MICROPOLIS USERS GROUP

MUG Newsletter # 12 - July 1981

****

MORE ON THE FMT FUNCTION

by Burks A. Smith of DATASMITH
Box 8036, Shawnee Mission, KS 66208

Over the past few months, several members have written in with tips on using the Micropolis FMT function, so perhaps it is time to put all of the various uses of FMT in one place for the benefit on everyone.

All versions of BASIC that are even moderately powerful have some means of formatting output, because it is the only way that numbers can be printed in a right-justified manner, among other things. The FMT function is the Micropolis substitute for the PRINT USING statement in other BASICs, but is unique in that it offers tremendous versatility. Like PRINT USING, FMT formats numeric values according to a "picture" contained in a string variable or constant. The "picture" is simply a string of characters which represent the position of the decimal point, commas, dollar signs, etc. in coded form. Thus:

In MICROSOFT BASIC, the statement:
```
PRINT USING "$$$#.##";A
```

is equivalent to MICROPOLIS BASIC:
```
PRINT FMT(A,"$$$9V.99")
```

In the above example, both statements print the value of A in a format which includes two decimal places and a dollar sign that appears immediately to the left of the most significant digit. The big difference is that FMT is a general-purpose function for converting numeric values to string values while PRINT USING is useful only for the PRINT statement. This may seem like a small distinction on the surface, but is extremely powerful. For example, if you are PRINTing a report that is composed of several columns of figures that require different formats, you only need one PRINT statement with several different FMT functions being printed on the same line. However, if you were using another BASIC that only had the PRINT USING statement, you would need a separate statement for each column, because PRINT USING applies the same "picture" string to every value being printed with that statement.

The rules for the use of the FMT function appear in section 5-20 of the MICROPOLIS manual, so I won't repeat them here. Briefly, the "picture" string is composed of various combinations of the characters "9", "Z", "V", "$", "*", and ",", which have special meanings in the context of the FMT function. The "9" is replaced by a digit in the number to be formatted, or by a zero to make leading zeros. The "Z" is replaced by a digit in the number to be formatted, or by a blank to surpress leading zeros. "V" marks decimal point alignment; notice that the period is not a special formatting character, so it becomes part of the formatted string wherever you put it (usually next to the V).

The general-purpose nature of the FMT function allows converting numeric values into formatted strings in a useful variety of ways. For example:

```
For dates:   FMT(31581,"99-99-99")
                    returns "03-15-81"
       and   FMT(31581,"Z9/99/99")
                    returns " 3/15/81"
```

(Continued on page 8)

RENEWAL TIME

This might be your last MUG newsletter. Take a look at the date on the mailing label. If it reads 0781 then this is the end unless you renew your membership. Cash rates: U.S., Canada, Mexico; $18/year: Other, $25/year. Must be U.S. dollars. VISA and MASTERCARD accepted at 104% of cash rates.

.....

BASIC/S COMPILER

I imagine that most of you received the flier from Systemation. It seems that it's here. Honest to gosh. And does it ever look nice. What is IT? Systemation's compiler, of course!

Actually, the system, known as BASIC/S, is incorrectly referred to as just a compiler. Compiling it does, and more. It edits, it error checks, it debugs.

Let's walk through a programming session. You enter BASIC/S by typing its name. Statements are entered by using the edit mode of BASIC/S. So far, everything is just like you're used to doing in Micropolis BASIC.

The first difference you'll note is the power of the edit mode. It is very similar to Systemation's BEM program. Automatic line numbering, global search, search and replace, paginated listings with user defined headers - lots of new power if your reference is using Micropolis BASIC.

Then comes the first surprise. You can't enter a line with improper syntax. Try typing "FR" for "FOR" or "MEXT" for "NEXT". No go. BASIC/S tests each line as you type it and rejects syntactically incorrect input. Now that's nice. Most systems (every system in my experience) would make you save the file, then run the compiler, which would then tell you of the typing error.

Variable names can be any length. No longer do you have to use "N$,A$,C$,Z for "NAME$,ADDRESS$, CITYST$,ZIP". Just as you are accustomed to, the names of real variables have no suffix, integers have the % suffix, strings have the $. In fact, in general (though not without exception) everything is as you are accustomed to. All your current programs should be easily convertable to BASIC/S.

When you have finished typing in your program, you can compile it. Compiling, for those of you not acquainted with it, is the total, one-time translation for the English language type of statements, which you can read efficiently, into machine language, which the computer can read efficiently.

The Micropolis BASIC interpreter does the same sort of translation, but it does it as the program runs - each and every time the program runs. It takes a fair amount of time to do the translation. Having the English language statements and the translator in memory at the same time as the execution routines also takes up a lot of memory space. This time and space is saved when you compile the "source" program into a "load" program.

The "load" program is executed under a part of BASIC/S named RUN/S. RUN/S overlays MDOS but uses RES. That's transparent to the user. Under MDOS you type RUN/S "program name". The RUN/S module loads; it loads and then executes the program, and reloads MDOS when it's done.

For those of you who may want to develop software, Systemation's policy on sale of object (load) programs is interesting. Unlike Microsoft, Systemation's policy has no royalty payments. No bookkeeping of how many programs at what price. No temptation to bend or break a hard-to-police rule - which you know a lot of other people break.

