....

THE BASIC/Z COMPILER
Product Review

By
Joel Shapiro
BONJOEL ENTERPRISES
P.O. Box 2180
Des Plaines, Ill. 60018

One of the nicest programming tools and languages I've been privileged to use and review is this latest entry of System/z in CP/M compatible software. What the other Basic's lack, BASIC/Z seems to have included. What appears limited in others is expanded and enhanced in this fine system. Nowadays, it takes more than improved speed to sell people on a compiler and this system offers that and more.

Having just completed the development of several pieces of commercial software using the system, I can say it's pretty easy to learn how to take advantage of the built-in statements and functions, let alone the excellent editor for getting the program into the system in the first place. More about these later.

Basic System Structure

BASIC/Z is considered a native 8080/Z80 code compiler. The object code produced by the compiler is executed directly by the microprocessor without the use of an interpreter.

BASIC/Z, because of its structure, can be considered a 'hardware independent' language. Once a program is developed, it can be used with practically any hardware using CP/M because of the method of interface used.

The BASIC/Z system utilizes a 'Run-time' program and it is resident in the computer whenever a compiled program is being used. The program, Run/z, contains most of the subroutines used by the compiled program. It is this program that is interfaced to the selected computer.

The interface or 'Installation' of the Run/z program is simple and can easily be done by the user. Once completed, any BASIC/Z compiled software can run on the system.

Possibly one of the reasons a large number of inexpensive programs using sophisticated CRT displays have not appeared in the general purpose CP/M marketplace is they would require a different version for each combination of computer, terminal and printer. Higher costs are certainly associated with having to maintain several versions of the same program. This approach may help reduce software costs, hopefully to the level of the vertical marketplace. At the same time it could encourage the development of more, much needed software.

Once the Run/z program is installed, you have the capability of using absolute cursor addressing for both the CRT and the printer. You can have reverse video, blinking video, erase to end-of-line, erase to end-of-screen, clear screen and a unique method for editing your data input at run-time.

These are all controlled by your Basic program and normally by single statements and functions. Of course, the video functions must be available within your terminal.

The editing feature is excellent as many of us make errors when entering data. This gives us a chance to edit before sending the data into the system. With another statement, the existing data can be recalled and changed using the same format as the input editing function.

Using pre-defined control codes or those you define yourself, there are many editing functions available. These are: Non-destructive cursor movements to beginning of line, end of line, left, right; delete left, right, entire line; select Insert mode or Change mode.

You can even mask the input line for a specific character count. The system prints a line of dots on the screen; the dots being replaced with the characters entered. There is never any doubt as to how much room remains on the input line.

String Handling

In today's user friendly environment it's necessary to provide more prompting and error handling in the program structure than it has been since the introduction of the micro. This necessitates storage of string data and the capability of extracting and presenting it in the most efficient manner.

Perhaps one of the biggest differences between BASIC/Z and other modern Basics is that string space is allocated statically instead of dynamically. This means there are no "garbage collection" routines to take up operating time and delay completion of program functions. Further, array space can be dynamically allocated and, after use, erased. The memory formerly occupied by the array is then available for the program.

For those not familiar with these techniques let me take a few lines to explain them both. Dynamic string space allocation provides string space as it's required by the program. Because the system doesn't know how long the string will be, it finds the space required for storing it and sets a pointer to the location.

If your program line reads; A$ = "JOHN Q. JONES", then the system will find a space for the line (13 characters and spaces), and will set a pointer to the line. If a subsequent line reads; A$ = "LARRY P. MURPHY", then the system finds a place for the new line and changes the pointer for A$. However, the space formerly used for A$ isn't available to the program at this time. It isn't until all of the available memory is allocated that the Basic will try to recover any unused space.

It does this by first finding unused space and moving current strings into the space. It then sets new pointers and designates the reclaimed space as available. This can be time consuming as there can be many strings to be moved around. If you're using a routine that requires string exchanges such as sorts, string parsing, etc., they may require the use of this reclaiming procedure many times during the operation of just the one routine.

