LIFELINES®

THE SOFTWARE MAGAZINE™

$2.50 January 1982 Volume II, No. 8 (ISSN 0279-2575, USPS 597-830)

A Substitute For The SUBMIT Utility

A Dynamic BIOS Extension Technique

NOTICE

With our November issue, we began using a new mailing class for subscription copies. Some complaints about poor delivery of the December issue were experienced in the New York area. The December issue was placed into the U.S. Mail on November 25th. If you had a problem with the timeliness of this issue please call our Subscription Department at (212) 722-1700, or write to: Lifelines Subscription Department, 1651 Third Ave., New York, N.Y. 10028. We expect to place this issue, dated January 1982, into the mail on December 22nd. We will print each month the date of mailing and would appreciate help in tracking the deliveries.

LIFELINES®

January 1982

Volume II, No. 8

Contents

Opinion

Features

The CP/M™ Users Group

Software Notes

Product Status Reports

Miscellaneous

Lifelines (ISSN 0279-2575, USPS 597-830) is published monthly at a subscription price of $18 for twelve issues, when destined for the U.S., Canada, or Mexico, $40 when destined for any other country. Second-class postage paid at New York, New York. POSTMASTER, please send changes of address to Lifelines Publishing Corporation, 1651 Third Ave., New York, N.Y. 10028.

Editorial Comments

The Virtues and Perils of Standards

In a previous issue I focused on the importance of standards to the microcomputer industry. The argument for the significance of standards was based partly on the observation that such mechanisms tend to force the common good to prevail and suppress those who would rather serve their own interests (though often in the name of the common good).

The response of the readership has been gratifying, and the current interest in discussions of this type appears strong, widespread and growing rapidly.

Watch for the coming announcements of formal committees to develop the fully sanctioned standard for sixteen bit operating systems proposed in a previous issue.

It is interesting to note that not all software standards develop in the same way. In the case of the language C, committees did not meet to determine a standard. Rather, the text of Kernighan and Ritchie served to define the C language — and admirably at that. It is encouraging to know that there are responsible individuals seeking to develop and maintain languages of this caliber as standards, and not at the expense of the rest of us. Had the authors cared to, they could have sallied forth into the land of extensions and infinite revisions, like those before them who trespassed upon eternity.

The ANSI committee on Pascal, X3J9, made rapid and impressive progress until they arrived at the subject of extensions; then suddenly the pace slackened considerably, as the attendees argued long, hard and often eloquently for their favorite extension.

Many of you are familiar with the text *Software Tools*, considered a classic. It was authored in part by Kernighan and reviewed by Ritchie during its preparation. Curiously enough, RATFOR was modeled after C. As elsewhere, in the world of software good ideas seem invariably to spawn other good ideas. As you recall, UNIX served as the environment for C's development and has been since largely rewritten in C.

Dennis Ritchie's name shows up again and again, always in relation to some outstanding contribution to the field of software. In the case of UNIX, Ritchie was determined to "eradicate explicit machine dependencies" and it was this determination that ultimately led to the development of C, resulting in the widespread use of UNIX in many multi-user environments. This came as a direct result of C's transportability and architecture.

There are in fact machine dependent versions of C which systems programmers employ to write hardware dependent code. In this case the advantage is that though these versions are at best only quasi-standard, the code is easier to write and maintain.

However, these "degeneracies" have not resulted in the destabilizing effects that other languages have suffered; for instance, Pascal is to some extent plagued by the effects of its extensions.

The Department of Defense is responsible for the "standardization" of ADA, if only by legislating that it will be utilized extensively by the DOD agencies.

IBM has produced a great litany of standards which achieve this stature through their reputation for quality. The latest example is their selection of "THE" operating system for the IBM Personal Computer: MSDOS™(AKA SB-86™), which is rising with meteoric speed and ascendancy in the universe of microcomputerdom.

And no wonder ... this operating system has already set new technical and commercial standards heretofore unknown in this industry. If you want to see how an operating system should be documented, take a hard look at the IBM Personal Computer DOS manual. While the eight bit world continues to struggle with often woefully inadequate documentation (much of which is marked proprietary, in some cases on every page), IBM has done the finest job possible in providing the reader with well-illustrated, definitive descriptions of every aspect of the software. This seems to be in keeping with their apparent policy of disclosing all details of the hardware and software, in order to encourage end users, implementers and others to take full advantage of the systems capabilities. One west coast manufacturer has announced some twenty add-on peripherals for the IBM system already. This would have been difficult, if not impossible, had IBM chosen to take a proprietary posture with respect to the software and hardware architecture.

A language which is rapidly gaining in popularity is FORTH, but it lacks the infra-structure provided by an accepted standard. Should an effort be made to develop such a standard it would do much to accelerate its evolution into a state of widespread use and popularity, i.e. its acceptance as a "standard" language.

Readers interested in FORTH-like languages should review the excellent treatise on "Threaded Languages" by R.G. Loeliger. Also perhaps one of the most interesting contributions to the CPMUG, and an excellent example of a threaded interpretive language, is STOIC (Stack Oriented Interactive Compiler), copyrighted by MIT and Harvard in 1977. If you are interested in peering into the innermost workings of a language STOIC may be for you. It is particularly useful in developing hardware diagnostics but serves equally well as a pedagogical tool for those of you fascinated by assembly language, compilers and interpreters.

It is interesting how often in the pursuit of intellectual stimulation one encounters interesting people. When I first discovered STOIC it was late one evening in Los Angeles. In the course of perusing the "DOC" files on disk 23A I discovered the name of Wink

(continued on page 37)

The Pipeline

Carl Warren

Here are more products to consider

Making your system do what you want, the way you want it to can be difficult. And if you're using a number of software packages to perform your work, making them all work in tandem can be difficult. The Micropro packages – WordStar, DataStar, SpellStar, and the newly introduced InfoStar, for example – provide you with a great deal of data handling capability, but to get the most use out of them you must be able to manage their operation in as sophisticated a manner as the packages operate.

A company called Epic Computer Corp, 9181 Chesapeake Dr, San Diego, CA 92123, (714) 569-0440, has provided the mechanism that allows you this high level of interactivity. The product, called Supervyz, is priced at $99 and serves as a front end preprocessor to CP/M.

What Supervyz does make both CP/M and MP/M user-friendly by taking the mystery out of the popular operating systems.

The package works by using a series of user-friendly menus that support commands by employing the functions found in the O/S but handling them as command lines transparent to the user.

Users are greeted with a series of self-prompting, easy to understand menus that you design to interface directly to the applications you install. In addition, the powerful control system allows you to give the user commands intrinsic to the O/S by specifying them as menu items.

For turnkey systems, Supervyz can be set up to come up when the system is turned on, thus making it unnecessary for users to know proper boot procedures.

The turnkey function is implemented by using the built-in setup menu to write the SuperCCP to the track and sector that holds CP/M or MP/M's CCP. If your system doesn't appear on the master list, and odds are that it won't be there, you'll have to find the exact track and sector it lies on and provide the necessary offset.

A quick way of finding it is to use a disk dump program and look for the copyright notice. The first byte after the copyright is where you want to place the SuperCCP. For Heath/Zenith 89 systems that are employing Magnolia's double-density controller, however, it isn't as simple.

The reason? Magnolia put the CCP in

(continued next page)

user 31 of the disk, making it inaccessible to the casual user. Consequently, you'll need to use DDT to read USER 31 and find the correct locations.

Besides supporting applications, Supervyz provides a super time of day facility that interacts to your systems real-time clock. This then is available to the user by a simple menu entry. Interestingly, if your system doesn't support a hardware clock, and you're using software to create the clock from interrupts, you can have Supervyz call this routine as a matter of course.

Setting up a Supervyz menu is simple. Employing a Computer Aided Instruction technique, the powerful program leads you through the various formats required to establish a menu and desired help messages using tiny PILOT.

Such features as specifying default drives for the system diskette are available, as is the ability to specify several data drives, or in the case of a Winchester based system under MP/M, specific user areas.

Because Supervyz uses a command line function for setting up commands to the O/S, multiple commands can be imbedded in one menu call. This feature makes it possible to perform batch type tasks by a single entry.

Moreover, the package permits interfacing to a variety of terminals andprinters and can be used to put the system in a typewriter type mode. Since the Console Command Processor of CP/M or MP/M has been modified, all disks are logged in on entering the Supervyz program. This feature avoids irritating BDOS errors. Furthermore, once booted, there is no need to have a system disk resident in the system, since Supervyz makes itself memory resident with full CP/M functionality.

The basic package delivered on an 8-in. IBM compatible single density diskette includes: Supervyz the interactive system manager, Super CCP and extended console command processor, Menus Def and interactive menu builder, Help and interactive program tutor, and Install(S), a configuration program.

Supervyz requires a Z80 or 8080 compatible microprocessor, at least 40K bytes of RAM, CP/M 2.x or MP/M 1.1 or 2.0 operating systems, and at least 20K bytes of available disk space. There is no limit on the number of menus that can be nested; each menu can contain up to 10 functions with a description length of 32 characters and a command line length up to 64 characters with 4 parameter requests.

We found that the Supervyz can be put on any type of CP/M supported system, but located a bug in an MP/M operation. What occurs is that Supervyz fails to interact correctly with MP/M's disk map and will try to select non-existent drives or logical devices on a hard disk system. A call to Epic brought surprise, and few hours later acknowledgement that the problem did exist and would be fixed shortly–which by the time you read this should already be done.

Incidentally, Supervyz does take up a lot of disk space (20K), so you might want to consider it for larger systems that use a hard disk. However, for an Apple with 5.25-in. drives you can pare down the program for just those functions you need. We found that you could get Supervyz, dBASE II and WordStar (minus SpellStar) on one system diskette along with the three dBASE II command files we use for creating information databases.

You've probably heard CP/M may not be the best, and that others could have done better, and rather than talk about it Software 2000 Inc, PO Box 945, Los Alamitos, CA 90720 (213) 429-2317, has done something about it.

Rather than try to create an operating system that requires new support from the software community, the folks at Software 2000 have developed a CP/M, MP/M compatible system called TURBODOS, by carefully surveying what was available in O/S weighting the good with the bad to come up with those features that best suit a high-performance system.

TURBODOS is developed around modules that perform specific tasks and is ideal for OEMs who are creating very specialized systems. However, you can order single user copies at $250 without spooler, add $50 for the spooler; and if you want to live in the multiuser/multitasking world plan on spending $500 for the non-networking version, and for $250 you can have networking capability.

The package comes on 8-in. IBM compatible diskettes with the necessary modules and utility programs associated with the model you buy. In addition, if you happen to be an OEM you get documentation both in camera ready form and and in ASCII format. And yes the TURBODOS diskettes can be read under CP/M and vice versa.

Although we don't have the space to go into depth on the unique O/S, here are just a few of the features.

Unlike CP/M, TURBODOS is written for the Z80 and uses all the features found in the powerful microprocessor. In addition, TURBODOS supports a multiuser network of interconnected microcomputers which can share a common pool of mass storage, printers (up to 16) and other peripherals. In the network environment, you have full file lockout, a sophisticated multiqueue print spooler, and full error detection and correction.

Moreover, the power O/S supports floppies, and hard disks without any special redefiniton to the BIOS. And because TURBODOS uses 512 byte or 1024 sector sizes, and eliminates the reserved system tracks a storage increase of about 25 to 35% can be achieved, as well as faster access to files in a read mode.

Implementing TURBODOS is easy, according to software manager John Larber at The Systems Group, Orange CA; he's putting the O/S on their System 2800 series of hard disk computers. Larber says that it takes less than a day and that the documentation eases you right into it.
Interestingly, it appears, although not mentioned in the TURBODOS documentation, that CP/M and MP/M can be treated as a task. This might bring up some interesting possibilities, especially in a network environment.

For that total system approach, you might want to talk to the folks at Computer Development Inc, 6700 SW 105th, Beaverton, OR 97005 (800) 547-1831. This company that you

(continued on page 21)

SUPERSUB, A Replacement For The CP/M SUBMIT Utility

Ron Fowler

One of the most useful programs provided with CP/M is the SUBMIT utility, which allows system commands to be read from a disk file for automatic processing. Its command-line substitution facility provides a convenient means of developing "prototype" command files, with actual parameter substitution performed when the command file is invoked.

There are, however, some shortcomings associated with SUBMIT that recur frequently. Frequently enough to inspire me to write my own batch processor, SUPERSUB, that completely replaces, and is fully compatible with, the standard SUBMIT utility.

PROBLEMS WITH SUBMIT

SUBMIT will produce a "garbaged" output file whenever it encounters an empty line in the input file. This is a real handicap when attempting to pass command lines to PIP.COM, for example, using XSUB (the CP/M2 utility that extends the SUBMIT capability to transient programs), since the only way to exit PIP is to enter an empty command line (you can enter a control-"C" as the first character of the line, but this is a "real-time" function, and cannot be done within a SUBMIT file). SUPERSUB generates the necessary zero-length line in the output file.

Another SUBMIT drawback is the constant need to enter an editor and create a command file, even for the simplest of jobs. This added step discourages the use of SUBMIT for very simple "command stacking" applications. With this in mind, I gave SUPERSUB the ability to interactively accept command lines from the console. An additional option allows the entire job to be specified on the SUPERSUB command line.

COMMAND FILE NESTING

I felt that a nesting capability would be a significant feature to add to SUPERSUB. For example, it would be useful to create a submit file called COMPILE.SUB, to facilitate the program compilation process, and "clean up" any leftover files. COMPILE.SUB might look like this:

```
BASCOM =$1        ;COMPILE PGRM SPECIFIED IN CMD LINE
L80 $1,$1/N/E     ;LINK THE PGRM (PROGRAM)
PIP B:=$1.COM     ;MOVE THE OBJECT FILE TO B DRIVE
ERA $1.REL        ;DELETE THE .REL FILE
ERA $1.COM        ;AND THE OBJECT FILE
```

Now let's say I want to compile several programs at once (a common situation in applications programming). Under SUBMIT, I would have to type in a SUBMIT invocation for each program:

```
A>SUBMIT COMPILE LEDGER     [COMPILE.SUB executes,
                            operator waits for it to
                            complete, then types:]
A>SUBMIT COMPILE PAYABLES   [more waiting for
                            COMPILE.SUB, then:]
A>SUBMIT COMPILE RECVBLES   [and the job is done.]
```

Using SUPERSUB, I can create a file (CMPILALL.SUB) containing the commands

```
SUBMIT COMPILE LEDGER
SUBMIT COMPILE PAYABLES
SUBMIT COMPILE RECVBLES
```

and execute the whole thing with the single command line

```
A>SSUB CMPILALL
```

(Note that I've abbreviated SUPERSUB to SSUB on all my disks; SUPERSUB as a name is a bit grandiose anyway). I can also save the trouble of creating CMPILALL.SUB (especially if I want to do this only once) and enter the single command line

```
A>SSUB /COMPILE LEDGER;SSUB COMPILE
      PAYABLES;SSUB COMPILE RECVBLES
```

(This should appear on one line.)

Or, if the individual command lines to be executed are a bit long, I can specify the Interactive mode by entering only a slash ("/") on the command line (after the SUPERSUB invocation). SUPERSUB will then prompt for each command line with an asterisk. To continue the previous example,

```
A>SSUB /
*COMPILE LEDGER
*SSUB COMPILE PAYABLES
*SSUB COMPILE RECVBLES
*                          [empty line here terminates]
```

ON-LINE HELP FUNCTION

I believe all programs of any complexity should be "self-instructive" by providing the user a means of typing out, on the terminal, the major modes of operation, and any necessary command line syntax. Further, the command line used to invoke this "help summary" should be the simplest possible form not otherwise used by the program. In the case of SUPERSUB, this is a command line with no arguments at all:

```
A>SSUB
```

SUBMIT.COM COMPATIBILITY

SUPERSUB is fully compatible with CP/M's SUBMIT util-

(continued next page)

ity, including command parameter substitution and control-character translation:

1) Parameters within the "prototype" file, of the form $1, $2, etc., are substituted from the command line on a one-for-one basis. Parameters are passed through to any nested files. Two successive dollar-sign characters ("$$") may be used to introduce a single "$" into the output file.

2) An up-arrow symbol ("^") may precede an alphabetic character to insert the associated control-character into the output file.

DETAILED INSTRUCTIONS

In addition to the normal SUBMIT mode of operation and associated command line usage, SUPERSUB provides two additional modes of input: Interactive and Summary.

SUMMARY MODE

Summary mode allows the entire SUBMIT job to be specified in the CP/M command line. This mode is enabled by using the slash ("/")character as the first character of the command line. The individual submit lines must be separated with a semicolon. For example,

```
A>SSUB / CRCK *.* F;CRCK B:*.* F;COMPARE CRCKFILE.CRC B:
       ↑ (this space is optional)
```

will create a file of CRC's of all files on A:, then create a similar file on B:, then compare the two. (CRCK, by Keith Petersen, and COMPARE, by Ward Christensen, are available from The CP/M Users Group).

INTERACTIVE MODE

You may enter the interactive entry mode by typing "SUPERSUB /<CR>"(i.e, "SUPERSUB /" with no arguments). SUPERSUB will prompt for input with an asterisk, and you may then enter SUBMIT lines from the keyboard. Multiple commands may be entered on a line by separating them with semicolons. An empty line terminates the input. Example:

```
A>SUPERSUB /
*CRCK *.* F
*CRCK B:*.* F
*COMPARE CRCKFILE.CRC B:
*                                <empty line here>
A>CRCK *.* F                     <submit file begins execution>
```

has the same effect as the above SUMMARY mode example.

HELP FUNCTION

Typing SUPERSUB with no arguments will print the built-in help file.

NOTES

1) Nested SUBMIT runs are only usable up to a maximum of 128 nested commands at any one time. This is a limitation of the CP/M Console Command Processor.

2) If you change the drive specification for the output file, you may want to do the same thing with XSUB (Digital Research's function nine extender). Within XSUB, find the submit File Control Block (search for "$$$ SUB" within XSUB.COM) and change the first FCB byte (i.e, the byte before the first "$") to:

0 - to use default drive
1 - to use drive A:
2 - to use drive B:
etc.

3) In SUMMARY and INTERACTIVE modes, passed parameters have no meaning. When these modes are used, the parameter flag, "$", will be passed through literally to the output file.

4) Zero-length output lines may be created in SUMMARY and INTERACTIVE modes by using two consecutive semicolons. This is, in effect, a blank logical line.

5) Interactive mode may be aborted by typing control-C as the first character of a line. Also, all normal CP/M editing characters are available in this mode.

HOW IT WORKS

Those who have been following Ward Christensen's series of tutorials on CP/M know that the output file ($$$.SUB) consists of lines from the input file written in reverse order, one line per CP/M record. To fully understand how this file is built, refer to the assembly listing while reading the following program description.

SUPERSUB can be divided into three distinct sections: initialization, input file read, and output file write ($$$.SUB). Each will be discussed, along with any called subroutines.