(Continued on page 7)

LANGUAGES

by Marc Lewis
PSC Box 55, APO NY 09611

I would like to address the subject of languages, and re-open the can of worms that was opened about 30-odd years ago when high level languages (HLLs) were first developed. In the December newsletter, Jim Harden innocently asked about Pascal and BASIC. I would like to comment on and add to Buzz's answer.

Well, Virginia, there is no Santa Claus. Which is another way of saying, there is no ONE perfect language. BASIC comes close in a lot of ways. Starting with BASIC, we have the "Basic Algebraic Symbolic Interpreter Compiler" or "Beginner's All-purpose Instruction Code", depending on which page you look at in the dictionary (mine is the one by Charles Sippl, which has both). Basic has been with us for some 20-odd years, and my guess is that Professors Kemeny and Kurtz wouldn't recognize their brainchild now.

According to my aforementioned dictionary, BASIC resembles Fortran II. That should give an idea about its age. However, the keywords in the acronym, whichever one you use, are Basic/Beginner's, and All-purpose. BASIC was designed with the specific intent of having a language to teach programming in that was reasonably simple, without requiring mental gymnastics. It does this very well. BASIC has come a long way. We have S-BASIC which looks like Pascal, we have BASIC-80/MBASIC, which I consider to be a masterpiece, since you can debug on-line, then compile. We have BASIC-E, which is supposedly free when you buy CP/M. We have North-star, IMSAI, MITS, PolyMorphics A00, Tiny BASIC, Extended Disk BASIC, and hundreds more.

Buzz's comment that "All the hotshots that have Pascal and FORTH still have BASIC" is true. Damned right! Because you can't buy a major piece of computer hardware these days without BASIC. A case in point. My Computer Science "instructor" is a senior programmer and wants a micro to play with at home. Given his requirements, I was forced to recommend the Apple. He wanted Fortran, color graphics, voice synthesis and price tag under $5K. To get all this, he is upset that not only does he have to pay extra for Fortran, but has to get Pascal to get Fortran and he gets a BASIC in ROM and another on disk! Like I said, you can't hardly buy a computer without it. That's why the hotshots have it. That's not to say they use it, though.

BASIC is easy to use. I like it. It's also fun sometimes. I have been learning Fortran on a PDP11/03 system. It's a whole different world. Fortran used to be THE language for programming, and is still sort of a Rosetta Stone. A professional programmer must have some familiarity with it. The Cray-1 is programmable in Fortran IV and Assembly ONLY. Fortran stinks for the way I program. Most all of my programs are I/O intensive, meaning I like chatty programs. This tends to fall theough the cracks in the hundreds of FORMAT statements I have to write all the time. I even go so far as to test my logic in BASIC, on the interpreter, before I code it in Fortran. There are few languages that can outrun Fortran for number-crunching, though. Fortran has its place.

My wife is a CPA, and wants to open her own business when I get out of the Air Force. She will need Cobol, if only to retain some compatibility with the rest of the business world. Cobol's strong point is its capability to handle enormous volumes of output on a routine basis. BASIC can handle it, but not as well as the specialized routines in Cobol.

I recently watched an 18-year high-school kid write a LIFE program in Apple Pascal in less than an hour, and it works with no debugging needed. Try that in BASIC. It's pretty slow, but it works. Pascal was designed by "one of those computer scientists". He (Nikolaus Wirth) believes in his brainchild. I don't care for it much, which feeling is shared by a growing number of early converts. Pascal is useful for writing algorithms in but there is some question that it might not have gotten the foothold it has without the boost from UCSD.

FORTH has its proponents as well. Currently, it also has the distinction of being the only other HLL than BASIC to be available for MDOS, as opposed to CP/M. Not being able to even read it has caused me some difficulty, but it has been getting enough good press to make me curious enough to plunk down $150 sometime in the near future. Imagine, an interpreted language with the speed of a compiled language! I swear I'm going to write a Fortran interpreter someday...

The last "language" I intend to mention is assembly language. The nicest thing about assembly language is that no matter what machine you are working on, if the problem can be solved on a computer, you can write the program in assembly language. That tremendous power carries a bit of a difficulty with it however. It tends to take longer to learn, to write in, and to debug. Ya get whatcha pay for, and there is no such thing as a free lunch.

This is not intended to especially espouse any language in particular, but more with an eye to expanding the horizons of probable newcomers to the field, and to remind all those buyers of packaged hardware and software (as opposed to phreaques like me) that there is a huge world out there in personal/home/small business computing. I don't think there is any room in a group of people like MUG for blindness to the capabilities of their very expensive equipment. BASIC, like I said, is a good beginner's language, because that's what it was designed for.

I didn't mention other more specialized languages like Pilot, which is very specifically a CAI language. I didn't mention much about C or tiny c (not a typo), which is apparently an easier to use language than Pascal, while retaining most of its characteristics. I didn't mention LISP (LISt Processor), which is widely used for AI research, as well as Microsoft's incredible MuMath/MuSimp package.