BASIC/Z does away with this by forcing the programmer to declare the space required for each string and string array. This is quite easily done as there are default line lengths for string variables and it's normally simple to determine the maximum line length you'll have in an array. There are also a few less conventional ways to use string arrays with this Basic but more about that later.

By using this method of allocating string space, A$ will always occupy the same space in memory so there is no need for any reclaiming function and the resultant loss in time. Further, we always know where the string is and can get its address from Basic. This allows us to check characters with PEEKs or change them with POKEs.

BASIC/Z limits the maximum string length to 250 characters which is on a par with most good Basics. Also, since the string length is declared in a DIM statement if other than the default length, it's very easy to use only the amount of string space you need for that particular string variable.

With the addition of functions permitting string parsing and manipulation, we have capability matching or exceeding most major Basics. Some other functions and statements for use with strings are;

UPCASE$, UPCASE, LOWCASE$ and LOWCASE. These permit forcing lower case characters to upper case at the console or individual string level and resetting for lower case when desired.

MID$ can be used for both extraction and insertion of data as a substring within a string.

MIN$ and MAX$ are used for alphanumeric comparisons of strings.

INCHAR$ permits recognition of a single input character. INKEY$ permits acceptance of a specified number of characters and doesn't echo them to the screen. INPUT$ prints a specified number of dots on the screen and allows the line to be edited before pressing RETURN. EDIT$ performs almost the same function except that it displays the existing contents of the string.

VERIFY verifies that all characters of a substring exists within a string without regard to their sequence. INDEX locates a substring within a string.

SPC$ returns a string of a specified number of spaces. REPEAT$ returns a string of one or more characters repeated any number of times.

One of the less conventional ways of using string arrays with this basic is with the use of the single-byte, multi element array. For example, if we DIMensioned an array as A& (1024) we would have an array of 1024 elements, each element representing one (1) character.

By visualizing the array as a string, we would then have a string 1024 characters in length and, by using the built-in statements and functions, we can parse and manipulate this string as easily as a standard string. Further, since we can have a file record length of 1024 characters, System/z saw fit to include functions that automatically parse file records and insert them into our array. Naturally, we can write to the file in this manner as well giving us a very versatile system.

Noteworthy of mention are the functions that normally do so much more when using a compiled language. The efficiency and speed are further improved by building them into the system. An example of these are the built-in SORT and SEARCH functions.

The sort permits sorting ANY array and ANY type array. You may select the starting element, ending element, key length and other options. Sorting is done at the rate of 2,000 elements in 2 seconds according to System/z. We can safely say it's much faster that most Basic sorts.

The SEARCH function is just as versatile as the sort. Besides designating the starting and ending array elements, you can also specify your own relational operators (<,< = , = , = >, >, <>) to be used in the search. This particular function does well in a Keyed-Index environment and, since it requires only a single statement, it fits well in a IF-THEN-ELSE structure.

"all floating point math is done in BCD which avoids the rounding errors associated with systems calculating in binary"

Numeric Data

In most programs the handling of numeric data is extremely important to the final result. Having a floating point number range of 1E − 61 to 1E + 61 with a precision of from six to eighteen digits definitely provides a measure of comfort in math programs. Having the capability of adjusting math precision even while a program is running can be interesting.

In BASIC/Z, all floating point math is done in BCD (Binary coded decimal) which avoids the rounding errors associated with systems calculating in binary. Integer calculations are also done in BCD with their own separately specified precision of from six to eighteen digits. One and two byte binary type variables are also included and, used properly, can improve the efficiency and speed of the system.

The one byte variables are called 'Control' variables and have a range of from 0 to 255. The two byte variables have a range of from 0 to 65535.

By using these for counters, flags, menu selectors, etc., you can not only save memory but time as well. The system doesn't have to make complex calculations as it does with floating point numbers. Because of their contribution to the overall efficiency of the system, several functions that allow their exploitation are included in the Basic.