PROGRAM INITIALIZATION

Initialization begins with a check for a HELP request (an empty command line), which suppresses the sign-on message, and branches directly to the HELP print routine.

If there is no help request, then the sign-on message is printed and the routine INITVAR is called. INITVAR zeros the working variable area and initializes to empty the table which will later hold the address of each command-line parameter.

Next, the routine GETPAR is called. GETPAR parses the input command line, setting up the variables OPTION and CLFLAG (which will be used later to determine the input mode), and calls the routine PUTPAR as each command-line parameter is encountered. PUTPAR stores each parameter (prefixed by its length) into free memory, advances the free memory pointer, and stores a pointer to the parameter

into TABLE. This pointer will later be used to locate the parameter when user parameter substitution is done.

Finally, the input file control block is prepared for use by the routine SETUP. SETUP insures that the input name specified is of type "SUB". If the type field is left blank, SETUP moves the correct type into place.

READING THE INPUT FILE

The RDFILE routine has the responsibility of reading in the user's text file. RDFILE reads each line sequentially into memory, prefixed by a link word that contains the absolute memory address of the previous line (the first line has a link value of zero) followed by a length byte, then the line itself (stripped of carriage-return/line feed characters). No parameter substitution is done during the file read.

RDFILE increases the value of LINNUM with each line read. This variable is used by the error handlers to provide line number reporting when an error is encountered.

RDFILE gets its input character-by-character by calling GNB, the character "GET" routine. This is the level at which the command-line option flags OPTION and CLFLAG must be used to determine the source of characters. If the OPTION variable is an ASCII slash ("/"), then the character is retrieved from the GNBKBD routine. Otherwise control passes to GNB1, which handles disk file character input. GNB1 uses the IBP byte to form a pointer into TBUF, the disk buffer. If IBP is pointing past the buffer end, the FILL routine is called to refill TBUF. IBP is then incremented, and the character is returned in the accumulator.

The GNBKBD routine is used when the input source is NOT a disk file. At this point, there are two possible sources of input: the original command line (Summary mode) and the console (Interactive mode). The flag CLFLAG is used to decide which source to use. If CLFLAG is non-zero, then the Summary mode is active, and characters are read from the original command line (passed by CP/M in TBUF). A CLFLAG of zero indicates Interactive mode, and characters are read from the keyboard buffer CLBUF (which is filled using CP/M function #10 whenever it is found empty by GNBKBD).

The code at label GNBCL is common to both modes, and substitutes an end-of-line character for a semicolon, to allow multiple logical lines on a physical line.

The end of file condition is returned by GNB as a 1AH in the accumulator. This value is normally returned only from a disk file, so special logic in GNBKBD must detect the end condition. This logic returns EOF at the end of the input command line (in Summary mode) or upon entry of an empty line (in Interactive mode).

WRITING THE OUTPUT FILE

You may have wondered why the input file was read in with link pointers and length bytes added. That's because CP/M's CCP (and XSUB utility) read the temporary submit file BACKWARD, one line packed into each disk record. This allows CP/M to use the record count and next record fields of the directory entry as file pointers. This also means that SUPERSUB must write out the last line first, followed by the next-to-last line, etc., with the first line of the input file being at the END of the output file. The linked list provides a simple means of traversing the lines in reverse order.

The subroutine WRSUB writes the output file, and is the last routine called by the mainline code. WRSUB first scans backward from the end of the input file, looking for the last line in the file that is not empty. If one is found (an error message is issued if one is not), control passes to the WRLOP loop, which calls PUTLIN to move the line to the output buffer, decrements the line number for possible error reporting (remember, we're working BACKWARD now), and scans back to the previous line to repeat the process. If there is no previous line (indicated by a link pointer with zero value), then the file is closed, and a warm-boot to CP/M is executed, causing the batch job to begin.

The PUTLIN routine moves characters one-by-one from the input line stored in memory to the output buffer at TBUF. This is the point where parameter substitution and control-character translation take place (unless Summary or Interactive modes are active, which have no reason to do substitutions). Buffer pointers and counters are set up for the GETCHR and PUTCHR routines, after which characters are received one-at-a-time by calling GETCHR.

Each character is checked to see if it is either of the option characters, "$" or "^". In the case of the "$" character, LKAHEAD is called to determine if the "$" is present twice successively, which is interpreted as a single "$" to move from input file to output file. If there is no second "$" character, the next character is considered to be the parameter number (as in "$1", "$2", etc.). The code at label SUBS collects the parameter number from the input stream (which may be more than one digit, if the conditional NPAR is greater than ten) and uses it to index into the parameter address table. The parameter, prefixed by its length, is then moved from the parameter storage area into the output buffer.

After each line is moved into the output buffer, the subroutine FLUSH writes the buffer to the disk.

When WRSUB has written the last record to the output file, control returns to the mainline routine, which warm-boots CP/M and begins execution of a new file.

(continued next page)

OOPS

Our apologies to Miken Optical Company for a bad typo in our December issue. On page 44, under New Products, their **IBIOS** was described, and an important **not** was omitted in the first sentence of the write-up; the sentence should read:

"IBIOS is an interactive BIOS for CP/M-80, designed **not** to lock the user into a running program without allowing interruptions."

```
;
;
;*************************************************
;*                EXTENDED SUBMIT FOR            *
;*                     CP/M                      *
;*************************************************
;
; REVISED 09/13/81 (RGF): added control character translation
;                         fixed bug in line number reporting
;
;       VERSION 1.1                by Ron Fowler
;       2/18/81 (first written)    WESTLAND, MICH.
;
;
; This program is intended as a replacement for the
; SUBMIT program provided with CP/M.  It provides sev-
; eral new facilities:
;       1) Nestable SUBMIT runs
;       2) Interactive entry of SUBMIT job (no need
;          to use an editor for simple SUBMIT runs)
;       3) Command line entry of small SUBMIT jobs
;       4) Ability to enter blank lines in an edited
;          SUBMIT file
;       5) User customization of number of parameters
;          and drive to send $$$.SUB to
;
;
;
; DEFINE BOOLEANS
;
FALSE   EQU     0
TRUE    EQU     NOT FALSE
;
;*************************************************************
;
;               --  User customizable options --
;
NPAR    EQU     10      ;NUMBER OF ALLOWABLE PARAMETERS
SUBDRV  EQU     0       ;MAKE '0 FOR DFLT, 1,2,3,ETC FOR A,B,C
QUIET   EQU     FALSE   ;SET TO TRUE TO ELIMATE SIGN-ON MSG
CPBASE  EQU     0       ;SET TO 4200H FOR HEATH CP/M
;
;
;*************************************************************
;
; CP/M DEFINITIONS
;
FPCHAR  EQU     2       ;PRINT CHAR FUNCTION
PRINTF  EQU     9       ;PRINT STRING FUNCTION
RDBUF   EQU     10      ;READ CONSOLE BUFFER
OPENF   EQU     15      ;OPEN FILE FUNCTION
CLOSEF  EQU     16      ;CLOSE FILE FUNCTION
DELETF  EQU     19      ;DELETE FILE FUNCTION
READF   EQU     20      ;READ RECORD FUNCTION
WRITEF  EQU     21      ;WRITE RECORD FUNCTION
MAKEF   EQU     22      ;MAKE (CREATE) FILE FUNCTION
;
BDOS    EQU     CPBASE+5
;
FCB     EQU     5CH     ;DEFAULT FILE CONTROL BLOCK
FCBRC   EQU     15      ;FCB OFFSET TO RECORD COUNT
FCBNR   EQU     32      ;FCB OFFSET TO NEXT RECORD
FN      EQU     1       ;FCB OFFSET TO FILE NAME
FT      EQU     9       ;FCB OFFSET TO FILE TYPE
TBUF    EQU     CPBASE+80H  ;DEFAULT BUFFER
TPA     EQU     CPBASE+100H ;TRANSIENT PROGRAM AREA
;
PUTCNT  EQU     TBUF    ;COUNTER FOR OUTPUT CHARS
;
; DEFINE SOME TEXT CHARACTERS
;
CR      EQU     13      ;CARRIAGE RETURN
LF      EQU     10      ;LINE FEED
TAB     EQU     9
;
; START OF PROGRAM CODE
;
        ORG     TPA
;
;       GET THE BALL ROLLING
;
SUBMIT: LXI     H,0     ;SAVE STACK IN CASE
        DAD     SP      ;  ONLY HELP REQUESTED
        SHLD    SPSAVE  ;(NOT OTHERWISE USED)
        LXI     SP,STACK
        CALL    START
;
; SIGN ON MESSAGE
;
        IF      NOT QUIET
        DB      'SuperSUB V1.1'
        ENDIF
;
        DB      CR,LF,'$' ;NEWLINE EVEN IF QUIET
;
START:  POP     D       ;RETRIEVE STRING POINTER
        MVI     C,PRINTF
        LDA     FCB+1   ;ANYTHING ON CMD LINE?
        CPI     ' '
        JZ      HELP    ;NO, GO PRINT HELP
        CALL    BDOS    ;PRINT THE SIGN-ON
        CALL    INITVAR ;INITIALIZE THE VARIABLE AREA
        CALL    GETPAR  ;GET COMMAND LINE PARAMETERS
        CALL    SETUP   ;SET UP READ OF SUBMIT FILE
        CALL    RDFILE  ;READ THE SUBMIT FILE
        CALL    WRSET   ;SET UP WRITE OF "$$$.SUB"
        CALL    WRSUB   ;WRITE "$$$.SUB"
        JMP     CPBASE  ;GO START THE SUBMIT
;
;
; SETUP SETS UP THE FILE CONTROL BLOCK
; FOR READING IN THE .SUB TEXT FILE
;
SETUP:  LXI     H,FCB+FT ;LOOK AT FIRST CHAR OF
        MOV     A,M     ;   FILE TYPE.  IF IT IS
        CPI     ' '     ;   BLANK, THEN GO MOVE
        JZ      PUTSUB  ;   "SUB" INTO FT FIELD
        LXI     D,SUBTYP ;IT'S NOT BLANK SO MAKE
        MVI     B,3     ;   SURE IT'S ALREADY
        CALL    COMPAR  ;   "SUB"
        JNZ     NOTFND  ;IF NOT, IT'S AN ERROR
        RET
;
; MOVE "SUB" INTO THE FILE TYPE
;
PUTSUB: XCHG            ;BY CONVENTION, MOVE FROM
        LXI     H,SUBTYP ; @HL TO @DE
        MVI     B,3
        CALL    MOVE
        RET
;
; MOVE # BYTES IN B REGISTER FROM @HL TO @DE
;
MOVE:   MOV     A,M     ;PICK UP
        STAX    D       ;PUT DOWN
        INX     H       ;I'M SURE
        INX     D       ; YOU'VE SEEN THIS
        DCR     B       ; BEFORE...
        JNZ     MOVE    ;100 TIMES AT LEAST
        RET             ;I KNOW I HAVE!
;
; GETPAR MOVES THE SUBSTITUTION PARAMETERS SPECIFIED
; IN THE COMMAND LINE INTO MEMORY, AND STORES THEIR
; ADDRESSES IN THE PARAMETER TABLE.  THIS ALLOWS
; SUBSTITUTION OF $1, $2, ETC., IN THE SUBMIT COMMANDS
; WITH THEIR ACTUAL VALUES SPECIFED IN THE COMMAND
; LINE.
;
GETPAR: LXI     H,TBUF+1 ;WHERE WE FIND THE COMMAND TAIL
        CALL    SCANTO  ;SKIP SUBMIT FILE NAME
        STA     OPTION  ;FIRST CHAR OF CMD LINE IS OPTION
        RC              ;LINE ENDED?
        CPI     '/'     ;NO, CHECK OPTION
        JNZ     GLP0    ;NOT KEYBOARD INP, READ FILE
        INX     H       ;POINT PAST '/'
SLSCAN: SHLD    CLPTR   ;SAVE CMD LINE PTR
        MOV     A,M     ;KBD IS SOURCE, GET EOL FLAG
        STA     CLFLAG  ;SAVE AS EOL FLAG
        CPI     ' '     ;ALLOW SPACES AFTER '/'
        RNZ             ;GOT NON-BLANK, DONE
        INX     H       ;ELSE CONTINUE SCAN
        JMP     SLSCAN
GLP0:   MOV     A,M     ;INPUT IS FROM A .SUB FILE..THIS
        INX     H       ;  CODE SKIPS OVER THE NAME OF
        ORA     A       ;  THE SUB FILE TO GET TO THE
        RZ              ;  COMMAND LINE PARAMETERS
        CPI     ' '
        JZ      GLP
        CPI     TAB
        JNZ     GLP0
GLP:    CALL    SCANTO  ;PASS UP THE BLANKS
        RC              ;CY RETURNED IF END OF CMD LINE
        CALL    PUTPAR  ;NOW PUT THE PARAMETER INTO MEM
        RC              ;CY RETURNED IF END OF CMD LINE
        JMP     GLP     ;GET THEM ALL
;
; SCANTO SCANS PAST BLANKS TO THE FIRST NON-BLANK. IF
; END OF COMMAND LINE FOUND, RETURNS CARRY SET.
;
SCANTO: MOV     A,M
        INX     H
        ORA     A       ;SET FLAGS ON ZERO
        STC             ;IN CASE ZERO FOUND (END OF CMD LIN)
        RZ
        CPI     ' '
        JZ      SCANTO  ;SCAN PAST BLANKS
        CPI     TAB     ;DO TABS TOO, JUST FOR
        JZ      SCANTO  ;  GOOD MEASURE
        DCX     H       ;FOUND CHAR, POINT BACK TO IT
        ORA     A       ;INSURE CARRY CLEAR
        RET
;
; PUTPAR PUTS THE PARAMETER POINTED TO BY HL INTO
; MEMORY POINTED TO BY "TXTPTR".  ALSO STORES THE
; ADDRESS OF THE PARAMETER INTO THE PARAMETER TABLE
; FOR EASY ACCESS LATER, WHEN WE WRITE $$$.SUB
;
PUTPAR: PUSH    H       ;SAVE POINTER TO PARM
        LHLD    TXTPTR  ;NEXT FREE MEMORY
        XCHG            ;  INTO DE
        LHLD    TBLPTR  ;NEXT FREE AREA OF TABLE
        MOV     A,M     ;NON-ZERO IN TABLE
        ORA     A       ; INDICATES TABLE
        JNZ     PAROVF  ; TABLE OVERFLOW (TOO MANY PARMS)
        MOV     M,E     ;STORE THE PARM ADRS
        INX     H
        MOV     M,D
        INX     H
        SHLD    TBLPTR  ;SAVE TABLE PNTR FOR NEXT TIME
        POP     H       ;GET BACK PARM POINTER
        PUSH    D       ;SAVE FREE MEM POINTER BECAUSE
                        ;  WE WILL HAVE TO HAVE IT BACK
                        ;  LATER TO STORE THE LENGTH
        INX     D       ;POINT PAST LENGTH STORAGE
        MVI     B,0     ;INITIALIZE LENGTH OF PARM
PPLP:   MOV     A,M     ;GET NEXT BYTE OF PARM
        INX     H
        ORA     A       ;TEST FOR END OF CMD LINE
        JZ      PP2     ;JUMP IF END
        CPI     ' '     ;TEST FOR END OF COMMAND
        JZ      PP2
        CPI     TAB     ;TAB ALSO ENDS COMMAND
        JZ      PP2
        STAX    D       ;PUT PARAMETER BYTE-BY-BYTE
        INX     D       ;INTO FREE MEMORY
        INR     B       ;BUMP LENGTH
        JMP     PPLP
PP2:    XCHG
        SHLD    TXTPTR  ;NEW FREE MEMORY POINTER
        POP     H       ;REMEMBER OUR LENGTH POINTER?
        MOV     M,B     ;STORE THE LENGTH
        XCHG            ;HAVE TO RETN HL > CMD LINE
        ORA     A       ;NOW RETURN END OF LINE FLAG
```

```
        STC
        RZ              ;RETURN CY IF ZERO (EOL MARK)
        CMC
        RET
;
;       RDFILE READS THE .SUB FILE SPECIFIED
;       IN THE SUBMIT COMMAND INTO MEMORY
;
RDFILE: LXI     H,0     ;INIT LINE NUMBER
        SHLD    LINNUM
        LDA     OPTION  ;USING A FILE?
        CPI     '/'     ;'/' OPTION TELLS
        JZ      LINE    ;JUMP IF NOT
        LXI     D,FCB   ;WE ARE, OPEN IT
        MVI     C,OPENF
        CALL    BDOS
        INR     A       ;IF OFFH RETURNED,
        JZ      NOTFND  ;  THEN FILE NOT FOUND
LINE:   LHLD    LINNUM  ;BUMP LINE NUMBER
        INX     H
        SHLD    LINNUM
        LHLD    PREV    ;GET PREV PREVIOUS LINE POINTER
        XCHG
        LHLD    TXTPTR  ;GET CURRENT FREE MEM POINTER
        SHLD    PREV    ;MAKE IT THE PREV LINE (FOR NXT PASS)
        MOV     M,E     ;STORE AT BEGIN OF CURRENT LINE,
        INX     H       ;  A POINTER TO THE PREVIOUS
        MOV     M,D
        INX     H
        PUSH    H       ;LATER WE WILL PUT LENGTH HERE
        INX     H       ;SKIP PAST LENGTH
        MVI     C,0     ;INITIALIZE LENGTH TO ZERO
LLP:    CALL    GNB     ;GET NEXT BYTE FROM INPUT SOURCE
        JC      EOF     ;CY SET IF END OF FILE FOUND
        CALL    UCASE   ;CONVERT TO UPPER CASE
        CPI     1AH     ;SEE IF CPM END OF FILE INDICATOR
        JZ      EOF
        CPI     LF      ;IGNORE LINEFEEDS
        JZ      LLP
        CPI     CR      ;IF IT'S A CARRIAGE RETURN,
        JZ      EOL     ;  THEN DO END OF LINE
        MOV     M,A     ;STORE ALL OTHERS INTO MEMORY
        INX     H
        CALL    SIZE    ;MAKE SURE NO MEMORY OVERFLOW
        INR     C       ;BUMP CHAR COUNT
        JM      LENERR  ;MAX OF 128 CHARS PER LINE
        JMP     LLP     ;GO DO NEXT CHAR
;
;       DO END OF LINE SEQUENCE
;
EOL:    SHLD    TXTPTR  ;SAVE FREE MEMORY POINTER
        POP     H       ;CURRENT LINE'S LENGTH POINTER
        MOV     M,C     ;STORE LENGTH AWAY
        JMP     LINE    ;GO DO NEXT LINE
;
;       END OF TEXT FILE
;
EOF:    SHLD    TXTPTR  ;SAVE FREE MEMORY POINTER
        POP     H       ;CURRENT LINE'S LENGTH POINTER
        MOV     M,C     ;STORE LENGTH AWAY
        RET             ;ALL DONE READING SUB FILE
;
;       GET NEXT BYTE FROM INPUT FILE
;
GNB:    PUSH    H       ;DON'T ALTER ANYBODY
        PUSH    D
        PUSH    B
        LDA     OPTION  ;INPUT FROM .SUB FILE?
        CPI     '/'     ;TOLD BY ORIG CMD LINE OPTION
        JNZ     NSLASH  ;JUMP IF WE ARE
        CALL    GNBKBD  ;NO, GET A BYTE FROM KBD INPUT
        JMP     GNBXIT  ;THEN LEAVE
NSLASH: LDA     IBP     ;GET BUFFER POINTER
        ORA     A       ;PAST END?
        CM      FILL    ;WRAPPED AROUND
        JNC     GNB1    ;NO END OF FILE
        MVI     A,1AH   ;FAKE EOF
GNB1:   MOV     E,A     ;PUT IN DE
        MVI     D,0
        INR     A       ;POINT TO NEXT
        STA     IBP     ;PUT AWAY
        LXI     H,TBUF  ;NOW OFFSET INTO BUFR
        DAD     D
        MOV     A,M     ;GET CHAR THERE
GNBXIT: POP     B       ;RESTORE EVERYBODY
        POP     D
        POP     H
        ORA     A       ;TURN ON CARRY
        RET
;
;       FILL INPUT BUFFER
;
FILL:   MVI     C,READF
        LXI     D,FCB
        CALL    BDOS
        ORA     A       ;GOT GOOD READ?
        MVI     A,0     ;(NEW BUF PTR)
        STC
        RNZ             ;RETN CY=EOF
        CMC             ;NO EOF, NO CY
        RET
;
; COME HERE TO GET A .SUB CHARACTER WHEN
; WE'RE NOT USING A .SUB FILE ("/" OPTION)
;
GNBKBD: LDA     CLFLAG  ;USE CP/M CMD LINE?
        ORA     A
        JNZ     GNBCL   ;THEN GO DO IT
        LDA     CLCNT   ;NOT, CHECK LOCAL
        ORA     A       ;  CMD LINE CHAR COUNT
        CM      CLFILL  ;REFILL WHEN IT WRAPS BACK
        JC      GKEND   ;GOT CARRY (FROM CLFILL), RETURN EOF
        DCR     A       ;COUNT DOWN
        STA     CLCNT
        JP      GNBCL   ;IF PLUS, BUFFER NOT EMPTY
        MVI     A,CR    ;OUT OF CHARS, RETURN A CR
        RET

GKEND:  MVI     A,1AH   ;RETURN EOF
        RET
;
; GET NEXT BYTE OF INPUT FROM CMD LINE @CLPTR
;
GNBCL:  LHLD    CLPTR   ;LOAD THE POINTER
        MOV     A,M     ;GET THE CHAR
        INX     H       ;BUMP POINTER FOR NEXT TIME
        SHLD    CLPTR
        CPI     ';'     ;LOGICAL END-OF-LINE?
        JNZ     NSEMI   ;JUMP IF NOT
        MVI     A,CR    ;YES, TRANSLATE IT
        RET
NSEMI:  ORA     A       ;PHYSICAL END-OF-LINE
        RNZ             ;THIS ONLY NEEDED WHEN INPUT
                        ;  SOURCE IS ORIG CPM CMD LINE
        MVI     A,1AH   ;TRANSLATE THAT TO END OF FILE
        RET
;
; SUBROUTINE TO RE-FILL THE LOCAL COMMAND LINE
;
CLFILL: LXI     D,PROMPT ;PRINT A PROMPT
        MVI     C,PRINTF ;USE CP/M FUNCT 9
        CALL    BDOS
        LXI     D,CLBUF  ;NOW FILL THE BUFFER
        MVI     C,RDBUF
        CALL    BDOS
        LDA     CLCNT    ;RETURN WITH COUNT IN A
        LXI     H,CLTEXT ;RESET THE CMD LINE POINTER
        SHLD    CLPTR
        ORA     A        ;SET CY ON LEN NZ
        STC
        RZ
        CMC
        RET
;
;       MAKE SURE NO MEMORY OVERFLOW
;
SIZE:   LDA     BDOS+2  ;HIGHEST PAGE POINTER
        DCR     A       ;MAKE IT BE UNDER BDOS
        CMP     H       ;CHECK IT AGAINST CURRENT PAGE
        RNC             ;NC=ALL OKAY
        JMP     MEMERR  ;OTHERWISE ABORT
;
;       SET UP THE $$$.SUB FILE
;       FOR WRITING
;
WRSET:  LXI     D,SUBFCB
        MVI     C,OPENF
        CALL    BDOS    ;OPEN THE FILE
        INR     A       ;CHECK CPM RETURN
        JZ      NONE1   ;NONE EXISTS ALREADY
;
;       $$$.SUB EXISTS, SO SET
;       FCB TO APPEND TO IT
;
        LDA     SUBFCB+FCBRC ;GET RECORD COUNT
        STA     SUBFCB+FCBNR ;MAKE NEXT RECORD
        RET
;
;       COME HERE WHEN NO $$$.SUB EXISTS
;
NONE1:  LXI     D,SUBFCB
        MVI     C,MAKEF
        CALL    BDOS
        INR     A
        JZ      NOMAKE  ;0FFH=CAN'T CREATE FILE
        RET
;
;       WRITE THE "$$$.SUB" FILE
;
WRSUB:  LHLD    PREV    ;THIS CODE SCANS BACKWARD
        MOV     A,H     ;  THRU THE FILE STORED IN
        ORA     L       ;  MEMORY TO THE FIRST NON-
        JZ      NOTEXT  ;  NULL LINE.  IF NONE IS
        MOV     E,M     ;  FOUND, ABORTS
        INX     H
        MOV     D,M     ;HERE, WE PICK UP PNTR TO PREV LINE
        INX     H       ;NOW WE POINT TO LENGTH
        XCHG            ;WE NEED TO STORE AWAY
        SHLD    PREV    ;  POINTER TO PREV LINE
        XCHG
        MOV     A,M     ;NOW PICK UP THE LENGTH
        ORA     A       ;SET Z FLAG ON LENGTH
        JNZ     WRNTRY  ;GOT LINE W/LENGTH: GO DO IT
        LHLD    LINNUM  ;NOTHING HERE, FIX LINE NUMBER
        DCX     H       ;(WORKING BACKWARD NOW)
        SHLD    LINNUM
        JMP     WRSUB
WRLOP:  LHLD    PREV    ;GET PREV LINE POINTER
        MOV     A,H
        ORA     L        ;IF THERE IS NO PREV LINE
        JZ      CLOSE    ;  THEN WE ARE DONE
        MOV     E,M      ;ELSE SET UP PREV FOR NEXT
        INX     H        ;  PASS THRU HERE
        MOV     D,M
        INX     H
        XCHG             ;NOW STORE IT AWAY
        SHLD    PREV
        XCHG
WRNTRY: CALL    PUTLIN   ;WRITE THE LINE TO THE FILE
        LHLD    LINNUM   ;BUMP THE LINE NUMBER
        DCX     H        ;DOWN (WORKING BACK NOW)
        SHLD    LINNUM
        JMP     WRLOP
;
;       $$$.SUB IS WRITTEN, CLOSE THE FILE
;
CLOSE:  LXI     D,SUBFCB
        MVI     C,CLOSEF
        JMP     BDOS
;
;       THIS SUBROUTINE WRITES A LINE
;       TO THE $$$.SUB FILE BUFFER,
;       AND FLUSHES THE BUFFER AFTER
;       THE LINE IS WRITTEN.
;
```