There is much more out there than BASIC, and one of the reasons I bought an S-100 system, followed by a Micropolis disk subsystem is that I really want to explore the world of programming. I have been playing with BASIC for four years now. The only difficulties I am having adjusting to Micropolis BASIC are the LEFT$/RIGHT$/MID$ functions and all the disk functions, because none of these were available in the PolyMorphic BASIC A00. I like BASIC. I think it might just replace Fortran as the computer Rosetta Stone if it hasn't already. But it's not all there is to programming a computer.

.....

O.K., Marc, You win. I didn't mean to try and brainwash the members. Here are some options.

.....

FORTH COMES TO MICROPOLIS

The FORTH language features the built-in advantages of high speed execution, minimal memory requirements and ability to store more data in less space than other languages. Now ACROPOLIS has written a special version of FORTH specifically for Micropolis users.

A-FORTH operates as its own compiler allowing you to compile directly from the keyboard as well as source files on your disks. A-FORTH compiler security traps errors by insuring all control structures are completed within your program.

A-FORTH is an interactive language that allows creating and testing programs right from the

keyboard without creating a disk file. This user-oriented keyboard testing is interpretive in nature to aid in debugging your program.

A-FORTH is a structured language so your programs are more readable. It is completely extensible so you can add your own commands to the language.

A-FORTH is up to 50+ times faster than Micropolis Basic. Even faster execution (up to 600+ times) is possible for time critical applications by using A-FORTH's multi-level language feature to program in assembler. A-FORTH also features an 8080/8085 macro-assembler that allows you to use any mixture of A-FORTH and assembly code you want in a single program. You can even use A-FORTH's macro-assembler to create a whole new set of instructions for your machine. A-FORTH's macro-assembler contains control structures for forward and backward references and is provided in source form so you can make your own modifications.

ACROPOLIS sponsors an A-FORTH contributors software applications library and provides A-FORTH updates & patches at no charge for one full year. Special features are:

- Supports system printer using standard MDOS assigned statements
- Enhanced disk procedures to reduce response & compiling time & # of disk accesses
- Physical disk support for disk diagnostics and disk copy
- Allows access to MDOS file directory
- Can be set up for multiple tasks and users

A-FORTH is available on MICROPOLIS compatible diskettes for $150.00. For more information write or call; ACROPOLIS, 17453 Via Valencia, San Lorenzo, CA 94580; (415) 276-6050. Does NOT require CP/M.

.....

NEVADA COBOL

Nevada COBOL is simple, easy to use and learn, and very fast. It brings to you and your micro all the power and simplicity of the most universal business language of all - COBOL. It's ideal for use on small computers with simple operating systems since all the facilities you need are provided in a single integrated package. All you need, in addition to the Nevada COBOL compiler and Run Time package is a text editor (one comes with CP/M) to prepare the source program.

Compiling rate upto 10X faster than some major competitors.

Requires approximately 8K bytes exclusive of the operating system and table space. Consists of several overlays each executed in turn in the same memory space.

"Two-pass": the source code is read while an intermediate file is written, then the intermediate file is read and the generated object code is written.

Written in 8080 assembly language using minimal memory (16K RAM).

Very fast. Generates in-line machine language object code and subroutine calls (threaded code).

Handles functions such as arithmetic, string manipulation and editing in addition to managing sequential and relative files. Object code generated by the compiler is NOT interpreted.

Ellis Computing supplies an easy to read COBOL Reference Manual describing in detail how to write a program, compile it, load it into memory and execute it. Includes many examples and 10 complete sample programs showing program to program linkage as well as all file types.

COBOL contains English-like phraseology making it easy to understand and learn. Unique error-finding messages are also in English, making programming mistakes easy to detect and correct. The Big surprise to most is that COBOL is as easy to learn, if not easier to learn, than BASIC.

Designed with the most widely used facilities of ANSI-74, Nevada COBOL delivers a broad band of computational power, fast compile and run time, yet consumes only a minimum of 16K RAM. A growing number of Nevada COBOL professional customers are developing COBOL application object code programs for sale to third parties. Unlike many competitors, Ellis Computing allows sale and distribution of such software developments on a royalty-free basis.

Nevada COBOL is available from Ellis Computing, 600 41st Ave, San Francisco, CA 94121, (415) 751-1522 for $99.95 on MOD II! Requires CP/M.

.....

CP/M TECHNICAL TIPS

by S. Tattersall, ITT
London Road, Harlow, Essex, England CM179NA

A PHANTOM SUBMIT COMMAND

The SUBMIT Transient command is used to perform indirect command line processing, or batch style processing. To use this feature a file has to be created using ED, or any other ASCII output editor, consisting of CP/M command lines, e.g.

```
RUN PROGRAM
DIR A:
PIP B:=A:*.DAT
```

The file created must have a file extension of SUB.