Functions such as INCR and DECR are included for incrementing or decrementing any numeric value by one (1). Instead of using a less efficient X = X + 1 or X = X − 1, you would use INCR X or DECR X with the same result but a savings in time and code.

File Handling

Perhaps one the most important functions of any Basic is it's ability to read and write data from the disk files. When versatility is provided in this function, you find you're able to become very creative in using it's many features. Most current disk Basics provide the versatility required for most applications; I just think BASIC/Z provides more.

First, up to 30 files can be open at any one time and each with their own file error trap. This way, if a file error is encountered control can be transferred to a statement covering a routine for the individual file.

Second, no declaration for the pre-allocation of buffer space is required as it is in other Basics. BASIC/Z allocates a 128 byte buffer for each open file as each of them is opened.

Third, and perhaps most important, there is no limit to the length of a file record. As long as there is enough space on the disk for the file, it can be successfully opened and used. Further, you can always interrogate the system as the program is running to establish the amount of space left on the disk. This way you can decide dynamically whether or not you have enough space for a file or additional entries.

Files can be of three types: sequential, random and UNFMT random and with their unique attributes, make each preferable for certain applications.

Sequential files allot variable amounts of disk space to each data item and are therefore most suitable for saving text and variable length string data. Because the file may only be written or read sequentially, the records can be packed into the file, one after the other, with the loss of little disk space.

The disadvantage is that you must read and write the file from the beginning each time the file is opened as only one pointer is maintained.

"Sorting is done at the rate of 2,000 elements in 2 seconds"

Still, the capability of using this type of file is valuable. In fact, BASIC/Z has provided a function called SPOOL that allows the saving of text from PRINT statements in a sequential file for printing at a later time.

Random files are probably the type most often used, as the ability to locate any record within the file is required by most programs. In using a random file, you're required to declare (explicitly or dynamically within the program) the record length associated with the file. The record length cannot be changed from that point forward.

Since BASIC/Z does not insert delimiters between logical records, it's necessary to know the record length to know where each record is on the disk. This packing operation maximizes the use of disk space.

As the record length may be of any length, the only limiting factors are the BASIC/Z limit of logical record numbers to 65535 and the CP/M limit of the logical file size to 8 megabytes. Also, when the record length exceeds 250 characters, additional buffer space is assigned for the longest record length used. All files will use the buffer so only one buffer is used regardless of the number of files open.

"the capability of using absolute cursor addressing for both the CRT and the printer"

Another set of pointers is maintained for each random file that allows sequential read and write of the file in addition to random access. Whenever a read or write occurs with use of the RECORD option, the operation will affect only that logical record. If the RECORD option is not included, then the sequential, or indexed operation is assumed.

In this case, the read or write affects the next record pointed to by the PUT or GET pointer for the file. Yes, you can find out where the pointers are pointing and can set either or both to an initial value or anywhere within the file at anytime during program operation.

BASIC/Z maintains an End-of-file pointer for each open file which is equal to the greatest logical record ever written to that file. This remains true unless the logical size of the file has been reduced with the EOF statement. In any case, you cannot access a record beyond the end of file with a GET statement as it will result in an error. Since the EOF pointer is maintained by the system, you can always access it's value and establish the current size of the file.

The capability of blocking and deblocking data to and from the file using an array, the use of separate buffer space, and the designation of a single buffer for the longer records, provides an advantage in reading and writing these files. The result of this is a reduction of time in completing these operations.

For example, if your logical record length for a file is 10 characters you could, using the array concept, read and write several records at one time. You would parse them using the PUTVEC and GETVEC functions of the system. Since disk access is the most time consuming of a disk operation, it is reduced using this method.

Further, BASIC/Z is structured to read as much as possible into as large a buffer as possible automatically, anticipating another read in a subsequent operation. If, for instance, your file had a logical record length of 25 characters and you executed a read of record number 7, BASIC/Z would, if buffer space was available, read records 7, 8, 9, 10, etc., into the buffer.