(continued next page)

```
PUTLIN: MOV     A,M     ;PICK UP LENGTH BYTE
        INX     H       ;POINT PAST IT
        STA     GETCNT  ;MAKE A COUNT FOR "GET"
        SHLD    GETPTR  ;MAKE A POINTER FOR "GET"
        LXI     H,TBUF+1;TEXT GOES AFTER LENGTH
        SHLD    PUTPTR  ;MAKE POINTER FOR "PUT"
        XRA     A       ;INITIALIZE PUT COUNT
        STA     PUTCNT
        MOV     B,L     ;COUNT FOR CLEAR LOOP
CLR:    MOV     M,A     ;ZERO OUT BUFFER LOC
        INX     H
        INR     B       ;COUNT
        JNZ     CLR
;
;       THIS LOOP COLLECTS CHARACTERS
;       FROM THE LINE STORED IN MEMORY
;       AND WRITES THEM TO THE FILE.
;       IF THE "$" PARAMETER SPECIFIER
;       IS ENCOUNTERED, PARAMETER SUB-
;       STITUTION IS DONE
;
PUTLP:  CALL    GETCHR  ;PICK UP A CHARACTER
        JC      FLUSH   ;CY = NO MORE CHAR IN LINE
        CPI     '^'     ;CONTROL-CHAR TRANSLATE PREFIX?
        JNZ     NOTCX
        CALL    GETCHR  ;YES, GET THE NEXT
        JC      CCERR   ;ERROR: EARLY END OF INPUT
        SUI     '@'     ;MAKE IT A CONTROL-CHAR
        JC      CCERR   ;ERROR: TOO SMALL
        CPI     ' '
        JNC     CCERR   ;ERROR: TOO LARGE
NOTCX:  CPI     '$'     ;PARAMETER SPECIFIER?
        JNZ     STOBYT  ;IF NOT, JUST WRITE CHAR
        LDA     OPTION  ;CHECK OPTION: '$' DOESN'T
        CPI     '/'     ;  COUNT IN '/' MODE
        MVI     A,'$'   ;(RESTORE THE '$')
        JZ      STOBYT
        CALL    LKAHED  ;PEEK AT NEXT CHAR
        JC      PARERR  ;LINE ENDING MEANS PARAM ERR
        CPI     '$'     ;ANOTHER "$"?
        JNZ     SUBS    ;IF NOT THEN GO DO SUBSTITUTION
        CALL    GETCHR  ;GET THE 2ND "$" (WE ONLY LOOKED
                        ;  AHEAD BEFORE)
STOBYT: CALL    PUTCHR  ;WRITE CHAR TO FILE
        JMP     PUTLP
;
;       PARAMETER SUBSTITUTION...LOOKS UP THE
;       PARAMETER # AFTER THE "$" AND PLUGS IT
;       IN IF IT EXISTS.
;
SUBS:   CALL    NUMTST  ;IT BETTER BE A NUMBER
        JC      PARERR  ;  OTHERWISE PARAM ERROR
        MVI     B,0     ;INITIALIZE PARM #
        JMP     LPNTRY  ;WE JOIN LOOP IN PROGRESS...
SUBLP:  CALL    LKAHED  ;LOOK AT NEXT CHAR
        JC      DOSUBS  ;IF LINE EMPTY, THEN PLUG IN PARM
        CALL    NUMTST  ;CHECK FOR NUMERIC
        JC      DOSUBS  ;DONE IF NOT
LPNTRY: CALL    GETCHR  ;NOW REMOVE THE CHAR FROM INPUT STREAM
        SUI     '0'     ;REMOVE ASCII BIAS
        MOV     C,A     ;SAVE IT
        MOV     A,B     ;OUR ACCUMULATED COUNT
        ADD     A       ;MULTIPLY  BY TEN
        ADD     A
        ADD     B
        ADD     A
        ADD     C       ;THEN ADD IN NEW DIGIT
        MOV     B,A     ;RESTORE COUNT
        JMP     SUBLP
;
;       PERFORM THE SUBSTITUTION
;
DOSUBS: MOV     A,B     ;GET PARM #
        DCR     A       ;MAKE ZERO RELATIVE
        JM      PARERR  ;OOPS
        CALL    LOOKUP  ;LOOK IT UP IN PARM TABLE
        JC      PARERR  ;IT'S NOT THERE
        MOV     B,A     ;LENGTH IN B
SUBLP1: INR     B       ;TEST B FOR ZERO
        DCR     B
        JZ      PUTLP   ;DONE
        MOV     A,M     ;GET CHAR OF REAL PARAMETER
        INX     H       ;POINT PAST FOR NEXT TIME
        PUSH    H       ;SAVE REAL PARM POINTER
        CALL    PUTCHR  ;PUT IT IN THE FILE
        POP     H       ;GET BACK REAL PARM POINTER
        DCR     B       ;COUNTDOWN
        JMP     SUBLP1
;
;       COME HERE WHEN A LINE IS FINISHED,
;       AND WE NEED TO WRITE THE BUFFER TO DISK
;
FLUSH:  LXI     D,SUBFCB
        MVI     C,WRITEF
        CALL    BDOS
        ORA     A
        JNZ     WRERR   ;CPM RETURNED A WRITE ERROR
        RET
;
;       GETCHR GETS ONE CHAR FROM
;       LINE STORED IN MEMORY
;
GETCHR: LXI     H,GETCNT
        MOV     A,M     ;PICK UP COUNT
        DCR     A       ;REMOVE THIS CHAR
        STC             ;PRESET ERROR
        RM              ;RETURN CY IF OUT OF CHARS
        MOV     M,A     ;UPDATE COUNT
        LHLD    GETPTR  ;CURRENT CHAR POINTER
        MOV     A,M     ;PICK UP CHAR
        INX     H       ;BUMP POINTER
        SHLD    GETPTR  ;PUT IT BACK
        CMC             ;TURN CARRY OFF
        RET
;
;       PUTCHR PUTS ONE CHAR TO
;       THE OUTPUT BUFFER
;
PUTCHR: LXI     H,PUTCNT
        INR     M       ;INCREMENT COUNT
        JM      LENERR  ;LINE WENT TO > 128 CHARS
        LHLD    PUTPTR  ;GET BUFFER POINTER
        MOV     M,A     ;PUT CHAR THERE
        INX     H       ;BUMP POINTER
        SHLD    PUTPTR  ;PUT IT BACK
        RET             ;ALL DONE
;
;       LOOK AHEAD ONE CHAR IN
;       THE INPUT STREAM.  SET
;       CARRY IF NONE LEFT.
;
LKAHED: LDA     GETCNT
        ORA     A       ;SEE IF COUNT IS DOWN TO ZERO
        STC             ;PRE SET INDICATOR
        RZ
        MOV     A,M     ;PICK UP CHAR
        CMC             ;TURN OFF CARRY FLAG
        RET
;
;       LOOK UP PARAMETER WITH NUMBER IN
;       A REG. RETURN A=LENGTH OF PARM,
;       AND HL => PARAMETER
;
LOOKUP: CPI     NPAR
        JNC     PAROVF  ;PARM # TOO HIGH
        MOV     L,A
        MVI     H,0     ;NOW HAVE 16 BIT NUMBER
        DAD     H       ;DOUBLE FOR WORD OFFSET
        LXI     D,TABLE
        DAD     D       ;DO THE OFFSET
        MOV     E,M     ;GET ADDRESS OF PARM
        INX     H
        MOV     D,M
        MOV     A,D     ;ANYTHING THERE?
        ORA     E
        JNZ     LKUPOK
        XRA     A       ;NO, ZERO LENGTH
        RET
LKUPOK: XCHG            ;NOW IN DE
        MOV     A,M     ;PICK UP LENGTH
        INX     H       ;POINT PAST LENGTH
        RET
;
;       UTILITY COMPARE SUBROUTINE
;
COMPAR: LDAX    D
        CMP     M
        RNZ
        INX     H
        INX     D
        DCR     B
        JNZ     COMPAR
        RET
;
;       NUMERIC TEST UTILITY SUBROUTINE
;
NUMTST: CPI     '0'
        RC
        CPI     '9'+1
        CMC
        RET
;
;DECIMAL OUTPUT ROUTINE
;
DECOUT: PUSH    B
        PUSH    D
        PUSH    H
        LXI     B,-10
        LXI     D,-1
;
DECOU2: DAD     B
        INX     D
        JC      DECOU2
        LXI     B,10
        DAD     B
        XCHG
        MOV     A,H
        ORA     L
        CNZ     DECOUT
        MOV     A,E
        ADI     '0'
        CALL    TYPE
        POP     H
        POP     D
        POP     B
        RET
;
; PRINT CR, LF ON CONSOLE
;
CRLF:   MVI     A,CR
        CALL    TYPE
        MVI     A,LF
;
; PRINT CHAR IN A ON CONSOLE
;
TYPE:   PUSH    H       ;SAVE REGS
        PUSH    D
        PUSH    B
        MOV     E,A     ;PUT IN E FOR CP/M
        MVI     C,FPCHAR
        CALL    BDOS    ;PRINT IT
        POP     B       ;RESTORE ALL
        POP     D
        POP     H
        RET
;
; CONVERT CHAR IN A TO UPPER CASE
;
UCASE:  CPI     'a'     ;VALIDATE CASE
        RC
        CPI     'z'+1
        RNC
        ANI     5FH     ;GOT LC, CONV TO UC
```

```
        RET
;
;       ERROR HANDLERS
;
WRERR:  CALL    ERRXIT
        DB      'Disk full$'
NOMAKE: CALL    ERRXIT
        DB      'Directory full$'
MEMERR: CALL    ERRXIT
        DB      'Memory full$'
NOTFND: CALL    ERRXIT
        DB      'Submit file not found$'
PARERR: CALL    ERRXIT
        DB      'Parameter$'
PAROVF: CALL    ERRXIT
        DB      'Too many parameters:$'
LENERR: CALL    ERRXIT
        DB      'Line too long:$'
NOTEXT: CALL    ERRXIT
        DB      'Submit file empty$'
CCERR:  CALL    ERRXIT
        DB      'Control character$'
ERRXIT: POP     D
        MVI     C,PRINTF
        CALL    BDOS
        LXI     D,ERRMSG        ;PRINT 2ND HALF MSG
        MVI     C,PRINTF
        CALL    BDOS
        LHLD    LINNUM          ;TELL LINE NUMBER
        CALL    DECOUT
        CALL    CRLF
        LXI     D,SUBFCB        ;DELETE THE $$$.SUB FILE
        MVI     C,DELETF
        CALL    BDOS
        JMP     CPBASE
;
ERRMSG: DB      ' error on line number: $'
;
; PROMPT FOR COMMAND LINE INPUT
;
PROMPT: DB      CR,LF,'*$'
;
;       INITIALIZE ALL VARIABLES
;
INITVAR:
        LXI     H,VAR
        LXI     B,ENDVAR-VAR
INITLP: MVI     M,0     ;ZERO ENTIRE VAR AREA
        INX     H
        DCX     B
        MOV     A,B
        ORA     C
        JNZ     INITLP
        LXI     H,TABLE ;INIT PARM TABLE POINTER
        SHLD    TBLPTR
        LXI     H,0FFFFH ;MARK END OF TABLE
        SHLD    ENDTBL
        LXI     H,FREMEM ;FREE MEMORY STARTS TXT AREA
        SHLD    TXTPTR
        MVI     A,80H   ;FORCE READ
        STA     IBP
        STA     CLCNT   ;FORCE CONSOLE READ
        RET
;
; PRINT HELP WITH PROGRAM OPTIONS
;
HELP:   LXI     D,HLPMSG ;PRINT THE HELP STUFF
        MVI     C,PRINTF
        CALL    BDOS
        LHLD    SPSAVE  ;THEN RETURN W/NO WARM-BOOT
        SPHL
        RET
;
HLPMSG: DB      CR,LF,'How to use SUPERSUB:',CR,LF
        DB      CR,LF,'SUPERSUB<CR>             :print this HELP message'
        DB      CR,LF,'SUPERSUB /<CR>           :go into interactive mode'
        DB      CR,LF,'SUPERSUB /<cmd lines>    :use SUMMARY mode'
        DB      CR,LF,'SUPERSUB <FILE> <PARMS> :as in standard SUBMIT.COM'
        DB      CR,LF
        DB      CR,LF,'In "/" (interactive) mode, SUPERSUB will prompt you'
        DB      CR,LF,'a line at a time for the SUBMIT job input...logical'
        DB      CR,LF,'lines may be combined on the same input line by sep-'
        DB      CR,LF,'erating them with semicolons.  Example:'
        DB      CR,LF,'  A>SUPERSUB /STAT;DIR'
        DB      CR,LF,'specifies two commands on the same input line.',CR,LF
        DB      CR,LF,'Submitted jobs may be nested...SUPERSUB does not erase'
        DB      CR,LF,'any existing submit job (appends to them instead).'
        DB      CR,LF
        DB      CR,LF,'To insert a control character into the output, pre-'
        DB      CR,LF,'fix it with a "^" (works in any mode).'
        DB      CR,LF,'$'
;
;       VARIABLE STORAGE
;
VAR     EQU     $
;
TXTPTR: DW      0       ;FREE MEMORY POINTER
TBLPTR: DW      0       ;POINTER TO PARM TABLE
LINNUM: DW      0       ;CURRENT LINE NUMBER
PREV:   DW      0       ;POINTER TO PREV LINE
GETCNT: DB      0       ;COUNTER FOR 'GET'
GETPTR: DW      0       ;POINTER FOR 'GET'
PUTPTR: DW      0       ;POINTER FOR 'PUT'
IBP:    DB      0       ;INPUT BUFFER POINTER
CLPTR:  DW      0       ;COMMAND LINE POINTER
CLFLAG: DB      0       ;USE CP/M CMD LINE FLAG
OPTION: DB      0       ;'/' OPTION FLAG STORE
TABLE:  DS      NPAR*3  ;PARAMETER TABLE
ENDTBL: DW      0FFFFH  ;END OF PARAMETER TABLE
;
ENDVAR  EQU     $
;
; COMMAND LINE BUFFER...NOT INITIALIZED
;
CLBUF:  DB      128     ;BUFFER LENGTH
CLCNT:  DB      0       ;CHARACTER COUNTER
CLTEXT: DS      128     ;THE BUFFER ITSELF
;
SPSAVE: DW      0       ;STACK POINTER SAVE
;
;       FCB FOR $$$.SUB
;
SUBFCB: DB      SUBDRV  ;DRIVE SPECIFIER
        DB      '$$$    '
SUBTYP: DB      'SUB'
        DW      0,0,0,0 ;INITIALIZE REST OF FCB
        DW      0,0,0,0
        DW      0,0
;
;       STACK AREA
;
        DS      200     ;WHY NOT?
STACK   EQU     $
FREMEM  EQU     $
;
        END     SUBMIT
```