When SUBMIT BATCHPG is entered on the console the file BATCHPG.SUB is read from the disk and converted into command lines by SUBMIT.COM. A temporary file is created named $$$.SUB containing these command lines, e.g.

```
0A 52 55 4E 20 50 52 4F 47 52 41 40 00 24
|   R  U  N     P  R  O  G  R  A  M  |  $
|          Command line               |
Length of command line                |
                          End of command line
```

This temporary file is executed, on warm start, by CCP: executing and removing one command line at a time. A '.SUB' file may call another '.SUB' file enabling the chaining of command line files, e.g.

```
RUN PROGRAM
DIR A:
PIP B:=A:*.DAT
SUBMIT BATCH
```

If when, say, a CBASIC program needs to alter the $$$.SUB file during processing, a simple routine can be inserted into the CBASIC program to allow this to occur.

```
COMMAND.LINE$="SUBMIT BATCHPG"
LEN.COMMAND%=LEN(COMMAND.LINE$)
CREATE "$$$.SUB" AS 1
PRINT USING "&"[illegible];CHR$(LEN.COMMAND%)+COMMAND\
  .LINE$+CHR$(0[illegible]$"
CLOSE 1
STOP
```

This routine is useful for running CBASIC menu programs which require different mixtures of CBASIC and machine level programs, depending on the option chosen.

If a $$$.SUB file is present on the disk it will be automatically executed on warm start whether, or not, a SUBMIT command was used to create it.

.....

READING DIR FROM BASIC

by Ed Burkhardt
Box 97, Mequon, WI 53092

The two programs listed below are an attempt to answer Ken Findlay's problem. They may not be exactly what he had in mind, but may find some use as a general utility for those members using a memory mapped video terminal. I haven't been able to determine if this approach is possible with a serial terminal, such as my Visual 200, but will continue to play around and may come up with something of merit.

The principle is to write the DIR directory to the screen and then read it back off the screen. The first listing sorts the directory and lists it back to the screen. If you want to print a label for your disk jacket, replace lines 400-420 with the second set of lines numbered 400-500. Be careful of the backslash, "\", in lines 250 and 350 (and 460 of the print code). It's a "\", not a "/".

Title: DISKMENU

```
   10     ! DIRECTORY - PROGRAM TO READ DISK DIRECT
          ORY AND PLACE INTO A SORTED ARRAY.  PROGR
          AMS MAY THEN BE RUN FROM A MENU SELECTION
          .  ARRANGED FOR SOL-20 TERMINAL WITH VIDE
          O MAPPED MEMORY.
   20     ! By E. G. Burkhardt - Box 97 - Mequon, W
          I - 53092
   30     !
   40     ! THE NEXT THREE LINES SET VARIABLES
   41     ! WHICH ARE UNIQUE TO THE SOL.
   50     ! REASSIGN THEM FOR YOUR SYSTEM.
   60     !
   70     A%=11: ! CLEAR SCREEN
   80     B%=252: ! KEYBOARD DATA INPUT PORT
   90     L%=16RCC00: ! START OF VIDEO MEMORY
  100     !
  110     PRINT CHAR$(A%):! CLEAR SCREEN
  120     PRINT:PRINT"THIS PROGRAM WILL TAKE SEVERA
          L SECONDS TO READ A DISK DIRECTORY, STORE
          IT INTO AN ARRAY AND SORT IT.  PLEASE BE
          PATIENT.":PRINT:INPUT"  WHICH DRIVE IS TO
          BE USED ";G$
  130     DIM D$(80,10):K%=0:N%=0
  140     PRINT CHAR$(A%):DISPLAY G$+":"+"DIR"
  150     FORI%=L%TOL%+1024:J%=PEEK(I%):IF N%=1 GOT
          O190
  160         IFJ%<>32GOTO190
  170   > NEXTI%
  180     K%=K%-1:GOTO230
  190   > IFJ%=32 N%=0:K%=K%+1:GOTO170
  200     N%=1:D$(K%)=D$(K%)+CHAR$(J%)
  210     GOTO170
  220     ! SORT FILES
  230   > PRINT CHAR$(A%):PRINT:PRINT ".....SORTING
          , PASS ";
  240     M%=K%:H%=K%
  250   > M%=M%\2:C=C+1:PRINTC;
  260     IFM%=0 GOTO 350
  270     J%=1:K%=H%-M%
  280   > I%=J%
  290   > L%=I%+M%:IFD$(I%)<D$(L%)THEN320
  300     D$=D$(I%):D$(I%)=D$(L%):D$(L%)=D$:I%=I%-M
          %:IFI%<1THEN320
  310     GOTO290
  320   > J%=J%+1:IFJ%>K%THEN250
  330     GOTO280
  340     ! DISPLAY SORTED DIRECTORY
  350   > H%=H%-1:L%=(H%)\4:IF4*L%<(H%)THENL%=L%+1
  360     PRINTCHAR$(A%):PRINT:PRINT:FORI%=1TOL%:FO
          RJ%=0TO3:IF(I%+L%*J%)>(H%)GOTO380
  370             PRINT TAB(15*J%)CHAR$(I%+L%*J%+64
                  );"- ";D$(I%+L%*J%);
  380   >       NEXTJ%:PRINT
  390     NEXT I%
  400     PRINT:PRINT TAB(20)"ENTER YOUR CHOICE:  "
          ;
  410   > A$=CHAR$(IN(B%)):IFA$<"A"ORA$>CHAR$(H%+64
          )THEN410
  420     PLOADG G$+":"+D$(ASC(A$)-64)
```