If you then executed a read of record 8, BASIC/Z would not access the file again as it knows it has record 8 in the buffer. Of course, if you asked it to read record 234 at this time it would probably make a file access if the data wasn't in the buffer. The system waits until the last moment before it writes to the file.

Again, if several record lengths can be held in the buffer, the system would perform write operations to the buffer but not to the file until the buffer had to be emptied. BASIC/Z maintains the system so these file read and write operations are transparent to the user. It's nice to know it exists and how the files are handled.

BASIC/Z's UNFMT random file is really a method of maintaining compatibility with other languages. The format of a BASIC/Z file is slightly different than most others so the UNFMT version has been provided to get around this. It allows you to work directly with files generated by most other Basics or read and rewrite them in the BASIC/Z format. Both are provided; you have only to choose.

The nice part about this is that you could use your program data with your word processor or access your accounting files for reports. In other words, the compatibility feature between systems can help you become very creative.

User Friendliness

The term "User Friendly", has been seen much too often these days with it's meaning too interpretive. To some it only means a few additional prompts; to others, covering for operators mistakes. I prefer a combination of both as the more that is done for the operator and the quality of input data, the better the program will perform.

The ability to edit input strings that was covered earlier is one of the good things we can present to an operator. Another is absolute cursor addressing and screen enhancements.

Using absolute cursor addressing is simple in BASIC/Z as it is done with the TAB function. In most Basic's (BASIC/Z included), the statement; PRINT TAB (20); "A", will write the character A in column 20 on the screen. Using BASIC/Z, the statement; PRINT TAB (12,20); "A", will print the character A in line 12 and column 20 on the screen.

A control array (one byte type) can be generated in the program to produce a 'form' that will be 'filled in' by the operator as data is entered. This method of entry has been used successfully by many commerical software writers.

Reverse video can be accessed with a RVIDEO statement and normal video with NVIDEO. Use BLINK to set the text blinking, NOBLINK to stop it. FORMFEED clears the screen and returns the cursor to home. ERAEOL erases to the end of line and ERAEOS erases to the end of screen.

Of course, these features must be available in the terminal so BASIC/Z can allow you to use them, but it's nice to know you have direct control over them instead of having to write routines to produce the same result.

An extension of user friendliness or perhaps one of its best contributors is the use of extensive error trapping. Too many programs I've used, and a few that I've written, seem to drop dead if I enter an incorrect character, too many characters, too few characters, etc. In fact, anything different from what is expected by the programmer will do it with some programs.

BASIC/Z provides for two (2) levels of error trapping. The first is declared when a file is opened and causes a GOTO Line # (or label) whenever a disk error is encountered. The second, a type of global error trap, effects a GOTO Line # (or label) whenever an error other than disk (or disk if an error trap hasn't been declared) is encountered.

The nice thing about these is that BDOS errors in CP/M which are normally fatal are also trapped by the system. With the use of the traps the errors can be handled in exactly the manner you desire.

Another error generated by some programs is caused by the programmer's inability to query the system forcing error causing activities. For example, your error handling routine requires that you close a file but the file hasn't been opened successfully, you may generate an error when you attempt to close the file.

In some cases telling the system to close the file causes a 'File not open' error that takes you back to the same error trap, and now you're caught in a loop. With BASIC/Z, you can query the status of the file and take appropriate measures.

You can also get the filename of each open file, the default drive and the number for the last file specified in a statement. You can easily determine the space left on disk or in memory and a few other things that will help prevent causing your own errors.

Other Goodies and Delights

Keywords such as CONSOLE, LPRINTER, NULL, ECHO and SPOOL are used to direct the output from PRINT statements. CONSOLE and LPRINTER direct output to either device and ECHO provides output to the console and printer simultaneously. NULL supresses all output from print statements and SPOOL directs the output to a sequential file.