"ONEDRIVE" On CP/M Version 2.25B For The TRS-80 Model II

CONFIG.COM does not properly set the ONEDRIVE bit in this version of CP/M. In order to use PIP.COM and COPY.COM with one drive, this bit must be set in the sysgen image. The following procedure will accomplish this.

1) Obtain a sysgen image in the following way:
   A. Run SYSGEN
   B. Name drive A as the source drive
   C. When asked "Destination Drive or return to skip," hit the ENTER key.
   D. Save 45 CPMSYS

2) Change the byte
   A. DDT CPMSYS
   B. examine locations 2100h as per below
      s2100 10 ____ ← old value was 10h.
      Insert a number here, as defined below.
   C. The value to be inserted to replace the 10 at address 2100 is the sum of:
      80 if you are on a one-drive system;
      2 if you want to implement an auto-startup command on WARM boot;
      1 if you want to implement an auto-startup command on COLD boot;
      (for example, 83 if one-drive and autobooting warm and cold; set the autoboot command in CONFIG.COM later).
   D. After entering the appropriate number to the right of the 10, DDT will step on to address 2101, which we do NOT want to change; please type a period and ENTER by it.
   E. Back at the hyphen prompt, type G0 (NOT go! that's a zero after the g) to exit from DDT.

3) Generate a new system
   A. Run SYSGEN
   B. When asked for source drive, hit the <CR> or ENTER key
   C. When asked for destination drive, type "A"

PIP and COPY will now work properly with the ONEDRIVE program.

A Dynamic CP/M BIOS Extension Technique

Michael J. Karas

A common CP/M system implementation problem occurs when the hardware/software integrator wishes to produce a system that has physical mass storage devices which are not media compatible with default standard media formats, or when the mass storage device media is non-removable. The question is how to provide an efficient software exchange mechanism that allows the computer to recognize some type of standard media format without requiring modification of the basic system architecture.

This article describes a technique to dynamically add logical "disk drives" to the CP/M BIOS of a CP/M-80 System. It makes it possible for the user to access software drivers that permit his system to interpret a standard disk media format. The hardware support mechanism to allow functioning of the software drivers will vary widely with the system type.

For CP/M version 2.2, the default standard software exchange medium is a single density, single sided, eight inch floppy diskette formatted to IBM 3740 format specifications. The file structure on the medium conforms to the scheme defined by Digital Research, Inc. in conjunction with the release of CP/M Version 1.4 and subsequently Version 2.2. Most software applications vendors and software distributors make their products available on this media format. Currently popular types of CP/M system configurations will often use 5 1/4 inch mini-floppy disk drives for compactness and cost minimization. Obviously a system of this type can have difficulty in handling the "standard 8 inch" media format as a means of moving software in and out of the 5 1/4 inch disk environment.

A typical example of this problem arose when I implemented the Digital Research CP/M 2.2 operating system on a desk top computer system. The computer system cabinet accommodated a mini floppy diskette drive and a 5 1/4 inch 5 megabyte Winchester hard disk drive. As the standard product is sold, use of a single density diskette was not generally possible. However, in the engineering development of the system software, an 8" drive was needed to provide software exchange capability. The hardware of the computer system consists of a Monolithic Systems, Inc MSC 8009A Z-80 CPU board with an integrated floppy disk controller. The controller is normally utilized to interface the 5 1/4 mini-floppy to the computer. Its design permits an 8" floppy to be connected to the same controller through a special cabling setup.

Once I had the hardware connection problems ironed out, the software developed to run the drive resulted in the two programs PRLMOVE and SINGLE; these will be described in the remainder of this article. The scheme was to develop a BIOS software package that could relocate itself out of the way in the CP/M system memory, add extra software drivers to run the eight inch disk, and still allow the system user to run any normal CP/M based transient.

THE RELOCATOR MODULE

The program source code, Listing 1, included in this article entitled "PRLMOVE.ASM", is the source for a CP/M transient relocator that allows a transient program to be relocated to the top of the user's CP/M TPA (transient program area). The basis of the mover is a page relocation bit map that contains specific information showing which bytes within an 8080 object program require modification so that the program may be loaded for execution at an address other than the one where it was originally assembled. This module is the program used to relocate the separately produced "BIOS EXTENSION MODULE" to the area of memory under the operating system. The following paragraphs present an explanation of how the PRLMOVE software works and the procedure for getting it made into the necessary ".HEX" file.

The operation of the PRLMOVE program is very similar to the relocation module on the front of two utility packages sold with CP/M 2.2 by Digital Research. Both DDT and XSUB are moved to an executable position in memory other than the standard TPA base address. Relocation is handled by a bit map included with the object code image. The layout of a program with PRLMOVE on the "front" is as follows:

Base Execution Address
of PRLMOVE 0100H

LXI B,MODSIZE
(set BC registers to length in bytes of relocatable module. Placed here by PRL format of the developed BIOS add-on module.)

Base Load Address
of BIOS ADD-ON MODULE...0200H

(linked .PRL image of relocatable module.)

Base Load Address of RELOCATION
BIT MAP .....0200H+MODSIZE

(linker placed image of bit map for relocatable program module.)

The execution of PRLMOVE firstly sets up an execution stack pointer; it places as the first item on the stack the value of the old stack pointer as passed to PRLMOVE by the CP/M CCP. Recall that the CCP (Console Command Processor) passes control to a loaded transient module with a stack pointer indicating a position within the CCP of a return address, to allow restart of CCP without warm boot. Note also that this restart is possible if execution of the transient does not use the memory area occupied by the CCP. The BIOS Extension technique requires this CCP restart capability, so the entry address pointer is saved.

The PRLMOVE program then calculates the base execution address of the BIOS add-on module. This address is determined by three factors:

(1) The current execution base address, the CP/M BDOS determined by the address value at addresses 6 & 7 of the base memory page. This address also, by default, "tells" us where the top of CCP is located.
(2) The size of the CCP module as it sits in memory below the BDOS.
(3) The size add-on BIOS module that we intend to move into a memory position below the CCP.

Once the base execution address is determined, the add-on module code is block moved to that position. The number of bytes moved depends upon the constant "MODSIZE" that was loaded into the (BC) registers as the first instruction of PRLMOVE. The block move completes with the post incremented memory pointer aimed right at the first byte of the relocation bit map.

The construction of the bit map contains a bit for each byte of the relocatable module. If the relocation is done in a manner such that the base of the BIOS add-on package is always on the base address of a page of memory, then the 8080 type object program may be relocated simply by adding a relocation offset to the upper byte of address parameters within the software. The bits of the MAP show where these upper byte address parameters are located. The most significant bit of the first MAP byte corresponds to the first byte of the just moved add-on module.

The bit map is scanned one bit at a time while a pointer is moved correspondingly through the "moved" add-on module. Each time a "1" bit is found the corresponding code byte has the appropriate offset added. (Note that PRLMOVE adds an offset one less than the value you may be led to expect in that the linker used to produce the PRL (Page ReLocatable) image sets the absolute execution page base of the BIOS add-on module to 0100H, or one page different than the zero based "CODE SEGMENT ORIGIN" of the add-on module source).

Once all bits of the add-on module have been scanned, again determined by the initial constant loaded into (BC), the PRLMOVE program sets the stack pointer back exactly as we received it from the CCP. Execution control is then transferred to the base of the relocated add-on module.

The file PRLMOVE.ASM is entered into a disk file in source code form using a typical editor or the like. I happen to like WordMaster (MicroPro International). Once entered the .HEX file may be produced with the standard CP/M assembler as follows:

A>ASM PRLMOVE<cr>
..assuming source on A: and you desire to produce a .PRN file.

The resulting short .HEX file appears as:

```
:0101000001FD
:1001030021000039310002E5C52107007ED6083DF4
:1001130090571E00D521000278B1CA28010B7E1228
:10012300132 3C31B01D1C1E5622578B1CA4B010B6F
:100133007BE607C23E01E37E23E36F7D176FD24761
:10014300011A841213C32D01D154141E00E1F9EBDB
:01015300E9C2
:0000000000
```

Note that the first byte is an "LXI B" opcode in a one byte hex record. This is to allow later direct overlay of this small hex file upon the .PRL file produced for the add-on module. DDT (Diagnostic Debugging Tool) is used to perform this function. The .PRL file format contains a MODULE SIZE word as the second and third bytes of the file; it elegantly becomes the immediate constant for the LXI B instruction in the overlaid PRLMOVE program. Observe also that the source code for PRLMOVE contains an ORIGIN statement to place the execution address at 0100H, the base of the TPA. (Some non-standard CP/M implementations may have other TPA base addresses, so this value would require modification for use in those systems).

THE ADD-ON MODULE PHILOSOPHY

The program given in Listing 2 is a partial BIOS add-on module designed to add single density drive capability to a CP/M implementation to a hardware system utilizing the floppy disk controller contained upon a Monolithic Systems Inc. MSC 8009 board. The specific application shown here tends to really be immaterial to most readers. The intent is to show a working application that I've made.

The overall sequence of operation may be deduced from the program listing but I will discuss some of the more salient features. In the discussion some people are bound to comment that the program philosophy disobeys certain programming rules. My answer is that this is a technique to add capability to a CP/M Version 2.2 system in a dynamic fashion. The scheme is not at all entirely new and has been demonstrated to work in various versions of:

(1) Micropro's WordStar 3.0 with its capability to run another transient program without exit from WordStar to the operating system.

(2) FAST, the CP/M track buffering utilities designed and placed into The CP/M Users Group by Ward Christensen.

(3) and, the Digital Research submit file extension utility called XSUB.

Each of these programs was designed to implement a philosophy based on making a job running under CP/M easier. I am sure that each implementation required that certain "GAMES" be played with the operating system in order to get it to work. The SINGLE program discussed here is really just another implementation of the same old "GAME". For those programmers that hate to see programming of this style, then I invite you to write your own operating system that has all the features that you desire. I chose to use the system that I already know and find works well within the development environment of my work.

The operation of the add-on utility is somewhat complex to understand to the casual user of CP/M, so the discussion below will attempt to be comprehensive in explaining the "PHILOSOPHY" but will contain a minimum of information on the "exact" code implementation of the SINGLE program. The essential parts of the program listing are included as a part of the article for those who want to really pull it apart.

The add-on utility is intended to supply the user of a CP/M system with software drivers to allow reference to logical disk drives without having to modify the existing BIOS environment. To this end the add-on module is designed to attach to the operating system so that the user can return to the command level of CP/M and ex-

(continued next page)

ecute whatever program he or she desires. The attached devices can then be referenced as real CP/M disk drives for file transfers, editing, and so forth. To provide this capability the normal entry points to the existing CP/M BIOS must be modified so that the new software may enter the picture.

The utility described here, once relocated, performs a block move of the current BIOS jump vector table to a location within the utility. The present table is then substituted with a new table of jump entry points to substitute BIOS I/O processing routines that make up a bulk of the add-on utility. The BIOS entry points from the old preexisting table are kept intact so that the add-on module can pass control to the necessary I/O drivers of the host system: the console, list device, reader, punch, and disk I/O; with already defined local disk drives just pass control to the existing standard BIOS.

The addition of a logical single density disk drive afforded by the SINGLE program hinges around the add-on SELECT DISK routine. Each time CP/M accesses the select disk entry point with a drive select code, the add-on utility checks the select code for reference to drive "O:" which has been defined to be the add-on single density drive. If the select disk add-on sees a select for O: a select flag is set true. If the current select request is not for O:, then the select flag is set false and select control is passed through the old moved vector table to the existing BIOS select routine. The select flag is then used to steer all subsequent disk I/O requests to the appropriate disk drivers. If the select flag is set, then the add-on module drivers are utilized for I/O. If it's not set then I/O requests are passed to the normal BIOS. Some special care in the design of the drivers is needed. For example, in the case of the SET DMA ADDRESS routine, the address is accepted by the add-on utility and passed to the normal BIOS. This is done because a transient program under CP/M may not necessarily reset the data buffer address each time the logical unit select is switched. Another area requiring special care may be the HOME routine, if the add-on module used sector deblocking. The SINGLE program does not use deblocking so the HOME routine will appear relatively plain if you look at the listing.

A casual observer may question the need to block move all of the existing BIOS Vector Table jumps when only the disk I/O routines require "steering". The primary reason for the add-on utility block moving the whole table is that use of the relocated table requires the defeat of the normal WARM BOOT procedure. Recall that the add-on software is relocated just below the CCP. The relocated software modifies two jump address locations in the base page of memory when it begins execution. The first jump modified is location 6&7 which normally points to memory at the base address of the BDOS. This address parameter is used by many transient utilities to "SIZE" the transient memory area in the calculation of available work space. If the BDOS base pointer was not modified to a new value, then a transient could overlay the SINGLE program, rendering an inoperable system the next time a system call is made. The locations 1&2 are also modified so that the new "WARM BOOT" address becomes an entry point to the SINGLE program. This allows the normal warm boot to be overridden. Normal warm boot in a CP/M system generally results in a reload of the CCP and BDOS into memory and resetting of the page zero jump addresses. If this were to happen while the add-on SINGLE program was resident, then the system might once again "BLOW UP". In the above discussion of the PRLMOVE program it was mentioned that the start of execution of the relocated add-on module had the stack pointer set to a value pointing to an entry address within the already resident CCP. The override entry point in the SINGLE program for warm boot utilizes the initial value of the CCP entry stack to re-execute the same resident CCP image each time a supposed WARM BOOT is performed. [One thing to remember is that since a new copy of CCP and BDOS are not reloaded each time the override warm boot function is performed, the normal re-log of all disk drives with respect to directory check summing is not done. This does not permit disk changes to be done while SINGLE is resident and active if you intend to write to the changed disk. You'll get R/O errors if you try!]

The reason that the whole BIOS vector table is moved is related to the fact that many, many CP/M based transient programs run by "direct BIOS access" for performing I/O. The location of the normal BIOS vector table is known to a transient program by the address parameter stored in location 1&2 in the base memory page. But remember, SINGLE modifies that address to allow normal warm boot override. SINGLE cleverly solves the problem of the transient program doing I/O routine indexing off the warm boot vector; when valid jump type entries of BIOS type I/O functions are used, it makes the transient program warm boot pointer appear to be "in the right place" .

Another feature of SINGLE is its modification of the BDOS address at locations 5&6 in memory, so that they still have the same offset within the execution page as when SINGLE is not resident. This allows certain programs that use the page address offset value from location 0006H for special functions to still run properly. (DR's XSUB appears to be a program of this type.)

Once SINGLE is installed by running the command file, the user of the system may inadvertently attempt to load the SINGLE program again. This would result in two copies of the same utility being located in memory at the same time. This is not needed so the second loaded copy of SINGLE does a SINGLE already present check. If it is found to already be present, the second loaded copy prints a message "Single Already Active" and returns to the CCP without activating the second copy. The scheme utilized by the second loaded copy to determine if one SINGLE program is already present is a bit obtuse but appears to work fine. A fake file name, not possible to be found on a CP/M disk, of "n,,,,,,,,,," is placed into a file control block by the second copy of SINGLE. This second loaded copy of SINGLE then makes a system call through location 5 in an attempt to OPEN a file with a name corresponding to the FCB. The first loaded copy of SINGLE filters all BDOS Calls through location 5 and checks for file OPEN functions with a file name of "n,,,,,,,,,,". If a BDOS call of this type is found, the first active copy of SINGLE places a zero in the accumulator and returns to the calling second loaded module. If the second loaded module sees the returned zero in the accumulator, it is an indication that the first copy of SINGLE is already present and active.