Title: LABELS

```
  400     ! REPLACEMENT CODE FOR PRINTING LABELS
  410     !
  420     PRINT:PRINT "PREPARE PRINTER FOR PRINTING
          LABELS....":PRINT
  430     INPUT "  ENTER NAME OR NUMBER FOR DISK ID
          ENTIFICATION:  ";A$
  440     OPEN 9 "*P"
  450     PUT 9 TAB(15)"****   DISK:   ";A$;"   ***
          *":PUT 9
  460     L%=(H%)\5:IF 5*L%<H% THEN L%=L%+1
  470     FORI%=1TOL%:FORJ%=0TO4:IF(L%*J%)>H%GOTO49
          0: ! STARTING ARRY AT 1 ELIMINATES D$(0),
          WHICH IS "DIR"
  480             PUT 9 TAB(13*J%)D$(I%+J%*L%);
  490   >     NEXTJ%:PUT 9
  500     NEXTI%
```

.....

BASIC PROGRAMMING

As discussed several months ago, it seems that there are a lot of you who don't understand programming, both in general, and in the specifics of Micropolis BASIC. After discussing the situation with a few of the MUG members, I've decided to "teach" by developing a program, rather than by going through BASIC a statement at a time.

This method has several advantages. First, the concept of structured programming is addressed. Somewhat tied to the structured programming is the use of a Subroutine Library. Second, you end up with a Mailing List program that you can use to sell the services of your computer. Three, the program uses two logical records per Micropolis physical record. The algorithms necessary for support of such a data structure are therefore included. You should be able to expand the technique to service 3, 4 or more logical records per physical record.

This system is not meant to replace things like the Bonjoel, CCA, or Land data base manager. This will have a much more limited use. In fact, it is very restricted.

This is one of the decisions a programmer must make. Do you design a very restricted system that is very efficient in processing or do you have a versatile system that is relatively extravagent in use of the operator's time and execution time. Versatile systems won't need reprogramming when new data contains something different. A restrictive system would. Versatile systems also take longer to program in the first place.

I know this works for most mailing list applications. Perhaps 75% of our data processing is done by this set of programs, although there are many variations of the print program. So you can make money with this. Well, you can if you can type, or are lucky enough, as in my case, to have Lynn, my wife, and Tammy, my daughter, who do.

Next month we'll discuss the processing and data requirements of a Mailing List system.

.....

BASIC SPEEDUP TECHNIQUES

The slick magazines, in the last few months, have had several articles which state that loops run faster if the loop variable is integer, rather than real. To see if this holds true for Micropolis BASIC, I ran the following two routines.

```
05 ! REAL LOOP VARIABLE
10 FOR I=1 TO 1000
20 NEXT I
30 END
```

Real Loop Variable execution time: 8.2 seconds (SOL)

```
05 ! INTEGER LOOP VARIABLES
10 FOR I%=1 TO 1000
20 NEXT I%
30 END
```

Integer Loop Variable execution time: 6.1 seconds (SOL)

That calculates out to a 25% decrease in execution time for the integer loop. Pretty impressive.

We've all heard of the benefits of crunching the code. Does it work? To find out I ran the set of programs again, but with the commands on one line.

```
10FORI=1TO1000:NEXTI:END
```
Execution time: 7.7 seconds (SOL)

```
10FORI%=1TO1000:NEXTI%:END
```
Execution time: 5.7 seconds (SOL)

Indeed, both cases executed faster when crunched.

Although crunching always reduces memory requirements, there are times when execution speed isn't increased. I have a print routine for the S/W Vendors Directory that runs at the same speed, whether crunched or uncrunched. There must be some instructions that like to see blanks.

.....

S/W AVAILABLE

BANKING

by Steven Guralnick
375 S Mayfair Ave, Daly City, CA 94015

About six months ago I joined MUG for the principal purpose of making some connections for software. My time as a lawyer is just too limited to try to write application software for our office. The editor of this illustrious publication referred me to Gerry Lenz, 3231 Vinyard #42, Pleasanton CA, 94566, 415-846-8406.

So, I went to Gerry and I said that I needed a good check ledger program. These were the specifications I gave him: that we be able to enter the number for the check, the payee of the check, the date of the check, the amount of the check and a code number, the latter indicationg the account to which the check should be charged. The program must also account for deposits, by amount and data. I also needed prompts, on payroll checks, for State and Federal withholdings, FICA and State disability insurance. When all that is taken care of, then the program must furnish the following information: (a) a complete printout of all checks input, or, at the option of the user, checks for any given period of month(s); (b) a breakdown of all the disbursement codes, for the same payment periods available for check printouts, showing the amount of money charged to each code and totalled; (c) an audit trail so that the user can get a list of all checks charged to a given code.

All that, plus friendliness by the program: i.e., good error trapping and clear screen prompting.

What Gerry has produced is a superb program. We have been using it since January and it has simplified our check ledgers to the point of being trivial. You can enter checks drawn (and deposits) very rapidly and I recommend this program without qualification. The price is $75.00, including shipping and tax. It runs under MDOS, and is furnished for Mod II systems. It may be possible to get a Mod I version.

It is always a pleasure to review application software of this quality.

.....