FMT provides the formatting of numerics required for columnar printing but, unlike most of the PRINT USING statements in other Basics, can format numerics regardless of whether they're being printed or written to file. The format statement can be used to right-justify a numeric and place it into a string variable. By having it justified in a string, it becomes a simple matter to pick off a number using a LEFT$, RIGHT$ or MID$ function.

"up to 30 files can be open at any one time and each with their own file error trap"

Printhead position is returned via the PCOL function and the print line is returned via PLINE. It's quite easy to control paging with this information. CCOL and CLINE provide similar functions for the console. As these are separate pointers, a console print statement won't change the pointer for the printer. Also, when spooling to a file, the system will assume line 1, column 1 for the start of the spooling operation without affecting the other column and line pointers.

SELECT and SELECT$ specify or return the default drive for the system. SETUP returns the base address for the user configuration area so you may get specific information about the hardware. This is the information that was entered when installing the run-time module.

CHAINing with COMMON variables and arrays is an excellent feature as with all of this error trapping and prompting capability, you may find your program became too large for memory. The CHAIN statement allows you to break it up and call in the program (or program segment) you need when it's required.

COMMON permits you to pass the contents of any variables or arrays between the CHAINed programs without losing data. In BASIC/Z, the COMMON would be used in place of DIM when dimensioning arrays or variables if the data is to be common between segments.

UPCASE forces all alphabetic console input to upper case. This emulates the CAP-LOCK feature of most keyboards. LOWCASE returns it to normal. UPCASE$ converts a string to upper case; LOWCASE$ converts a string to lower case.

This is very convenient for sorts and searches if the data retained in the file is in mixed case. It is then easily converted to upper case and placed into an array for the sort or search operation.

One of the best things I've found in BASIC/Z is the use of labels as well as line numbers in GOTO, GOSUB, and RESTORE statements. In other words, a label can be used wherever a line number would be referenced. Because I tend to use many of the same subroutines in my programs, it is a simple matter to merge the subroutines into the program and call them by label, without regard for their actual position in the program. As long as I don't duplicate line numbers during the merge (I'm careful and only do it sometimes!), the scheme saves a lot of time and debugging.

"no garbage collection routines to take up operating time"

As the labels can be any length up to 250 characters, they can easily replace the REMarks statement I'd usually use to describe the subroutine. Labels always start with a "@" character so a typical label in one of my programs may be; @PRINT.THE.HEADING.FOR.THE.REPORT or @COMPUTE.INTEREST.

Variable names may also be up to 250 characters in length and with all characters significant. You can get spoiled with this feature as the program becomes so well documented you expect to find it in things you wrote before you got BASIC/Z.

Since the result of the program is compiled, the length of the variable name doesn't affect the resultant code and it really helps you to remember how the program works; especially if you get into trouble.

Speaking of trouble, there is the handiest DEBUG system built into BASIC/Z. You can either debug in the TRACE mode where each line number is displayed as it's executed or, you can use the SINGLE-STEP mode where the program will pause after executing a statement. You can then examine up to four simple variables. The DEBUG affects the group of line numbers you specify so you don't have to single step through loops to get to the part you wish to work with.

My preference is to put a lot of print and input statements in the program to duplicate this but over a broader range of variables if I expect trouble. The DEBUG statement is a compiler option so it won't show up in the compiled program unless you tell BASIC/Z at compile time.

Many loop formats are available to the programmer. Besides a multiple statement loop, you can have a FOR-NEXT loop with the STEP option, A DO-UNTIL loop and a WHILE-WEND loop. This should satisfy many programming conventions.

PUSH and POP are used to place a line number or label onto the subroutine stack or remove one. I've used POP in a loop to clear the subroutine stack when I reenter the main menu in a program. This helps prevent embarrassment if a return to an unanswered subroutine call is encountered.