Since all executable, relocated copies of SINGLE are the same, the first loaded copy must also go through the SINGLE present check. This is done, but since SINGLE was not previously present to filter the file open function, then the OPEN call goes right to BDOS. Since, under normal circumstances, the file "n,,,,,,,,,," will not be found for the OPEN call, then the real BDOS will return 0FFH in the accumulator to indicate not found. This fact is used by the first loaded SINGLE to indicate "I'M HERE FIRST". The number "n" as the first character is a module type indicator assigned to the SINGLE add-on utility. During the BDOS Call filter scan, the value of "n" is checked for the appropriate module type value. If not found, the the first loaded copy of SINGLE will abort the scan and pass the file OPEN call onto the normal BDOS entry point. This scheme permits the design of multiple add-on BIOS extension modules, each with different module type numbers. Under the address scan and present checking that is done, a common scheme is utilized for all types of modules, but copies of various different relocated modules may all be resident within the CP/M system memory all at the same time. The number present is only limited by available memory space. Remember that each relocated module consumes some 8K bytes of memory space.

PUTTING IT ALL TOGETHER

The SINGLE program, or another like it designed for your application, is designed to be utilized under single user CP/M 2.2. The procedure to get a version up and running will be outlined below.

The source file, being very similar to a BIOS for CP/M 2.2, requires the add-on disk definitions to have disk parameter tables, check vectors and allocation vectors. In addition a directory buffer must be allocated. The method utilized by SINGLE to generate the appropriate tables is through the macro library DISKDEF.LIB provided by Digital Research on the CP/M 2.2 distribution diskette. This macro capability needs a macro assembler to properly process the DISKDEF macros. An additional implementation requirement is the need to have the assembler output available as a .REL file. The Digital Research Relocating Macro Assembler RMAC is just the ticket to generate the appropriate .REL file and process the DISKDEF.LIB macro includes. The command structure below shows how to assemble the SINGLE program included in Listing 2. The source is assumed to be on drive B: and the .REL desired to be placed on A:. The print file is sent to the printer while generation of the symbol table is inhibited.

A>RMAC SINGLE $AB RA PP SZ<cr>

The resulting .REL file has to be converted to a page relocatable format. After getting aggravated over the problem of how to easily make a "BIT MAP" I found out that the Digital Research LINK program distributed with RMAC and PL/I-80 contains the capability to generate page relocatable files (.PRL type) that are compatible with MP/M. After reviewing the LINK output format to a .PRL file I then designed the previously described PRLMOVE program to relocate the LINK output files. The command structure below shows how to produce the file SINGLE.PRL from the RMAC output on Drive A: entitled SINGLE.REL.

A>LINK SINGLE[OP]<cr>

...where [OP] directs LINK to generate the desired .PRL file. The program PRLMOVE and the page relocatable SINGLE.PRL must be combined into a single executable command file as shown in Figure 1 below.

This results in a command file that adds the extra drive capability to a CP/M System. If the reader has a desire to experiment with the BIOS add-on scheme shown here, I will be happy to share the complete source code and a text listing file of this article with anyone that wants it. To obtain a copy just send an eight inch single density formatted diskette to me in a diskette mailer with sufficient enclosed return postage and address label at this address: Michael J. Karas, 2468 Hansen Court, Simi Valley, CA 93065. If you can reach me via phone at (805) 527-7922 some evening between 7:30 PM and 10:30 PM California time, I can also arrange to send you the article text and programs via modem at 300 baud with the standard Ward Christensen MODEM program. Plan for about one hour of modem time at 300 baud.

Figure 1

```
A>DDT<cr>
DDT Vers. 2.2
-ISINGLE.PRL<cr>
-R<cr>                          <==Load of .PRL file to
                                file to address 0100H
                                with code image starting
                                at address 0200H

NEXT  PC
nnmm 0000                       <==Convert nn to decimal
                                and remember value]

-IPRLMOVE.HEX<cr>
-R<cr>                          <==Read PRLMOVE program in
                                over the .PRL file
                                at load address of
                                0100H.
-G0<cr>                         Exit DDT to CP/M

A>SAVE dd SINGLE.COM            <==Save dd pages of memory
                                to get command file.
                                dd = converted nn from
                                above!
```

(continued next page)

Listing 1

FILE: PRLMOVE.ASM

```
;*****************************************************************
;       PRL FILE FORMAT RELOCATER FOR BIT MAPPED FILES
;*****************************************************************
;
;
;       THIS SMALL PROGRAM BLOCK MOVES A BIT MAP RELOCATABLE
;       FILE FROM ADDRESS 0200H ABOVE THIS MODULE UP TO A SPOT
;       BELOW THE CCP AND THEN JUMPS TO THE BASE OF THE MOVED
;       CODE. THE DIGITAL RESEARCH RELOCATING ASSEMBLER AND THE
;       COMPANION LINKER PERMIT THE GENERATION OF THE PRL FILE
;       FORMAT. LINK WILL PUT THE CODE SIZE WORD IN TO ADDRESS
;       101 AND 102. THE CODE SPOT IS INTENDED TO BE 0200H WITH
;       THE BIT MAP IMMEDIATELY ABOVE ABOVE THE CODE. A ONE IN THE
;       BIT MAP INDICATES THE LOCATION OF A MOVED BYTE THAT
;       REQUIRES A RELOCATION ADDRESS  OFFSET.
;
;*****************************************************************
;
;
;START POINT FOR THE BEGINNING OF THE MOVER MODULE
;
        ORG     0100H
;
;
;GENERAL CP/M BDOS INTERFACE EQUATES.
;
BDOS    EQU     0005H           ;FILE MANAGER ENTRANCE LOCATION
CODE$START EQU  0200H           ;LINK 80 CODE START POINT FOR ORG 0 FILE
;
;
;START POINT OF MOVER CODE
;
        DB      01H             ;INSERT HEX FILE CODE FOR LXI B,XXXX
        DS      2               ;INSERT CODE SIZE FROM LINK .PRL FILE
                                ;WITH DDT HERE
        LXI     H,0             ;GET CCP STACK FOR LATER PASSING
        DAD     SP              ;TO RELOCATED PROGRAM
        LXI     SP,CODE$START   ;LET STACK WORK DOWN FROM 0200H
        PUSH    H               ;SAVE CCP POINTER ON OUR STACK
;
        PUSH    B               ;SAVE A COPY OF CODE SIZE ON STACK
;
;
;GET BDOS PAGE ADDRESS BOUNDARY
;
        LXI     H,BDOS+2        ;PAGE ADDRESS OF BDOS BASE
        MOV     A,M             ;INTO (A)
        SUI     8               ;DECREASE SIZE FOR CCP SIZE
                                ;OF EIGHT PAGES
        DCR     A               ;ONE MORE TIME TO ACCOUNT FOR
                                ;..PARTIAL PAGE CODE SIZE
        SUB     B               ;SUBTRACT CODE SIZE IN TRUNC INTEGER SIZE
;
        MOV     D,A             ;(DE) = DESTINATION ADDRESS BASE
        MVI     E,0
        PUSH    D               ;SAVE LOAD ADDRESS FOR LATER JUMP TO CODE
;
        LXI     H,CODE$START    ;START MOVE POINTER TO (HL)
;
;
;LOOP TO MOVE CODE UP IN RAM UNDER CCP
;
MOVLOOP:
        MOV     A,B             ;CHECK BYTE COUNT TO SEE IF ALL MOVED YET
        ORA     C
        JZ      MOVDONE         ;EXIT LOOP IF DONE
;
        DCX     B               ;DECREMENT BYTES TO MOVE COUNT
        MOV     A,M             ;GET A BYTE TO MOVE
        STAX    D               ;SAVE AT DESTINATION ADDRESS
        INX     D               ;BUMP SOURCE DESTINATION POINTERS
        INX     H
        JMP     MOVLOOP         ;GO MOVE MORE BYTES
;
;
;CODE MOVED SO SET UP TO SCAN BIT MAP
;
MOVDONE:
        POP     D               ;GET BACK A COPY OF THE DESTINATION ADDR
        POP     B               ;RESET (BC) TO BYTE COUNT FOR BIT MAP SCAN
        PUSH    H               ;SAVE ADDRESS OF BIT MAP ON TOP OF STACK
;
        MOV     H,D             ;SET (H) TO RELOCATE PAGE OFFSET
        DCR     H
;
;
;LOOP TO SCAN CODE BLOCK JUST MOVED AND TO ADD IN OFFSET OF EXECUTION
;PAGE ADDRESS ON ALL BYTES NEEDING RELOCATION.
;
RELOCLOOP:
        MOV     A,B             ;CHECK BIT MAP COUNTER TO SEE IF RELOC DONE
        ORA     C
        JZ      RELOCDONE       ;EXIT IF ALL BYTES CHECKED
        DCX     B               ;DECREASE BYTE COUNT
        MOV     A,E             ;IS (DE) ADDRESS MOD EIGHT BYTES?
        ANI     7               ;IF SO WE NEED NEXT BIT MAP BYTE
        JNZ     SAMEBYTE        ;STILL ON SAME BIT MAP BYTE
;
;
;GET NEXT BIT MAP BYTE VIA POINTER ON TOP OF STACK
;
        XTHL                    ;SAVE (HL) AND GET CURRENT MAP POINTER
        MOV     A,M             ;GET MAP BYTE TO (A)
        INX     H               ;INCREASE POINTER FOR NEXT TIME
        XTHL                    ;PUT POINTER BACK ONTO STACK
        MOV     L,A             ;MAP BYTE TO HOLDING REGISTER
;
SAMEBYTE:
        MOV     A,L             ;FETCH OUR CURRENT BIT MAP BYTE
        RAL                     ;SHIFT BIT MAP BYTE FOR NEXT BIT
        MOV     L,A             ;SAVE SHIFTED ONE FOR NEXT PASS
        JNC     NOOFFSET        ;SKIP OFFSET ADD IF BIT WAS NOT ONE
;
;
;GET CODE BYTE AND ADD IN OFFSET IF MAP BIT WAS 1
;
        LDAX    D               ;FETCH THE DESTINATION BYTE
        ADD     H               ;ADD IN OFFSET
        STAX    D               ;STORE BACK AWAY
;
NOOFFSET:
        INX     D               ;INCREASE THE MOVED CODE BYTE POINTER
        JMP     RELOCLOOP       ;GO TO PROCESS MORE BYTES
;
;
;HERE WHEN THE RELOCATION IS DONE READY TO JUMP TO THE MOVED CODE
;REMEMBER THAT (H) HAS PAGE ADDRESS OF MOVED CODE
;
RELOCDONE:
        POP     D               ;GET BIT MAP POINTER OFF STACK
        MOV     D,H
        INR     D               ;SET UP EXECUTION ADDRESS
        MVI     E,0             ;MAKE (DE) AN EVEN PAGE BOUNDARY ADDRESS
        POP     H               ;GET THE SAVED CCP STACK POINTER
        SPHL                    ;RESET FOR GOING TO MOVED PROGRAM
        XCHG                    ;GET TRANSFER ADDRESS FROM (DE)
        PCHL                    ;OFF TO THE MOVED CODE AREA
;
        END
;
;
;+++...END OF FILE "PRLMOVE.ASM"
```

CONCLUSION OF LISTING 1

Listing 2

FILE: SINGLE.ASM (partial)

```
        TITLE 'SINGLE DENSITY ACCESS UTILITY VERSION 1.6 8/25/81'

;*****************************************************************
;       CALLAN DATA SYSTEMS CP/M SINGLE DENSITY DISK ACCESS PROGRAM
;*****************************************************************
;
;
; THIS PROGRAM IS A SMALL TRANSIENT PROGRAM BASED BIOS SUBSTITUTE
; THAT ALLOWS FILE TRANSFER UTILITY BETWEEN THE NORMAL MINI FLOPPY
; DISKS OR THE 5 1/4 INCH WINCHESTER BASED CP/M IN THE CALLAN MULTI-
; BUS BASED WORKSTATION. THIS CP/M TRANSIENT PROGRAM IS SETUP FOR
; VERSION 2.2 CP/M ON A MONOLITHIC SYSTEMS INC MSC 8009 Z80 PROCESSOR
; BOARD WITH 64K OF RAM. SINGLE DENSITY FLOPPY DISK I/O IS HANDLED
; VIA THE ONBOARD 1793 FLOPPY DISK CONTROLLER. THE SINGLE DENSITY DRIVE
; IS MEANT TO BE CONNECTED TO THE SAME CABLE AS THE NORMAL MINI FLOPPYS
; AND MUST BE SETUP AS UNIT 2. I.E, THE SINGLE DENSITY DRIVE MUST USE
; THE THIRD DRIVE SELECT LINE OF THE CABLE. THE DRIVE WOULD OBVIOUSLY
; BE MOUNTED AWAY FROM THE CALLAN WORKSTATION COMPUTER UNIT.
;
; THIS PROGRAM SUPPORTS THE DIGITAL RESEARCH INC. STANDARD SINGLE DENSITY
; EIGHT INCH DRIVE FORMAT FOR CP/M 1.4 OR 2.2 FILE STRUCTURES THE FORMAT
; IS AS DESCRIBED BELOW:
;
;         77 TRACKS
;         26 RECORDS PER TRACK
;         SINGLE SIDED
;         128 BYTE RECORDS
;         PHYSICAL SECTORS SKEWED FOR LOGICAL ACCESS WITH ORDER
;           OF: 1,7,13,19,25,5,11,17,23,3,9,15,21
;               2,8,14,20,26,6,12,18,24,4,10,16,22
;
; PROGRAM ACCESS TO THE EIGHT INCH FLOPPY IS DONE IN SINGLE 128 BYTE
; SECTORS. THE INITIAL LOADING OF THIS SOFTWARE SWAPS OUT THE NORMAL
; CP/M DISK I/O ENTRY POINTS TO THE BIOS WITH A NEW SET OF ENTRY POINTS
; TO THIS MODULE. THIS MODULE THEN CHECKS ALL DISK ACCESS FOR FLOPPY
; UNIT 2, (ACTUALLY REFERRED TO AS CP/M DISK DRIVE O: AS THE OUTSIDE
; DRIVE) AND WILL STEER I/O REQUESTS FOR SINGLE DENSITY I/O THROUGH
; DRIVERS CONTAINED WITHIN THIS PROGRAM. NOTE THAT THIS PROGRAM STILL
; DEPENDS UPON THE HOST CP/M SYSTEM BDOS AS THE FILE INTERFACE MEDIUM.
;
; OPERATION OF THE PROGRAM IS DONE TO MOVE THE MODULE UP TO A WORKSPACE
; BELOW THE MEMORY RESIDENT CCP. THE WARM BOOT VECTOR AT THE SYSTEM
; WARM BOOT ENTRY POINT IS SWAPPED TO A NEW ENTRY POINT WITHIN THE
; RELOCATED I/O MODULE. THE NEW WARMBOOT FUNCTION SIMPLY RE-ENTERSS
; THE ALREADY PRESENT CCP FOR FURTHER OPERATOR COMMAND PROCESSING.
; THE BDOS ENTRY VECTOR IS ALSO MODIFIED TO PERMIT THE DYNAMIC
; MODIFICATION OF THE USER PROGRAM AREA SIZE SUCH THAT THE CCP AND
; THE RELOCATED I/O MODULE DO NOT GET OVERLAYED BY THE TRANSIENT
; PROGRAM AREA'S BUFFER SPACES. THE UTILITY, WHEN LOADED PERFORMS
; A CHECK TO VERIFY WHETHER A RELOCATED MODULE IS ALREADY PRESENT
; IN MEMORY. THE ALREADY PRESENT CHECK IS DONE VIA A SPECIALLY DEFINED
; BDOS CALL THAT REQUESTS THE OPENING OF A FILE WITH THE SPECIALLY
; DEFINED NAME SEQUENCE OF "A,,,,,,,,,,," AS THE FILE NAME WHERE (A)
; IS A REQUEST NUMBER FOR PRESENCE CHECKING. AS THIS CHARACTER SEQUENCE
; IS AN ILLEGAL FILE NAME SEQUENCE, THE CHECK PROGRAM WILL TRAP THE NAME
; AND RETURN A ZERO BYTE IN THE (A) REGISTER IF THE MODULE IS PRESENT.
; IF THE ADDRESS BYTE IN THE FIRST BYTE OF THE FCB IS NOT RECOGNIZED,
; THEN THE MODULE PASSES THE OPEN FILE REQUEST ON TO THE NEXT HIGHER
; LEVEL BDOS CALL. IN ANY CASE THE NON PRESENCE OF A FILE BY THE NAME
; OF "A,,,,,,,,,,," IS VIRTUALLY ASSURED TO CAUSE THE BDOS TO RETURN
; A NOT FOUND "OFFH" ERROR CODE IN THE (A) REGISTER. THIS WOULD INDICATE
; THE ABSENCE OF THE MODULE BEING CHECKED FOR.
;
;
;*****************************************************************
;
;
;DEFINE TRUE AND FALSE ASSEMBLY PARAMETERS
;
TRUE    EQU     -1              ;DEFINE TRUE
FALSE   EQU     NOT TRUE        ;DEFINE FALSE
;
;
;DEFINE SINGLE DENSITY MODULE SELECT ADDRESS AS SPECIAL VALUE
;
```

```
MODADDR EQU     0E5H            ;ADDRESS OF THIS MODULE
;
;
;CP/M BDOS INTERFACE EQUATES
;
        ASEG
BOOT    EQU     0000H           ;FIXED BOOT ADDRESS
BDOS    EQU     0005H           ;FIXED BDOS ADDRESS
DEFFCB  EQU     005CH           ;DEFAULT FCB LOCATION
DEFBUF  EQU     0080H           ;DEFAULT SYSTEM BUFFER
RESET   EQU     13              ;RESET DISK SYSTEM
OPEN    EQU     15              ;OPEN FILE
STDMA   EQU     26              ;SET DMA ADDRESS
;
;
;ASCII CHARACTER DEFINITIONS
;
LF      EQU     00AH            ;ASCII LINE FEED CHARACTER
CR      EQU     00DH            ;ASCII CARRIAGE RETURN CHARACTER
;
;
;
;
FRETRY  EQU     10              ;RETRIES FOR FLOPPY IF USED
;
SGLDSK  EQU     02H             ;PHYSICAL UNIT SELECT CODE FOR 8" DRIVE
;
;
;SECTOR DEBLOCKING ALGORITHMS FOR CP/M 2.2
;
        MACLIB  DISKDEF
;
SMASK   MACRO   HBLK            ;UTILITY MACRO TO COMPUTE SECTOR MASK
;
;       COMPUTE LOG2(HBLK), RETURN @X AS RESULT
;       (2 ** @X = HBLK ON RETURN)
;
@Y      SET     HBLK
@X      SET     0
;
;       COUNT RIGHT SHIFTS OF @Y UNTIL = 1
;
        REPT    8
        IF      @Y = 1
        EXITM
        ENDIF
;
;       @Y IS NOT 1, SHIFT RIGHT ONE POSITION
;
@Y      SET     @Y SHR 1
@X      SET     @X + 1
        ENDM
        ENDM
;
;
;MACRO DEFINITIONS FOR Z-80 BLOCK MOVE INSTRUCTIONS REQUIRED FOR DISK
;DATA INTERFACE ON THE MSC 8009 BOARD
;
OUTIR   MACRO
        DB      0EDH,0B3H       ;Z80 OUTPUT BLOCK MOVE
        ENDM
;
INIR    MACRO
        DB      0EDH,0B2H       ;Z80 INPUT BLOCK MOVE
        ENDM
;
;
;CP/M 2.2 TO HOST DISK CONSTANTS
;
BLKSIZ  EQU     1024            ;CP/M ALLOCATION SIZE
HSTSIZ  EQU     128             ;HOST DISK SECTOR SIZE
;
FDSPT   EQU     26              ;HOST FLOPPY DISK 256 BYTE SECTOR PAIRS
                                ;PER TRACK
HSTBLK  EQU     HSTSIZ/128      ;CP/M SECTS/HOST BUFF
;
SECMSK  EQU     HSTBLK-1        ;SECTOR MASK
        SMASK   HSTBLK          ;COMPUTE SECTOR MASK
SECSHF  EQU     @X              ;LOG2(HSTBLK)
;
;SECTOR SKEW INTERLACE FACTOR
;
SKEW    EQU     00              ;SECTOR SKEW FACTOR
;
;*****************************************************************************
;
;
        CSEG                    ;SET ORIGIN TO ZERO FOR RELOCATABLE
                                ;ASSEMBLY BY RMAC
;
;
;FIRST TIME START UP ENTRY POINT FOR THE SINGLE DENSITY
;AUTO RELOCATING I/O MODULE. ENTRY HERE ASSURES PRESENSE OF
;CP/M 2.2.
;
        JMP     CHKPRES         ;GO CHECK IF A MODULE OF
                                ;SAME FUNCTION ADDRESS IS PRESENT
                                ;IN SYSTEM
;
;
;DUMMY SPACE OFFSET TO PUT NEW BDOS ENTRY CODE AT A 06H PAGE
;BOUNDARY OFFSET TO PERMIT DIGITAL RESEARCH DDT AND XSUB TO
;OPERATE WITHOUT A GLITCH
;
        DS      3
;
;NEWLY DEFINED ENTRY POINT INTO THIS MODULE FOR THE BDOS
;CALL VECTOR FUNCTION.
;
L5ENT:
        JMP     BDOS$SCAN       ;TO ENTRY POINT TO SCAN BDOS
                                ;FUNCTION CALLS
;
;
;SUBSTITUTE BDOS ENTR POINT. EXECUTION ADDRESS IS PLACED HERE
;FROM LOCATION 6 & 7 BY THE START UP MODULE PROVIDED THIS
;MODULE IS DETERMINED TO NOT ALREADY BE IN MEMORY.
;
TO$BDOS:
        JMP     $-$             ;ENTER ADDRESS AT STARTUP
;
;
;START UP CHECK ROUTINE TO SEE IF THIS SOFTWARE WAS ALREADY
;LOADED BY A PREVIOUS OPERATOR COMMAND.
;
CHKPRES:
        MVI     C,OPEN          ;ATTEMPT TO OPEN FILE "A,,,,,,,,,,"
        LXI     D,CHKFCB        ;POINT AT THE CHECK FCB
        CALL    BDOS            ;CALL NORMAL BDOS ADDRESS
        ORA     A
        JNZ     NOT$PRES        ;NON ZERO RETURN MODULE IS NOT
                                ;..PRESENT
        LXI     D,PRESMSG       ;POINT TO PRESENT MESSAGE
        MVI     C,9             ;PRINT FUNCTION CODE
        CALL    BDOS            ;PRINT ALREADY PRESENT MESSAGE
        RET                     ;SIMPLE RETURN TO THE CCP
;
CHKFCB:
        DB      0,MODADDR,',,,,,,,,,',0,0,0,0
        DS      16
        DB      0
;
PRESMSG:
        DB      CR,LF,'Single Density Access Already Active','$'
;
;
;HERE IF THIS RELOCATED MODULE IS NOT PRESENT IN MEMORY
;
NOT$PRES:
        LXI     H,0             ;SAVE OFF THE CCP STACK POINTER
        DAD     SP
        SHLD    CCP$STACK       ;SAVE THAT RETURN ADDRESS
;
        LHLD    BOOT+1          ;GET ADDRESS OF OLD WARM BOOT VECTOR
        SHLD    OLDBOOT         ;SAVE AWAY FOR FUTURE EXIT DESIGN
        LXI     H,XFRTAB+3      ;SET NEW WARM BOOT OVERRIDE ENTRY
        SHLD    BOOT+1
;
        LHLD    BDOS+1          ;GET PREVIOUS BDOS ADDRESS
        SHLD    TO$BDOS+1       ;SET TO LOCAL REFERENCE VECTOR
;
        LXI     H,L5ENT         ;SET NEW BDOS ENTRY TO THIS MODULE
        SHLD    BDOS+1          ;BASE+6
;
;
;INITIALIZE ALL ITEMS FOR USE IN THIS I/O HANDLER
;
        MVI     B,ENDZ-STARTZ   ;ZERO DATA AREA IN PARAMETER TABLE
        LXI     H,STARTZ
ZLP:
        MVI     M,00H           ;PUT IN A ZERO PARM BYTE
        INX     H               ;POINT TO NEXT BYTE TO BE ZEROED
        DCR     B               ;CHECK BYTE COUNT TO SEE IF DONE
        JNZ     ZLP
;
        LXI     H,DRVOSEL       ;DISABLE DRIVE SELECT FOR "O:"
        MVI     M,00H
;
        LXI     H,TRTAB         ;SET TRACK TABLE TO NON LOGGED
        MVI     M,0FFH
;
        MVI     A,SGLDSK        ;RESTORE SINGLE DENSITY DISK
        STA     HSTDSK
        CALL    FRESTR
        CALL    MOVDN           ;MOVE DOWN THE BIOS VECTOR TABLE
        LXI     H,BOOTENT       ;SET MOVED DOWN TABLE TO LOCAL BOOT HANDLER
        SHLD    BWBOOT+1
;
        CALL    PRTMSG          ;PRINT SIGNON MESSAGE
;
        DB      CR,LF,'Micro Resources Systems Single Density Disk'
        DB      CR,LF,'Access Utility Version 2.3 of 10/7/81'
        DB      CR,LF,0
;
;
;
;HAVE THIS UTILITY QUEUE BOTH BIOS AND THIS DRIVER TO THE SAME
;CP/M DATA BUFFER ADDRESS
;
        LXI     D,DEFBUF        ;USE DEFAULT BUFFER
        MVI     C,STDMA         ;SET DMA FUNCTION CODE
        CALL    TO$BDOS
;
;
;RETURN TO CCP VIA THE OLD DEFINED REENTRY POINT
;
CCPGO:
        LXI     H,L5ENT         ;RESET NEW BDOS ENTRY TO THIS MODULE
        SHLD    BDOS+1          ;BASE+6
        LHLD    CCP$STACK       ;GET THE CCP ENTRY STACK
        SPHL
        RET                     ;BACK TO CCP
;
;
;NEW WARM BOOT ENTRY LOCATION THAT RESETS THE DISK SYSTEM
;AND TRANSFERS CONTROL BACK TO THE ALREADY PRESENT CCP
;
BOOTENT:
        JMP     CCPGO           ;NOW GO BACK TO THE CCP
;
;
;HERE FROM A BDOS ENTRY TO TRAP FILE OPEN I/O TO CHECK FOR
;MODULE PRESENT CHECK.
;
BDOS$SCAN:
        PUSH    D               ;SAVE CALLERS PARAMETERS
        PUSH    B
        MOV     A,C             ;GET FUNCTION CODE TO A
        CPI     OPEN            ;SEE IF THIS IS AN OPEN FUNCTION
        JNZ     CHKFAIL
        INX     D               ;POINT TO FCB CHECK BYTE
        LXI     H,10            ;SET SCAN COUNTER TO FAKE FILE NAME END
        DAD     D
```