LETTERS

Buzz,
I received Newsletter #9 and am once again properly impressed. The new format is fantastic. Proper size, presentable, and with type large enough for even these tired old eyes. Each issue comes with more and more useful information. I don't know how you manage to do it, but I hope that the membership appreciates your effort as much as I do.

The continuing effort to explain and implement assembly language programing is a decided benefit to those of us that have limited experience, but a need to know as much as we can master, with limited time. For a long time, I have been involved with the TRS-80 users groups, and I really can appreciate the benefits of information exchange as demonstrated by the contributors to the various newsletters published by these groups. I am convinced that the communication between these members have had a great deal to do with the phenominal success of the TRS-80 Model I (despite the indifference to the Tandy Corp.). These tutorials are of great benefit to me, and the group as a whole.

00-6A MEMORY

The tip on unused memory was of particular interest to me, since I do a great deal of development work on my Sol at home. After checkout, the programs have to be converted to be used on my Vector at work. Naturally, each has a different area of memory set aside for storage of dates, constants, etc. I haven't tried using 0000-006A yet, but I will, soon.

I find it more convenient to store the date as three consecutive bytes (MMDDYY). This allows me to peek significant discrete values for use in building arrays, or conditional branching. I use the FMT function as:

```
D$=FM5(10000*PEEK(MEM1)+100*PEEK(MEM2))+PEEK(MEM3),
"ZZ-ZZ-ZZ")
```

Either method works fine.

Y/N ANSWERS - ANOTHER APPROACH:

I frequently employ the following code:

```
 10 PRINT "ENTER YOUR ANSWER ";
 20 A$=CHAR$(IN(X)):IF A$<>"Y" AND A$<>"N"
    GOTO 20
 30 IF A$="N" GOTO 100
 40 ! Yes answer routine goes here
100 ! No answer routine
```

'X', the port number is 252 for the Sol and 2 for the Vector. Since a Serial Terminal (Visual 200) is used with the Vector and is configured for MARK parity, I need to strip the 7th bit in order to use the CHAR$ function, and the code becomes:

```
A$=CHAR$(IN(2)AND127)
```

The major disadvantage of this technique is that the called port retains the data until another key is depressed. A sequence of input queries as shown above would receive the same answer repeatedly, unless there were operation intervention between questions. I get around this (if I have to) by changing the nature of the question in order that the answer varies from Y/N to 1/2 or A/B.

A FEW TIDBITS:

The archaic END statement - seldom used anymore by contemporary Basics to terminate a program. Occasionaly, when breaking during the development of a program, I add a highest line number with a simple END. After this I can write short reminder notes to myself using even higher line numbers (no REM necessary). Then, when I return to the program, I list the end of the program and erase the lines as I complete the little tasks assigned to myself:

```
10000 END
10001 Don't forget to test variable A in loop
10002 Clean up screen image in 1100-1160
10003 Do I need a SIZES statement?
```

FLIPPY - USE BOTH SIDES OF THE DISK?

Templates are available from a number of sources that allow you to notch and punch the jackets of your disks, permitting you to use both sides of the disk (a hard cardboard home-made template and a one-hole paper punch do the job nicely, thank you). I am aware of two major objections to doing this.

1. The quality of the second surface is inferior and will probably contain flaws. An engineer with one of the major disk manufacturers (who will thank me for not mentioning his company) told me that each disk is certified. If the surface being tested is unsatisfactory, the disk is turned over and the disk is tested on the opposite side. Owing to improved manufacturing techniques, few disks fail certification on the first try. An occassional disk will be sold with a defective 'blind' side, or a disk may pass the testing with the untested side blemished. If this is the case, the condition will probably be discovered while formatting, and will certainly be found out by using Systemation 'DVERIFY' or some other similar utility.

2. The other objection deals with the 'pressure pad' of the drive bearing on a surface containing important data. The engineer mentioned above assured me that this was of little consequence. To my knowledge, I have not lost any data this way in the past year and a half.

I feel confident using two sides of a disk when the disk is used only occasionally (backup, history, program storage). When a disk is resident in my drive for an extended period of time, and running, I use only the primary surface.

Perhaps other members have observations about this.

BASIC REFERENCE BOOKS

Responding to the letter from Richard Herz - Dr. David Lien's 'The BASIC Handbook' has some good pointers on converting between the various Basics. When I have some spare moments, I'll try to compile a list of relationships between Micropolis and Microsoft (the similarity of FORMAT and PRINT USING, etc.). There is much to be said in praise of either Basic but no question about the superiority of Micropolis' file-handling capabilities.

Ed Burkhardt, Dynacom Inc.
Box 97, Mequon, WI 53092

.....

Buzz,
I want to pass along a couple of things I did to my DEVOUT & CBRK routines to make things a little easier.

SYSTEM MODS

The DEVOUT routine is the one that handles the output to the console device and the printer. One of the things it does is change the BACKSPACE character (08), into a BACKARROW character (5F). This is probably considered necessary because the output device might be a printer, but most systems have CRT's as console devices so it is really unnecessary. By replacing the code between 599H and 5A0H with NOP'S, this feature is eliminated, and pressing the BACKSPACE key simply backspaces the curser (on any CRT I've ever seen). As strange as it may seem, pressing the DELETE or the BACKARROW key does the same thing. The calling program must change them into the BACKSPACE character before sending them down.