User-defined functions are something I've stayed away from in interpreter Basics. They always seem to take a lot of time executing and their only advantage is that your're able to pass data into them for execution. Because of the disadvantages, I've used them only occasionally with the interpreter.

Using BASIC/Z has changed my approach to these as they execute quite rapidly and offer some good advantages. For one, BASIC/Z permits multiple line functions. You can therefore write a rather long mini-program, pass data into it and get data returned. Another is that data can be passed as a literal, an expression, a constant or a variable.

You can reference both global variables and those used within the function (dummy variables) with equal ease. You can also call other functions from that function so you can be flexible in your program design. The local variables remain in effect for each individual function so the variables used in one have no effect on the others.

These functions are called recursive as they can call themselves and pass variables back into themselves.

Assembly language subroutines can be defined and linkage is provided. These functions can also be redefined dynamically at any point in the program.

With the MEMEND statement, you can set the logical end of memory to provide room for the assembly language function. The ENDMEM function returns the highest byte you can use in the system. The statement; MEMEND ENDMEM - 1500 is therefore both legal and useful. As RUN/Z overlays part of CP/M, some of the area normally occupied by the operating system is available for Basic and for the user.

The Editor

BASIC/Z's editor is one of the best I've used and is certainly one of the easiest to learn. As each line in the program has a line number, it is just a simple E1200 that brings line 1200 to you for editing. With that command, the line is displayed, cursor set at the beginning and, you're ready to go. Yes, even the line number can be changed so the line can be copied or duplicated elsewhere in the program.

Automatic line numbering, directory display, file and buffer information display, renumbering and the listing of the program on the screen or printer are provided with simple commands. LOAD, SAVE, RESAVE and MERGE handle the program files.

You can issue commands directly to the printer, set the length and width of a logical page, write a heading for the printout and execute a program directly from the keyboard.

The best thing (for me, at least) is that all of your program input is syntax checked before it is allowed in the buffer. This prevents your loss of valuable time trying to compile a program with syntax errors or receiving a rejection at run-time.

System/z seems to have put a lot of thought into the program as they've even considered the possibility you may wish to use programs written for other Basics with BASIC/Z. Provisions have been made for the BASIC/Z editor to accept any programs it can read. It will usually read them if they are in the BASIC/Z format or in ASCII.

Most Basics will write the programs out on disk in ASCII (as an option with some); this can be read by BASIC/Z and syntax checked during the read operation. When a syntax error is encountered, BASIC/Z will mark the line as a REMARK and, with the global search function, the affected lines are easily found and corrected.

As BASIC/Z requires line numbers for operation, programs from editors not requiring line numbers can be assigned line numbers as they're read into the system by using an option available to the LOAD command. The BASIC/Z ability to use programs written for other Basics saves much typing and testing.

The Manual

The manual is complete and offers the reader a good explanation of the various features of the system. It does, however, leave a bit to be desired in the way of examples and details or suggestions in the application.

I'm sure the writer anticipated the use of the compiler system by experienced programmers and offered them very thorough knowledge of it's operation. I personally feel it is up to it's writer to provide more than just basic (but detailed) information but also guide the user into getting the most out of the system. Perhaps there's room for a user's guide here.

The manual, unlike some others we've all seen, covers all of the features of the system.

Summary

As a language, BASIC/Z offers much more than any other Basic I've used. As a compiler it offers excellent operating speed and reliability. I have not tested it against others for speed or code generation because of the many unique features and the extended capability of the system. I don't see either as a problem.

The experienced programmer will enjoy working with the system and the many statements and functions (See fig 1) available for development into finished applications. The less experienced programmer may have some trouble getting used to a compiler if an interpreter is presently used but, the resultant program, with all of the screen enhancements available, will be very professional in appearance and operation.

For those in the commercial software business, there are schemes built into the system to provide security and System/z doesn't charge for the distribution of software written with the use of BASIC/Z. The run-time package and it's installation program can also be freely distributed as long as it's accompanied by the Basic/z compiled programs. In other words, no royalties, run-time charges or other charges for distribution. These are very good reasons to investigate further.