(continued next page)

```
        MVI     B,10            ;NUMBER OF "," TO CHECK FOR
SCAN$LOOP:
        MOV     A,M             ;GET FILE NAME CHARACTER
        CPI     ','
        JNZ     CHKFAIL         ;PASS ON IF CHECK FAIL
        DCX     H               ;DECREASE BUFFER POINTER
        DCR     B
        JNZ     SCAN$LOOP       ;CHECKED ALL POSSIBLE CHARS YET
        MOV     A,M             ;CHECK IF ADDRESS BYTE IS OURS
        CPI     MODADDR
        JNZ     CHKFAIL         ;BALE OUT IF NOT
        XRA     A               ;RETURN ZERO BYTE IF ALL CHECK VALID
        POP     B
        POP     D
        RET                     ;BACK TO PRESENT CHECKER
;
CHKFAIL:                        ;PROPER OPEN CHECK FAIL
        POP     B
        POP     D
        JMP     TO$BDOS         ;OFF TO THE NORMAL BDOS ROUTINE
;
;
;
XFRTAB:
;
;       SUBSTITUTE BIOS VECTOR TABLE. THIS JUMP TABLE VECTORS ALL CP/M
;       DISK I/O TO THIS TRANSIENT MODULE FIRST. TABLE IS PUT INTO THE
;       BIOS VECTOR TABLE POSITION BY A CALL TO THE SUBROUTINE "MOVDN"
;
;
        JMP     BCBOOT          ;TO NORMAL BIOS COLD BOOT ROUTINE
        JMP     BOOTENT         ;TO LOCAL WARM BOOT HANDLER
        JMP     BCSTAT          ;TO NORMAL BIOS CONSOLE STATUS CHECK
        JMP     BCIN            ;TO NORMAL BIOS CONSOLE INPUT
        JMP     BCOUT           ;TO NORMAL BIOS CONSOLE OUTPUT
        JMP     BLOUT           ;TO NORMAL BIOS LPT OUTPUT
        JMP     BPUN            ;TO NORMAL BIOS PUNCH OUTPUT
        JMP     BRDR            ;TO NORMAL BIOS READER INPUT
        JMP     HOME            ;MOVE DISK TO TRACK ZERO
        JMP     SELDSK          ;SELECT DISK DRIVE
        JMP     SETTRK          ;SEEK TO TRACK IN REG A
        JMP     SETSEC          ;SET SECTOR NUMBER
        JMP     SETDMA          ;SET DISK STARTING ADR
        JMP     READ            ;READ SELECTED SECTOR
        JMP     WRITE           ;WRITE SELECTED SECTOR
        JMP     BLSTST          ;GO RIGHT TO NORMAL BIOS FOR THIS I/O
        JMP     SECTRAN         ;SECTOR TRANSLATE
;
;
LOCTAB:
;
;       LOCAL COPY OF THE ORIGINAL BIOS DISK I/O VECTOR TABLE
;       INITIALIZED BY CALLING THE "MOVDN" SUBROUTINE.
;
BCBOOT:
        JMP     $-$             ;TO BIOS COLD BOOT ROUTINE
BWBOOT:
        JMP     $-$             ;TO BIOS WARM BOOT ROUTINE
BCSTAT:
        JMP     $-$             ;TO BIOS CONSOLE STATUS CHECK
BCIN:
        JMP     $-$             ;TO BIOS CONSOLE INPUT
BCOUT:
        JMP     $-$             ;TO BIOS CONSOLE OUTPUT
BLOUT:
        JMP     $-$             ;TO BIOS LPT OUTPUT
BPUN:
        JMP     $-$             ;TO BIOS PUNCH OUTPUT
BRDR:
        JMP     $-$             ;TO BIOS READER INPUT
BHOME:
        JMP     $-$             ;TO BIOS HOME DISK ROUTINE
BSELDSK:
        JMP     $-$             ;TO BIOS SELECT DISK ROUTINE
BSETTRK:
        JMP     $-$             ;TO BIOS SET TRACK ROUTINE
BSETSEC:
        JMP     $-$             ;TO BIOS SET SECTOR ROUTINE
BSETDMA:
        JMP     $-$             ;TO BIOS SET DMA ADDRESS ROUTINE
BREAD:
        JMP     $-$             ;TO BIOS SECTOR READ ROUTINE
BWRITE:
        JMP     $-$             ;TO BIOS SECTOR WRITE ROUTINE
BLSTST:
        JMP     $-$             ;TO BIOS LIST STATUS ROUTINE
BSTRAN:
        JMP     $-$             ;TO BIOS SECTOR TRANSLATE ROUTINE
;
;
;SUBROUTINE TO INTERCHANGE BIOS DISK I/O VECTOR TABLE ENTRIES WITH
;THOSE CONTAINED LOCALLY.
;
TABSIZ  EQU     17*3            ;TABLE SIZE TO INITIALIZE WITH 17 JMP'S
;
;
MOVDN:
        LHLD    OLDBOOT         ;GET ORIGINAL WARM BOOT VECTOR POINTER
        DCX     H               ;ADJUST TO BASE OF COLD BOOT VECTOR
        DCX     H
        DCX     H
        MVI     A,TABSIZ        ;SET BYTE COUNT TO MOVE
        STA     BYTCNT
        LXI     D,LOCTAB        ;POINT TO LOCAL TABLE FILL FROM ABOVE
        LXI     B,XFRTAB        ;POINT TO TABLE TO MOVE UP
MDLP:
        MOV     A,M             ;GET A BIOS TABLE BYTE
        STAX    D               ;PUT IN LOCAL COPY TABLE
        LDAX    B               ;GET BYTE OF PATCH TABLE
        MOV     M,A             ;PUT PATCH BYTE INTO BIOS POSITION
        INX     H               ;MOVE UP TO NEXT BYTE
        INX     D
        INX     B
        LDA     BYTCNT          ;SEE IF DONE YET
        DCR     A
        STA     BYTCNT
        JNZ     MDLP            ;CONTINUE IF NOT DONE YET
        RET
;
BYTCNT:
        DB      0               ;LOCAL MOVE BYTE COUNTER
;
;
;PARAMETER TABLE FOR SINGLE EIGHT INCH SINGLE DENSITY
;FLOPPY DISK DRIVE.
;
        DISKS   1               ;ONE 8" DRIVE SUPPORTED
        DISKDEF 0,1,26,,1024,243,64,64,2
;
;
SELDSK:
        MOV     A,C             ;GET NEW UNIT NUMBER
        CPI     'O'-041H        ;IS THIS OUR DRIVE?
        JZ      SDSK1           ;IF SO THEN GIVE THEM A PARAMETER POINTER
        XRA     A               ;IF NOT CLEAR THE DRIVE "O" SELECT FLAG
        STA     DRVOSEL
        JMP     BSELDSK         ;IF NOT FOR US THEN LET BIOS HAVE SELECT
;
;
;HERE IF DRIVE SELECT WAS FOR THIS PIECE OF SOFTWARE
;
SDSK1:
        MVI     A,0FFH          ;SET THE DRIVE "O" SELECT FLAG
        STA     DRVOSEL
;
        MVI     A,SGLDSK        ;SET HSTDSK TO SELECT CODE FOR SINGLE DENSITY
        STA     HSTDSK
;
        LXI     H,DPBASE        ;H&L REGS. = DISK PARAMETER POINTER
        XRA     A               ;SET A REG. = 00
        RET                     ;RETURN FROM SELDSK
;
;
;MOVE DISK TO TRACK ZERO
;
HOME:
        LDA     DRVOSEL         ;SEE IF RESTORE FOR US
        ORA     A
        JZ      BHOME           ;NO MUST BE FOR BIOS DRIVE
;
        LXI     H,0000H         ;ITS LIKE WE ARE GOING TO TRACK 0
        SHLD    HSTTRK
;
        CALL    FRESTR          ;DO RESTORE TO UNIT IN "HSTDSK"
        ORA     A
        JNZ     HOMEERR
;
HOMEOK:
        XRA     A
        STA     ERFLAG          ;RESET ERROR FLAG
        RET                     ;RETURN FROM HOME, IF O.K.
;
HOMEERR:
        CALL    EREXIT          ;PRINT UTILITY ERROR MESSAGE
        MVI     A,01H           ;SET ERROR FLAG
        STA     ERFLAG
        RET
;
;
;SET TRACK NUMBER SPECIFIED BY B&C REGS.
;
SETTRK:
        LDA     DRVOSEL         ;SEE IF TRACK FOR US
        ORA     A
        JZ      BSETTRK         ;TO PROM IF NOT LOCAL
;
        MOV     H,B
        MOV     L,C
        SHLD    HSTTRK          ;TRACK TO SEEK
        RET
;
;
;
;TRANSLATE THE SECTOR GIVEN BY B&C REGS.
;
;       NO TRANSLATE DONE AT THIS TIME. WE WILL TRANSLATE THE
;       FLOPPY SECTOR AT THE PHYSICAL SECTOR BASIS IN THE
;       FLOPPY READ/WRITE SETUP ROUTINE.
;
SECTRAN:
        LDA     DRVOSEL         ;SEE IF SECTRAN FOR US
        ORA     A
        JZ      BSTRAN          ;TO BIOS IF NOT LOCAL
;
        MOV     H,B
        MOV     L,C
        RET                     ;RETURN FROM SECTRAN
;
;
;SET DISK SECTOR NUMBER
;
SETSEC:
        LDA     DRVOSEL         ;SEE IF SECTOR FOR US
        ORA     A
        JZ      BSETSEC         ;TO PROM IF NOT LOCAL
;
        MOV     A,C             ;GET SECTOR NUMBER
        STA     HSTSEC          ;SECTOR TO SEEK
        RET                     ;RETURN FROM SETSEC
;
;
;SET DISK DMA ADDRESS
;
SETDMA:
        PUSH    H
        MOV     H,B             ;MOVE B&C TO H&L
        MOV     L,C
        SHLD    DMAADR          ;PUT AT DMA ADR ADDRESS
        POP     H
        JMP     BSETDMA         ;TELL BIOS DMA ADDRESS
;
;
;WRITE THE SELECTED CP/M 2.2 SECTOR
;
WRITE:
        LDA     DRVOSEL         ;IS THIS WRITE FOR HERE
        ORA     A
        JZ      BWRITE          ;TO BIOS IF SO
;
```

```
;
WRTSEC:
        IN      CURTRK          ;SAVE 1791 TRACK VALUE IF BIOS IS
        STA     SAVTRK          ;ALSO USING SAME DISK CONTROLLER
        IN      UNITMO          ;SAVE CURRENT UNIT MODE WORD IF USED
        STA     UMSAV
;
        CALL    SETV            ;SETUP FLOPPY DISK UNIT/TRACK/SECTOR
        ORA     A
        JNZ     WRTERR          ;EXIT FOR SEEK ERROR
        LHLD    DMAADR          ;WRITE DATA FROM DMAADR
        SHLD    XFRPNT
        CALL    FPUTSEC         ;GO WRITE FLOPPY SECTOR
;
        PUSH    PSW
        LDA     SAVTRK          ;RESTORE TRACK OF 1791
        OUT     CURTRK
        LDA     UMSAV           ;RESTORE BIOS UNIT MODE WORD IF USED
        OUT     UNITMO
        POP     PSW
;
        ORA     A
        JNZ     WRTERR          ;EXIT FOR WRITE ERROR
;
WRTOK:
        XRA     A
        STA     ERFLAG          ;RESET ERROR FLAG
        RET                     ;RETURN FROM "WRITEHST", IF O.K.
;
WRTERR:
        CALL    EREXIT          ;PRINT BIOS ERROR MESSAGE
        MVI     A,01H           ;SET ERROR FLAG
        STA     ERFLAG
        RET
;
;
;READ THE SELECTED CP/M 2.2 SECTOR
;
READ:
        LDA     DRVOSEL         ;SEE IF OPERATION FOR US
        ORA     A
        JZ      BREAD           ;GO READ IN BIOS IF NOT FOR US LOCAL
;
;
READSEC:
        IN      CURTRK          ;SAVE 1791 TRACK VALUE IF USED BY BIOS
        STA     SAVTRK
        IN      UNITMO          ;SAVE CURRENT UNIT MODE WORD TOO
        STA     UMSAV
;
        CALL    SETV            ;SETUP FLOPPY DISK UNIT/TRACK/SECTOR
        ORA     A
        JNZ     RDERR           ;EXIT IF SEEK ERROR
        LHLD    DMAADR          ;POINT TO USER BUFFER
        SHLD    XFRPNT
        CALL    FGETSEC         ;GO READ SECTOR
;
        PUSH    PSW
        LDA     SAVTRK          ;RESTORE TRACK OF 1791
        OUT     CURTRK
        LDA     UMSAV           ;RESTORE BIOS UNIT MODE WORD IF USED
        OUT     UNITMO
        POP     PSW
;
        ORA     A
        JNZ     RDERR           ;EXIT FOR READ ERROR
;
RDOK:
        XRA     A
        STA     ERFLAG
        RET
;
RDERR:
        CALL    EREXIT          ;PRINT BIOS ERROR MESSAGE
        MVI     A,01H           ;SET ERROR FLAG
        STA     ERFLAG
        RET
;
;
;READ/WRITE BIOS ERROR PRINT ROUTINE
;WAITS FOR USER RESPONSE ON CONSOLE AFTER PRINTING MESSAGE
;AND THEN RETURNS TO THE CALLER
;
EREXIT:
        STA     IOERR           ;SAVE ERROR STATUS
        CALL    PRTMSG
;
        DB      CR,LF,CR,LF,' *** 8" FLOPPY'
        DB      ' DISK I/O ERROR - CODE(',0
;
        LDA     IOERR           ;PRINT ERROR CODE
        PUSH    PSW
        RRC
        RRC
        RRC
        RRC
        CALL    HEXOUT
        POP     PSW
        CALL    HEXOUT
        CALL    PRTMSG
;
        DB      ') *** ',0
;
        CALL    BCIN            ;WAIT FOR OPERATOR RESPONSE
        CPI     'C'-040H        ;IF CTL-C THEN GO DO WARM BOOT
        JZ      BCBOOT          ;LET HOST REBOOT IF OPERATOR WANTS THAT
;
        CALL    PRTMSG
;
        DB      CR,LF,0
;
        RET
;
;
;SUBROUTINE TO SEND A HEX CHAR TO THE CONSOLE
;
HEXOUT:
        ANI     0FH             ;LOW NIBBLE ONLY
        CPI     0AH             ;GREATER THAN 9?
        JM      HEX1
        ADI     07H             ;ADD OFFSET FOR A-F
HEX1:
        ADI     030H            ;ADD IN ASCII OFFSET
        MOV     C,A             ;PRINT RESULT
        CALL    BCOUT
        RET
;
;
```

```
;*****************************************************************
;
;       FLOPPY DISK CONTROLLER I/O ROUTINES
;
;
;<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
;<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
;<><>                                                                 <><>
;<><>     This part of the program SINGLE contains                    <><>
;<><>     entry points for 8" floppy I/O on the                       <><>
;<><>     MSC 8009A board. These I/O drivers have                     <><>
;<><>     been left out due to magazine space con-                    <><>
;<><>     siderations. The complete program is                        <><>
;<><>     available from the author. The program                      <><>
;<><>     entry points were:                                          <><>
;<><>                                                                 <><>
;<><>     SETV: To setup foppu unit select and per-                   <><>
;<><>           form seek if necessary. Also program                  <><>
;<><>           sector number to disk controller.                     <><>
;<><>     FPUTSEC: To write physical sector to single                 <><>
;<><>           density floppy diskette.                              <><>
;<><>     FGETSEC: To read physical sector from                       <><>
;<><>           single density floppy diskette.                       <><>
;<><>     FRESTR: To restore selected unit number                     <><>
;<><>           to track zero.                                        <><>
;<><>                                                                 <><>
;<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
;<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
;
;
;INLINE PRINT OF MESSAGE TILL A ZERO
;
PRTMSG:
        XTHL                    ;SAVE HL, GET MSG POINTER
;
PRTMLP:
        MOV     C,M             ;GET CHARACTER
        INX     H               ;INCREMENT POINTER TO NEXT CHAR
                                ;..OR RETURN ADDRESS
        MOV     A,C             ;CHECK IF ZERO END
        ORA     A
        JZ      PMXIT           ;EXIT IF ZERO
;
        CALL    BCOUT           ;OUTPUT IT
        JMP     PRTMLP          ;GO CHECK/DO NEXT CHAR
;
PMXIT:
        XTHL                    ;RESTORE HL, RET ADDR
        RET                     ;RET PAST MSG
;
;
;*****************************************************************
;
;       STORAGE AREA FOR VARIABLES BEGINS HERE...
;
;
;RELOCATION POINTER STORAGE AREA
;
CCP$STACK:
        DS      2               ;STORE CCP RE-ENTRY STACK POINTER HERE
OLDBOOT:
        DS      2               ;STORE OLD VALUE OF BIOS VECTOR
                                ;TABLE POINTER HERE
;
;
;I/O ERROR CODE
;
IOERR:
        DS      1               ;DISK I/O ERROR TYPE
;
;
;START POINTER FOR BEGIN EXECUTION PARAMETER CLEARING
;
STARTZ  EQU     $
;
;
;HOST DISK PARAMETER DATA AREA
;
FERCNT:
        DS      1               ;FLOPPY SEEK RETRY COUNTER
FRWCNT:
        DS      1               ;FLOPPY READ/WRITE RETRY COUNTER
;
HSTDSK:
        DS      1               ;HOST DISK NUMBER
HSTTRK:
        DS      2               ;HOST TRACK NUMBER
HSTSEC:
        DS      1               ;HOST SECTOR NUMBER
;
ERFLAG:
        DS      1               ;ERROR REPORTING
DMAADR:
        DS      2               ;DISK DMA TRANSFER ADDRESS
XFRPNT:
        DS      2               ;FLOPPY I/O ROUTINE DATA POINTER
;
;
;END POINTER FOR BEGIN EXECUTION DATA AREA ZERO LOOP
;
ENDZ    EQU     $
;
TRTAB:
        DS      1               ;TRACK TABLE FOR 8" DISK
;
DRVOSEL:
        DS      1               ;DRIVE SELECTED FLAG
SAVTRK:
        DS      1               ;BYTE TO STORE TRACK FROM THE 1791
UMSAV:
        DS      1               ;BYTE TO SAVE POSSIBLE BIOS UNIT MODE WORD
;
;
;SCRATCH RAM AREA FOR BDOS USE
;
        ENDEF                   ;LET DISKDEF FIXUP BDOS BUFFERS
;
;
        END
;
;
;+++...END OF FILE "SINGLE.ASM"
```