As for the CBRK routine, it seemed a little clumsy to me to use the CONTROL S to stop a fast scrolling screen. The space bar is a much more convenient target. Just change 541H to 20H and it's done. I'm not sure these changes apply to integrated systems put out by other companies, but I'm sure you would know more about that than me.

Mike Raney
Route 4, Box 115, Greenfield, IN 46140

.....

I tried the changes out. My SOL already knows about backspacing, but I put the mod in anyway. It messed things up. I would therefore recommend that you don't try this one unless you get the / / type of thing on your screen when you backspace. Well, you can try it. No harm done unless you SAVE it.

The spacebar mod worked fine and is a resonable system modification, as far as I can see.

For those of you not familiar with this sort of system modification, here's what you do. Make a DISKCOPY of some disk that has RES and MDOS on it. It will aid the testing if it has lots of additional files on it. Don't ever make system mods on a prime disk until you're sure everything works.

Put the duplicate disk in drive-0. Type FILES and press the CONTROL key and the S key at the same time. This should stop the display of the files. Pressing any other key starts the display up again.

Now type the following information. The ">" is printed by the system, you type the rest, each line concluded with a RETURN.

```
>ENTR 541
>20/
```

See appendix page E-5 for a listing of the code you are modifing. Now type FILES again. This time press the SPACE bar after the display starts. It should stop. As before, pressing any other key starts it back up. CONTROL S will have no effect.

If you like what you have you can save the new system by doing the following. Again, the ">" is from the system and there is a RETURN typed after each line.

```
>TYPE "RES" 0
>SCRATCH "RES"
>SAVE "RES" 2B1 1598 3
```

Now reset your computer and reboot from your newly generated system. If it comes up with MDOS you're in great shape. Try the FILES and the SPACE bar again. If you want to have this be your "normal" system, you'll have to copy your new RES to every disk that RES is on. The easiest way to do it is to boot up under your new system. Now (1) Remove that disk; (2) Put a disk that you want to modify in drive-0; (3) Repeat the set of three commands (TYPE,SCRATCH,SAVE) from above, and; (4) Go back to step (1).

For the backspace mod, type the following:

```
>ENTR 599
>00 00 00 00 00 00 00 00/
```

If you use to get the / / format, you should now just get the curser moved left one position. If, after testing, you want to save it, do it exactly the same way as above (TYPE,SCRATCH,SAVE). See appendix page E-7 for a listing of the code you are modifing.

As long as we're discussing system mods, I'll mention another thing. I recommend that you keep a journal when you play with your system. Log in what you are doing. In this case, record the current values of what is in the system code before you change it. Then you can always go back and rechange it if you want.

To see what is there, type the following:

```
>DUMP 541
```

The system will print (before or after):

```
0541 13
   or
0541 20
```

Type:

```
>DUMP 599 5A0
```

The system will print (before or after):

```
0599 FE 08 C2 A1 05 0E 5F
05A0 00
          or
0599 00 00 00 00 00 00 00
05A0 00
```

Record these values in your log.

.....

NEED FOR UTILITIES

Buzz,
Enclosed is my check for another years subscription to the User's Newsletter. I think that I got something out of every issue. Keep up the good work - it is appreciated. with so many terminal configurations, it seems an impossible task to find some standard but it looks like you are closing in.

I have some thoughts to share on the directions that I would like to go next year. There seems to be any number of Payroll, depreciation schedules and DBMS, etc., for Micropolis, but with the exception of Systemation, Inc., I haven't seen many utilites to aid the programmer. Some of the utilites that I would like to see are listed below:

A very useful enhancement would be a Control P On/Off for the printer. This feature is implemented in CP/M 2.2 and I find it to be very convenient. The Assign and Put statement, although powerful, are very clumsy for debugging and testing.

A screen oriented editor would be a great addition to Micropolis Basic. global search and replace - forward and reverse scrolling would provide a "video window" into the listing. Optional pagination would make this a very good programming tool.

And finally, a screen dump to the printer would be a valuable addition to these utilites. Many times I have wanted a hard copy of part of a listing or the results of run of debugging. This routine is a simple Control D on my other computer, a COMPAL, but I haven't been able to configure one for the Vector Graphic.

I am sure that we have the talent in the MUG to write these routines. I am certainly willing to help in any way that I can but I lack the expertise in assmbly language to do it myself.

Gene Riding
2227 Chicago St., San Diego CA 92110

.....

SECOND THAT MOTION

Buzz,
I agree with some of your comments of MDOS over CP/M. I find program loading much faster in MDOS and the system documentation is much better. Assembler level programming is much easier and faster in MDOS.

Micropolis BASIC has some limitations which I personally do not like, such as; fixed length allocated to records in a direct access file (250 bytes), and slow screen display.

My main dislike about MDOS is the lack of some form of "PIP". Transfering multiple files from one disk to another is very tedious. Also, the MDOS editor, although reasonable for program development, is not screen based (I use WORDMASTER).

S. Tattersall, Standard Telecommunication Labs
London Rd, Harlow, Essex, England CM17 9NA

.....