Additional information can be provided by:

daman

DAMAN Suite 14, 3322 Memorial Parkway, S.W., Huntsville, AL 35801 (205) 883-8113

Computer Programs • Systems Enhancements • Data Management • Support Services

EXECUTIVE COMMANDS

ATTRS	CONFIG	EDIT	LIST	PAGESIZE	SCRATCH
AUTO	DEC	EXEC	LISTP	PRINTER*	SEARCH
BIN	DELETE	FILE	LOAD	RENUM	SEARCHP
CHANGE	DISPLAY	FORMFEED	LOWCASE	RESAVE	SELECT
CLEAR	DISPLAYP	HEADER	MERGE	RESET	TITLE
COMPILE	DOS	HEX	OCT	SAVE	UPCASE

LOCAL EDIT COMMANDS

ABORT	BACKSPACE	END	LIST	SEARCH
ADVANCE	CHANGE	INSERT	QUIT	TAB
APPEND	DELETE	KILL	REPLACE	ZAP

RESERVED WORDS

ABS	DIM	GETSEEK	MIN	READ	STEP
AND	DDIM	GETSEQ	MIN$	RECGET	STOP
ASC	DISPLAY	GETVEC	MOD	RECLEN	STR$
ATN	DO	GOSUB	NAME	RECORD	STRING
ATTR	ECHO	GOTO	NEXT	RECPUT	STRING$
ATTRS	EDIT$	HEX$	NOBLINK	RECSIZE	TAB
BIN$	EOF	IF	NOT	REM	TAN
BLINK	ELSE	IMP	NULL	RENAME	THEN
CCOL	END	IN	NVIDEO	REPEAT$	TO
CHAIN	ENDMEM	INCHAR$	OCT$	RESET	UNFMT
CHAR$	EQV	INCR	ON ERROR	RESTORE	UNTIL
CLEAR	ERAEOL	INDEX	ON GOSUB	RETURN	UPCASE
CLINE	ERAEOS	INKEY$	ON GOTO	RIGHT$	UPCASE$
CLOSE	ERASE	INPUT	OPEN	RND	VAL
COMMAND$	ERR	INPUT$	OR	RUN	VARADDR
COMMON	ERR$	INSTAT	OUT	RVIDEO	VERIFY
CONSOLE	ERRLINE	INT	PAGESIZE	SAVE	WEND
COS	ERROR	LABEL	PCOL	SCRATCH	WHILE
CREATE	EXP	LASTFILE	PEEK	SEARCH	XOR
CVTC	FA	LEFT$	PEEKWORD	SELECT	+
CVTC$	FILL	LEN	PI	SELECT$	−
CVTCD	FILLSPC	LET	PLINE	SERIAL	*
CVTCD$	FIX	LINK	POKE	SETUP	/
CVTI	FMT	LN	POKEWORD	SGN	\
CVTI$	FN	LOAD	POP	SIN	^
CVTR	FNEND	LOG	PRINT	SIZE	=
CVTR$	FNEND$	LOWCASE	PUSH	SIZE$	>
DATA	FOR	LOWCASE$	PUT	SORT	<
DEBUG	FORMFEED	LPRINTER	PUTNUM	SPACELEFT	>=
DECR	FRAC	MAX	PUTSEEK	SPC$	<=
DEF FA	FREE	MAX$	PUTSEQ	SPOOL	<>
DEF FN	GET	MEMEND	PUTVEC	SQR	'
DEGREES	GETNUM	MID$	RADIANS	STATUS	!

FIRST CLASS MAIL

U.S. POSTAGE

HUNTSVILLE AL MAR 7 PM 1983

FIRST CLASS MAIL

DAMAN
Suite 14, 3322 S. Memorial Pkwy.
Huntsville AL 35801
(205) 883-8113

FIRST CLASS MAIL