CONCLUSION OF LISTING 2

Gift Subscriptions

You should consider gift subscriptions to *Lifelines/The Software Magazine* for your friends and relatives who are involved in microcomputing. As you probably realize from your own experience, the price of a subscription is small for the money *Lifelines* can save you in a year. Just send a check or credit card number and fill out the form below*. (Or call [212] 722-1700.) We'll send your gifted one a note to let them know of their good fortune, *and* we'll send you a free Zoso T-shirt. (Don't forget to tell us your size.)

Your name and address:

Name_____
Address_____
City _____ State _____ Zip _____

Shirt size _____
☐ Check enclosed
VISA or MasterCard Number
Expiration Date

_____

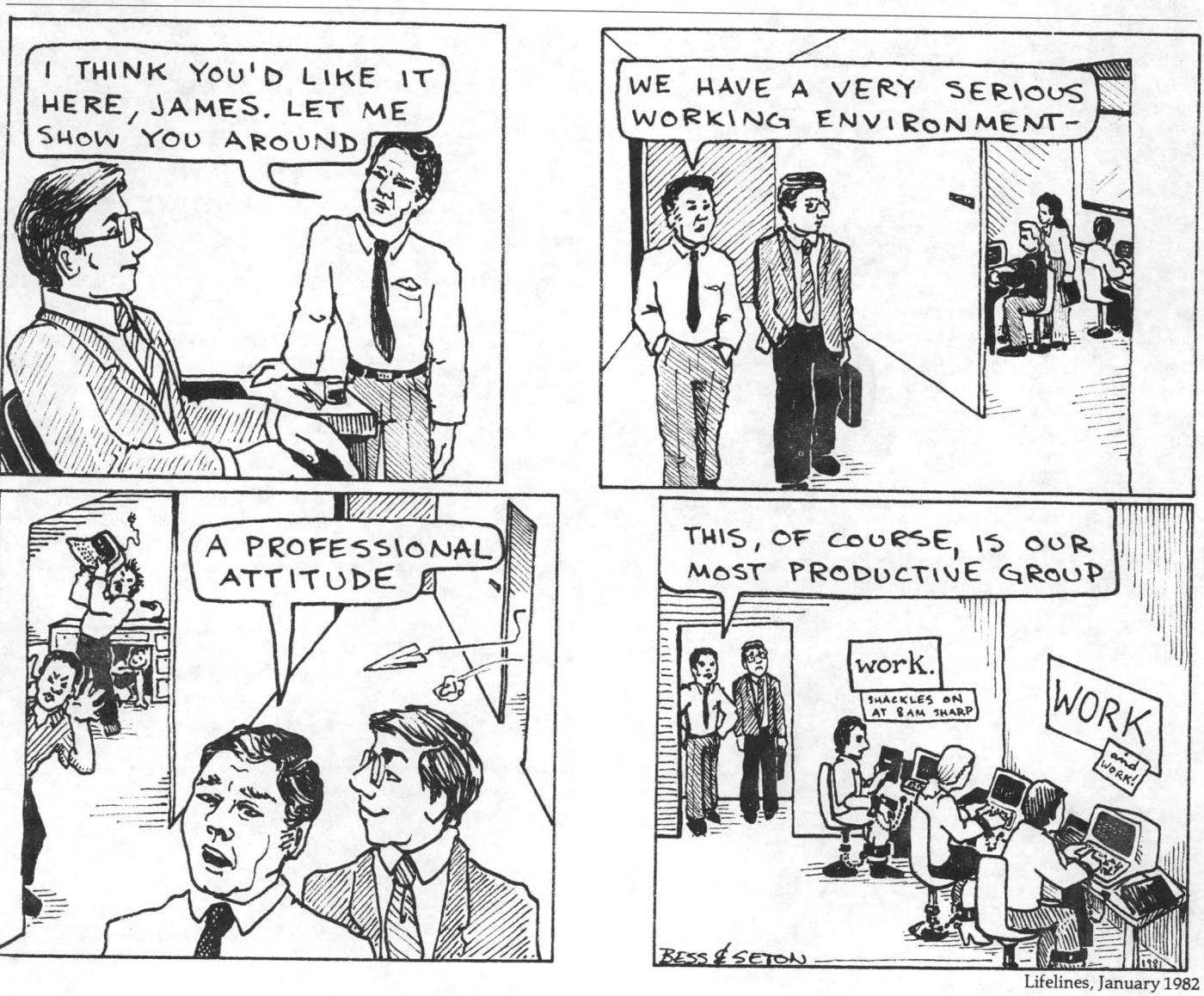
_____
Signature (if payment is by credit card)

The name and address of the gifted one:

Name_____
Address_____
City _____ State _____ Zip _____

*All orders must be prepaid by VISA, MasterCard or check. Checks must be in U.S. $, drawn on a U.S. bank. Subscription rates are $18 for twelve issues (one year) when the destination is the U.S., Canada, or Mexico. For subscriptions going to all other countries, the price is $40 for twelve issues.

KIBITS



(continued from page 4)
more than likely have never heard of, has developed a series of hardware/software packages that turn your system into a real data handling power house.

What CDI has done is take the Heath/Zenith 89 microcomputer and add some very interesting value in the form of both hardware and software; they've renamed the system Microcom.

What you get for $5995 is a portable communications system: the basic Z-89 with 64K RAM and an intelligent Modem along with CP/M. The modem is what makes the system unique since it compiles either 300 or 1200 baud capability with full error checking and permits transmission even over very low grade lines. In addition, for another $495 you can pick up Image, which has to be one of the more powerful word processing packages available.

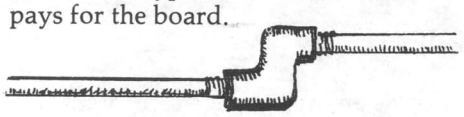
The Image package not only gives you the standard functions you'd expect in a word processor, but adds such things as real column handling, font definition, ability to type in any direction: right, left, up or down. But the real excitement is the ability to handle graphics right inside the package. Should you be doing a document with flowcharts, for example, draw them on the screen exactly they way you want them. Then protect the fields and enter information directly in the boxes.

Because word processors aren't any good unless you can get the words on paper, CDI has developed drives for the NEC Spinwriter that allows full graphics output. They should have other drivers available by this month.

If you need a forms generator to help you develop documents, CDI has that for an additional $295, and another $295 brings you Term, the communication package designed to work in tandem with the Image. This latter package not only gives you the ability to communicate over telephone lines but handles networking functions at the same time. This last function, called Comnet, supports up to 255 stations at 56K bits/sec with 2000-ft spacing between repeaters.

If you're worried because you don't have an 89, by NCC time CDI plans to offer the modem and software packages for any CP/M based system.

And if you don't have an NEC Spinwriter, you can achieve a high level of printer smarts with Integrator from Intek (780 Charcot Ave, San Jose, CA 95131 [408] 946-9041) for $650. This add-in board lets your Diablo 630 handle unique word processing functions, thus unburdening the software, lets it have a buffer as large as 48K thus improving throughput, and lets it handle graphics. You don't have to have three EE degrees to install the board either; just unpack it and pop it in and choose the protocol you want. When you couple this with Image, you can have typesetting-like capabilities without the typesetter. And that alone pays for the board.



Installing SuperCalc With The "Secret" Menu

Michael Olfe

In version 1.22 and earlier versions of "SuperCalc" there is an undocumented menu which allows complete control of terminal configuration. Since some of the standard console installations (e.g. ADM-31, perhaps others) do not use the reverse video that the console has, it may be necessary to use this menu to implement reverse video. The "secret" menu will also be useful if you do not have one of the standard consoles on the main menu.

You get to the secret menu by first doing an installation from the main menu. When you set to the following menu, choose "C" (which is not on the menu).

This is what you may now do:

A. Save SuperCalc on disk.

B. Return to first menu.

X. Quit Install Program

Enter A,B or X or ? : C

The menu below will now come up. To implement highlighting, you must edit attribute data and miscellaneous data from this menu.

These are the items you may now edit:

A. Edit screen controls.

B. Edit attribute data.

C. Edit input keys.

D. Edit GotoXY, printer init. string.

E. Edit miscellaneous data.

F. Edit terminal name.

X. Finished editing data.

Enter A-F or X or ? : X

8080 Assembler Tutorial: Arithmetic Instructions and Stack Instructions

Ward Christensen

DAD TRICKS

Previously, I covered the 8 and 16 bit arithmetic of the 8080. Now, I'll show some uses and "tricks" with the 8080 16 bit add, DAD instruction.

I'll start with using DAD to do what does not appear even to *be* an ADD. DAD H is a 16-bit shift left in disguise. What happens if I code DAD H? HL gets added to itself, i.e. doubled. If HL contains 134CH, then executing DAD H would leave 2E998 in HL. Looking at this in binary, before DAD H:

0001 0111 0100 1100

after DAD H:

0010 1110 1001 1000

There is an interesting effect: doing a DAD H acts like a 16-bit SHIFT LEFT, and is sometimes coded for that reason.

Also of interest: When using DAD H as a shift left, the left-most bit shifted out is placed in the CARRY bit in the PSW.

If DAD can shift, are there other things (other than ADDING) that it can do? Read on...

MULTIPLYING USING THE DAD INSTRUCTION

If I have a 16 bit number which I want to multiply by 2, 4, 8, 16, (any small power of 2), I can simply code DAD instructions:

```
DAD   H      ;HL= 2 x HL
DAD   H      ;HL= 4 x HL
DAD   H      ;HL= 8 x HL
DAD   H      ;HL=16 x HL
DAD   H      ;HL=32 x HL
```

Thus coding 5 DAD H instructions multiplies HL by 32. More generally, "n" DAD H instructions multiply HL by 2 to the "n"th power.

You can also multiply by numbers other than powers of 2, by putting the original value in DE, and using the "proper combination" of DAD D and DAD H instructions.

For example, to multiply HL by three:

```
MOV   D,H    ;copy H into D
MOV   E,L    ;copy L into E
DAD   H      ;Twice,
DAD   D      ;+once = 3 times
```

or, a more commonly used example – multiplying by ten:

```
MOV   D,H    ;copy H into D
MOV   E,L    ;copy L into E
DAD   H      ;HL= 2 x HL
DAD   H      ;HL= 4 x HL
DAD   D      ;HL= 5 x HL
DAD   H      ;HL=10 x HL
```

These instructions are often found in a subroutine which converts a decimal ASCII character number into a binary number. Why? because the process involved is: "multiply by 10, then add in next digit".

This technique can be applied to multiplying by *any* fixed number (fixed, meaning known at the time of the assembly of the program, rather than being specified at execution time).

The reason this is possible, is that the process of multiplication may be broken down into two more "primitive" processes: shifting, which multiplies by 2, and addition.

The process is a simplification of the mental process you go through to multiply in decimal: it consists of shifting and multiplying a digit at a time.

In decimal, the possible values which you have to multiply by go from zero to ten. Binary is fortunate, in that the values may be only zero or one. Thus, there is no "multiply by 7" intermediate step in multiplying in binary. If you "multiply by one" it reduces to the simple case of addition.

In my example, the DAD D does the addition, and the DAD H does the shift left, or doubling.

I have worked out a "cookbook" for multiplying by a fixed number. Just follow these steps:

1. Write down the number to be multiplied by, in binary. Do not code any leading 0's. For example, to multiply by 25, write down:

   11001

   This is 25, because it represents:

```
    1 x 16 = 16
  + 1 x  8 =  8
  + 0 x  4 =  0
  + 0 x  2 =  0
  + 1 x  1 =  1
                ----
        Sum = 25
```

2. Between each digit, put an "H":
   1 H 1 H 0 H 0 H 1

3. Change every 1 to a D:
   D H D H 0 H 0 H D

4. Delete all the 0's:
   D H D H H H D

5. Now, delete the leading D — There will always be one because we said to write our number with no leading zeros, (step 1) then to change every D to a 1 (step 3).
   H D H H H D

6. You now have the order to execute DAD D and DAD H instructions to perform the multiply. Before doing so, you have to "set up" the DE register pair. Code the following instructions:

```
MOV   D,H    ;copy H into D
MOV   E,L    ;copy L into E
```

Then code the sequence of DADs produced in step 5 above.