CLASSIFIED

MICROPOLIS Mod I Subsystem
2 drives, software, cables, controller. New, used only in prototype system for photos. Drives require power supply. List price $1140. First $600 takes all. Micropolis manual $40 extra. CP/M 1.4 and CBASIC for Mod I new sealed, $50 for both.

MICROPOLIS Controllers Model 1071 new $290 each.

VISA or M/C ok.

iPEX INTERNATIONAL INC.
16140 Valerio St., Van Nuys, CA 91406 213/781-0020

.....

SYSTEMATION COMPILER (cont.)

However, an end user must have the RUN/S module to execute the load module. RUN/S, by itself, costs $40. If all our vendor members buy BASIC/S and then offer new, compiled versions of their accounting, data base management, etc., programs - all we would have to buy is RUN/S and we could use them all.

To answer a few questions I have received - BASIC/S is a true compiler, not a pseudo-code interpreter like CBASIC. The RUN/S module provides all the system interfaces and library routines (such as the trig functions). There is no LINKing required. Both LOGe and LOG10 are implemented. Numeric ranges and precisions are the same as Micropolis BASIC -

```
Integers: -49,000,000,000,000,000,000 to
          +50,000,000,000,000,000,000

Real    : 1E-61 to 1E61
```

with a 20 digit precision available. (Do you realize that IBM-360s only have a 6 to 8 digit single precision, 12 to 16 digit double precision?)

There are a few new tidbits in BASIC/S. BCD arithmetic will do away with those rounding errors in your accounting programs (you can't represent 0.10 in binary arithmetic). There is an IF .. THEN .. ELSE structure which will allow us to do away with some GOTOs and make our programming more structured. There is a TAB(x[,y]) structure which enables absolute cursor addressing. PRINT TAB(10,10);"MUG" will print MUG, starting at column 10, line 10, of your screen (or printer).

Bad points? Oh, there are a few. You have to have the THEN in IF statements. If you now write

```
IF A=B C=D
```

it must be changed to

```
IF A=B THEN C=D.
```

You must have a space after key words. You can't write the above statement as

```
IFA=BTHENC=D.
```

This allows the use of key words in variable names. You wouldn't be able to title a variable RECORDSIZE, because RECORD and SIZE are key words.

Yup - all in all, seems pretty nice. Next month, I'll discuss the results of some specific translation from Micropolis BASIC to BASIC/S.

.....

FMT FUNCTION - (Continued from page 1)

For social security numbers:
FMT(512489135,"999-99-9999") returns "512-48-9135"

This use of FMT allows storage of certain commonly used numbers in a packed form and easy conversion to the familiar format at will. Notice that the numeric value is always right-justified in the format "picture", and remember that variables can be substituted for the constants in the above examples for both the numerics and string values.

Another handy use of FMT is the substitution of a variable drive number in a string containing a file name. Suppose D% is a variable containg a valid drive number, and F$ is a valid file name on that drive. With this information, you can write:

```
OPEN 1 FMT(D%,"9:")+F$
```

The FMT function in this case returns a single digit followed by a colon, to which the file name is concantenated by the plus sign, and the whole statement evaluates to a valid OPEN operation. In some cases you can even write:

```
OPEN 1 FMT(D%,"9:FILENAME")
```

FILENAME is the actual name of a file and D% is is drive number, so the FMT function returns a drive number and the FILENAME together. This use of FMT is somewhat dangerous, however, since you must make sure that the file name does not contain any of the meaningful FMT characters such as "Z" or "9". If it did, the numeric would be substituted within the file name itself. While this might be disasterous if you were trying to substitute a drive number, there might be instances where the substitution of a numeric in the body of a string would be useful.

The FMT function truncates numeric amounts at the last position indicated by the string "picture". If rounded values are desired, this characteristic can be used to advantage. By simply adding half a unit of the least significant position, FMT will return a string representing a rounded value. For example, to print the value of A to two decimal places with all smaller decimals dropped, you would write:

```
PRINT FMT(A,"ZZ9V.99")
```

But to round to the nearest hundreth use:

```
PRINT FMT(A+.005,"ZZ9V.99")
```

Either way, the thousandths position is dropped, but in the second example there will be a carry into the hundreths if the thousandths are equal to or greater than .005 because we are adding half a hundreth.

FMT is one of the most powerful features of our language and using it to greatest utility depends on an intimate knowledge of how it works. If you aren't too familiar with its properties, sit down at your computer and experiment with it. Learning to use a programming language is like learning any other language. Not everything you need to know is in the book and you have to speak it and try it out to become fluent. One nice thing about computers is they don't get insulted and are infinitely patient. The worst that can happen is a SYNTAX error or a wrong answer. Unlike TV and movie computers, there is no way you can cause smoke to pour out through the use of software.

.....

Published Monthly by the MUG
Subscription rates:
U.S., Canada, Mexico; $18/year: Other, $25/year

FIRST CLASS MAIL
===== ===== ====

FIRST CLASS MAIL
===== ===== ====

==========

MICROPOLIS USERS GROUP

Buzz Rudow, Editor
604 Springwood Circle
Huntsville AL 35803
(205) 883-2621

FIRST CLASS MAIL
===== ===== ====