I have commented to show the partial

results "along the way":

```
DAD   H      ;x 2
DAD   D      ;x 3
DAD   H      ;x 6
DAD   H      ;x12
DAD   H      ;x24
DAD   D      ;x25
```

Let's try multiplying by 7. Following the previous steps, you should have:

1. 1 1 1
2. 1 H 1 H 1
3. D H D H D
4. D H D H D
5. H D H D

and the resulting program:

```
MOV   D,H    ;copy H into D
MOV   E,L    ;copy L into E
DAD   H      ;HL= 2x
DAD   D      ;HL= 3x
DAD   H      ;HL= 6x
DAD   D      ;HL= 7x
```

The process should be pretty straightforward, so that you might in the future multiply by some number, even if you don't remember the details of the process.

If it "bugged" you to "throw away" the leading "D" in step 5, here's why: Because you had started with the number in HL as well as DE, no initial "DAD D" was done to move "one times the number" from DE into HL.

STACK INSTRUCTIONS

In the 8080, the stack is a reserved section of memory, which is pointed to by a 16 bit register, the stack pointer, "SP".

The stack works much like a "paper spindle" that is used to "spindle" paid bills on, in that it is "last-in-first-out" (LIFO).

If you take 3 pieces of paper, write "1" on the first, "2" on the second, and "3" on the third, then put them on a paper spindle in that order, you will find the last one "in" (3) will have to be the first one "out". To get at "2", you have to remove "3" first.

What use is a stack? As I noted in the definition of "STACK", a stack is used to keep track of the return addresses when calling subroutines, and to store data, such as the contents of registers, so you may modify the them, then restore their previous values.

Let's look at the way the stack works.

HOW THE STACK WORKS

The stack pointer must be initialized so that it points to a place in memory reserved for the stack of data.

Which "end" of the place in memory do you point the stack when initializing it? Well, the stack works *down* (meaning toward lower addresses), not *up* like my "paper spindle" example.

This means that the stack pointer is initialized to the END of the stack area. (Care to visualize a paper spindle pointing downward, glued to the ceiling???)

The stack pointer is initialized with an LXI instruction. For example, if I want to reserve 100 bytes for stack space, I would code the following in my program:

```
;
;define a 100 byte stack
;
        DS    100
STACK   EQU   $
```

The DS means to Define Storage; the EQU means "EQUated"; and $ means "here". Thus I have reserved 100 bytes, then put a label "here" right after the reserved bytes.

I can now initialize my stack pointer by coding:

```
LXI SP,STACK
```

When I put data on the stack, the 8080 will automatically decrement the stack pointer SP, BEFORE storing each byte, and will automatically increment SP after I get data from the stack.

INSTRUCTIONS TO ACCESS THE STACK

There are instructions, namely CALL and RET, which "implicitly" refer to the stack. They will be covered in the next tutorial section: "Control of Execution Sequence".

Other instructions reference the stack "explicitly". Read on...

PUSH and POP

To save the contents of registers on the stack, the PUSH instruction is used, and to restore it back into the registers, the POP instruction is used.

PUSH and POP always work on 16 bits of data. I can PUSH and POP any of the 3 register pairs, BC, DE, or HL:

```
PUSH B
PUSH D
PUSH H
```

then get them back at a later time, by coding:

```
POP H
POP D
POP B
```

NOTE the order of the POPs! It is opposite that of the PUSHes. This is "LIFO" at work again. Since H (implying HL) was the last register pair PUSHed, it must be the first POPped.

This sequence of instructions frequently appears in a subroutine, where you want to use the registers, but want the contents restored when you are done. Just PUSH all registers which you will modify, then POP them back when you are done.

SAVING THE ACCUMULATOR AND/OR THE PSW FLAGS

I can also PUSH and POP the accumulator, but it is only 8 bits, so the PSW (program status word) byte gets PUSHed and POPped along with it, and gives its name to the instructions: PUSH PSW and POP PSW.

Thus this single instruction is used either to save the accumulator, or the PSW bits, or both.

WHAT DOES THE STACK LOOK LIKE?

You need not be concerned about the order of the 2 bytes put on the stack by PUSH, but if you are interested: PUSH B decrements the stack pointer,

(continued next page)

places B into memory, decrements the stack pointer again, and places C into memory.

The same is true for PUSH D and PUSH H; namely D, then E is stored, or H then L. Following my earlier diagram of the registers, and considering the numbering of registers, I would have expect PUSH PSW to first put the PSW byte, then the accumulator, on the stack, but the reverse is true. Apparently they are not structured internally the way they are numbered externally.

Here are some samples, showing the contents of the stack after loading each register, then pushing all registers:

```
LXI   SP,200H ;uses 1FFH down
MVI   B,0
MVI   C,1
MVI   D,2
MVI   E,3
MVI   H,4
MVI   L,5
MVI   A,7
PUSH  PSW
PUSH  B
PUSH  D
PUSH  H
```

The stack from 1F8 to 1FF now contains:

```
1F8:  05
1F9:  04
1FA:  03
1FB:  02
1FC:  01
1FD:  00
1FE:  02
1FF:  07
```

As you can see, the PUSH PSW put the 07 (contents of A) on the stack, then the 02 (contents of the PSW).

MORE INSTRUCTIONS TO MANIPULATE THE STACK

The INX and DCX instructions apply to the stack pointer, but are infrequently used.

I have used INX SP to delete something on the stack which I no longer want (i.e. I pushed a register, but on some conditions I don't want it back). The easiest way to clear up the stack is by doing a "dummy" POP, being careful to comment it as such. This, however, takes a register pair and changes its value. Therefore, executing two INX SP instructions will "delete" the unwanted data from the stack, without using any registers.

I have heard people say: "Any program which does INX or DCX of the stack will not run under interrupts." This is *not* true in the case of using two INX SP instructions to delete an unwanted entry off the "end" of the stack.

A case where it *is* true may occur when you try to be "clever" and use INX SP several times in order to get back "up" to some previously pushed value on the stack, then try to "un-do" the INX SPs by issuing DCX SPs. This *will* work, but will likely cause problems running under interrupts, since the interrupt handling program may have overlaid the "unprotected" parts of the stack – parts that were "unprotected" by INX SP.

Since the stack deals with 16-bit quantities, you will usually want to do "INX SP" in "Pairs" so the stack stays "orderly".

That's it for the STACK related instructions. The next section, "Control of Execution Sequence", deals with "testing" conditions, and modifying the execution of the program based upon the results of that test.

The tests may be either "logical" instructions, or "comparison" instructions.

Next month I'll explain "control of the execution sequence", opening with the logical and comparison instructions, opening the way for the JMP, CALL, and RETurn instructions.

You might want to go back to the outline I presented at the beginning of the tutorial, and note that "logical and comparison instructions" will be covered in the "control of execution sequence" part of the tutorial.

I solicit your comments, questions, corrections, etc. write me C/O Lifelines, 1651 Third Ave., N.Y., N.Y. 10028.

A Review of T/MAKER II—Part 1 Features, Advantages, and Benefits

Raymond Sonoff

What makes any particular software package of interest? Certainly, it must offer essential benefits that make it worth considering in the first place. Secondly, it should provide distinct advantages over competitive offerings. Finally, its features must be presented in a manner that can be readily understood by potential customers for the product. T/MAKER II is a software package that addresses the market need for creating tables of data, calculations made from the raw data, and supporting text (words, symbols, superscripts, and subscripts, etc.) that aid in giving vitality, interest, and verbal significance to numerical data.

It is the purpose of this introductory review article to highlight features, advantages, and benefits of T/MAKER II. It is oriented towards readers who are either not intimately familiar with this particular software package or who want to know more about what makes T/MAKER II worthy of consideration.

BACKGROUND

T/MAKER is shorthand for Table MAKER. This task of table-making via microprocessor-based systems can be considered as a special case of word processing. An excellent review by Steve Patchen of two dominant products–VISICALC and T/MAKER–was published in Lifelines (2/1/81) on pages 2 through 5. Because a non-procedural language is employed, a user of either T/MAKER or of VISICALC can simply focus his efforts on what he wants done on the data. He does not have to concern himself with how the process is carried out. (All any of us need is another language to learn, right?). I initially decided to acquire T/MAKER II (rather than VISICALC) for two reasons: 1). T/MAKER works with CP/M-based operating systems (which is what I have); and 2). because I regularly have the requirement to generate technical reports that involve data calculations along with text processing, T/MAKER would give me this combined capability. As an additional endorsement for choosing T/MAKER, some comments regarding the primitive printing capabilities of VISICALC (see article by David Ahl in Creative Computing, Dec. 1981 issue, page 100) reinforced the selection of T/MAKER as the software package to adopt if I wanted to avoid wholly under-utilizing my Diablo 1622 printing terminal. In short, T/MAKER became the obvious choice.

The creator and copyright holder of T/MAKER II (an updated version of T/MAKER), is Peter Roizen; Lifeboat Associates is the exclusive distributor of this product. T/MAKER II is offered in CP/M-compatible formats on either 8" or 5-¼" diskettes. Documentation on T/MAKER II comes in a well-organized 3-ring binder manual. Included with the manual is a Quick Reference Card (QRC) that summarizes all the primary features and operations associated with T/MAKER II.

GETTING STARTED

Upon receipt of my software package from Lifeboat Associates, I proceeded to scan through the T/MAKER II manual. The manual is divided into four color-coded sections: (White) Reference Manual (98 pages); (Yellow) Tutorial (18 pages); (Pink) Initialization Notes (32 pages); and (Green) Addendum (9 pages). In the "Getting Started" section of the Reference Manual is the statement: 'You will have to read the third section and customize T/MAKER before you can use it.' Noting this, I formatted a blank diskette, did a "DISKTEST" utility on it to assure myself that there were no sector errors of any kind present, and I made a duplicate "working" copy of the master diskette, carefully storing away the master once this duplication process was completed. With a feeling of proper precautions having been taken, I continued my perusal of the various sections of the T/MAKER manual. When I noted that my particular system met all of the minimum system requirements cited in Appendix XV of the Pink Initialization Notes section, I began to feel that it was time to actually take the steps outlined in the Initialization Notes section of the manual.

Customizing T/MAKER is accomplished by changing the file labeled TMAKER.UTL. This file is not accessed directly but is altered by using a program called TMODIFY. This program allows up to fifteen user-selectable options to be called. From five to seven of these options have to do with the console terminal you intend to employ to talk with T/MAKER via your computer mainframe (if not a standalone desktop computer system, of course). I would like to say that all went smoothly. However, since my Heathkit/Zenith H19 intelligent terminal was not included as one of the several standard terminals to choose from under Option 12, I had to perform Options 7, 8, and 9 separately. Respectively, these TMODIFY options involve the number of rows and columns on the console (no problem), console control (I got sidetracked trying to use standard ANSI key code sequences rather than the ZDS mode of operation that the H19 initializes to upon powerup), and cursor addressing (which I did not pursue initially). Option 11 involved selection of the user-definable editing keystrokes.

After many hours of trial and error without any success, I phoned Michael Olfe at Lifeboat Associates. He quickly provided me with the appropriate decimal values associated with the ASCII keystroke sequences to enter to properly setup Option 8 of TMODIFY for an H19 configured for the ZDS mode of operation. Michael indicated that the more complicated ANSI mode requires additional keystrokes to satisfy Option 8, and he recommended my avoiding the ANSI mode in general. Upon taking his advice and using the Option 8 and Option 9 values supplied by him, I had a TMAKER.UTL file that resulted in T/MAKER II properly executing the self-tending demonstration programs included with the T/MAKER II disk-

(continued next page)

ette. It was a real joy to observe numerous examples of many of T/MAKER II's operations in action.

However, during activities involved with editing some files, I came to realize that some additional changes in several of the editing keystrokes would be required in order for my H19 terminal to remain in the ZDS mode or, as a matter of personal preference, prove easier for me to use. To remain in the ZDS mode, I had to eliminate the "ESC >" key sequence that was predefined as the command for breaking a line. This is because the "ESC >" is recognized by the H19 terminal as a command to enter the ANSI mode of operation, thereby leaving the ZDS mode (upon which all the codes that Michael Olfe gave me were based). I selected "ESC -" as a replacement keystroke sequence for this break line command, and this change alone eliminated the 'dropout' problem that occurred when "ESC >" was used. As a matter of convenience, the complementary command "ESC <" for joining lines was changed to "ESC +", and a few other keystroke sequences were changed to satisfy personal preferences. After this set of changes, the T/MAKER II software package was off and running.

FEATURES

T/MAKER II is perhaps best described in terms of functions accomplished by reserved names associated with utility files. These may or may not require filenames to be operational.

Categories of functions. Information file handling, control, editing, Align Print, Pagesize, Nonstop or Pause Markers or Nomarkers, Find, Drop Keep, and Replace, Clip, Arrange Sort, Match, Compute, Clean, Combine, Load, and Unload are representative of the wide variety of utilities available to handle text generation, table cleanup, comparing of files, etc.

Creation of data and mask files. Data files can be created either with the Editor or with the Unload function. Data files consist of lines which associate values with names. Masks are used by Load and Unload. When used with Load, a mask says where data from a data file should be placed. When used with Unload, a mask says where data is located in a working file, so that it can be converted to a data file.

Data calculations. Through the use of what are referred to as basic equations which are composed of a number of "+", "-", "*", and "/" signs usually followed by an "=" sign, tables of data involving either row equations (horizontal processing of basic equations being involved), column equations (vertical processing of basic equations being involved), or equations involving the Compute function can be processed by T/MAKER II. Through use of up to 23 other symbols in place of the "=" sign, you become able to calculate per cent, square root, absolute value, min/max, log or exponential, reciprocal, etc. of the basic equation. Column equations can be coded to specify to which subsequent rows the equation should be applied. Moreover, control codes for column equations can be selected and range from "always compute" to "suspend compute" as desired for any set of data. Other types of equations available in T/MAKER II include Constant equations and special column equations. Unfortunately, due to the short length of this article, examples of how these various equations are processed cannot be shown here. Subsequent articles would be required to do justice to the "data calculator" and text editing capabilities of T/MAKER II.

ADVANTAGES

T/MAKER II provides a ready-made means for implementing analyses of data together with supporting text material. Report generation with embedded tables of data both before and after analyses and in whatever format you desire can be achieved with a minimum of effort.

BENEFITS

T/MAKER II can be used to sort both numeric and alphanumeric data, ranging in topics from lists of names and salaries to zip codes, whether in ascending or descending order.

Reduction in repetitive data entry activities or periodic updating can be minimized using data files. Creation of mask files can be utilized in mailing list-related activities. And all of these tasks can include calculations that can be tailored to the specific requirements of each activity.

SUMMARY

T/MAKER II employs data calculator and text editor capabilities to achieve a highly flexible and powerful means for table-making.

In future articles, actual examples of how T/MAKER II accomplishes some highly useful data calculations will be presented.

T/MAKER II Tip

Now let me know if you wish to find out the Standard Deviation?

ex		999	999	999	999	999	999
acl		+	+	+	+	=	
ac2		+	+	+	+		avr
	Pupil	Test1	Test2	Test3	Test4	TOTAL	AVERAGE
++	Sue	23	34	45	56	158	40
++	Bob	43	65	76	87	271	68
++	Mary	12	23	34	45	114	29
++	Joe	22	33	44	55	154	39
++	Bill	76	65	54	43	238	60
=	TOTAL	176	220	253	286	935	234
avr	CLASS AVERAGE	35	44	51	57		47

Did you know that you could use "AVR" (average) in both row and column equations?

Two On WordStar

Caveat Installator WordStarus

(Let the WordStar Installer Beware)

James E. Korenthal

Try this: A>ren bleep.com=ws.com
A>bleep

The WordStar menu comes up, right? (If not, skip to the next article.) Now type "R" to "Run a program." You should get: File WS.COM Not Found --
Can't Run a program unless WS.COM is available.

Cute. So what, you say? Well, I got the same message without renaming WS.COM... but WordStar told me that it couldn't find WS2.COM! Wha hoppen?

The problem occurred when I *changed* my installation of WordStar 3.0 to accommodate Lifeboat Associates' CP/M 2.25-A for the TRS-80 Model II. It could just as easily occur on any system where a WordStar (2.0 and above) installation is changed. Here's how:

I started with an already installed WordStar in WS.COM, and then used INSTALL's option "B" to reinstall WS.COM and call the new file (you guessed it) WS2.COM. After verifying that WS2.COM worked properly, I erased WS.COM, and renamed WS2.COM to WS.COM.

Sounds OK, no? NO!!! I was left with a WordStar with an identity crisis! It was called WS.COM, but *internally* thought it was WS2.COM! This doesn't matter for most operations. When you try to run a program from within WordStar, however, the internal name is the one used to return to WordStar's control after the program finishes.

Once you understand the problem, the fix is easy. Use DDT.

```
A>stat ws.com

 Recs  Bytes  Ext Acc
  124    16k    1 R/W A:WS.COM
Bytes Remaining on A: nnnk

A>ddt ws.com
DDT VERS 2.2
NEXT  PC
3F00 0100
-d3e7,3ee     <= dump internal file name (vers.3.0)
03E7 57 53 32 20 20 20 20 20 WS2
-s3e9         <= change "2" (32H) to a space (20H)
03E9 32 20
03EA 20 .
-g0           <= exit ddt and replace modified WS

A>save 62 ws.com  <=62 pps=124/2 recds.(from stat)
```

That's all there is to it. Oh — an apology to you Latin fans for the title. My knowledge of the language extends little past "Mare Nostrum" (means "my horse doesn't play the guitar," no?...)

Jim Korenthal is President of JEKCU, Inc., a new software development firm in New York City. He can be reached at (212) 691-1459, or send correspondence to him in care of Lifelines, 1651 Third Ave., New York, N.Y. 10028.

WordStar Installation For The Hewlett-Packard Computer

Gerry Sawyer

```
**********************************************************************

WordStar installation for the Hewlett Packard computer.
The example has a COPY of the WordStar disk in drive B:

**********************************************************************

B>INSTALL



  COPYRIGHT (C) 1981  MicroPro International Corporation

 INSTALL version 4.2  for MicroPro WordStar release 3.00


Do you want a normal first-time INSTALLation of WordStar?
(Y = yes;  N = display other options): Y


This will INSTALL the WSU.COM on the current drive, save
the result on file WS.COM on the current drive, and then
run the INSTALLed WordStar.  OK (Y/N): Y


  MicroPro WordStar  release 3.00  serial # WS(your s.n.)


      *****  WordStar TERMINAL MENU #1  *****

A  Lear-Siegler ADM-3A          C  Lear-Siegler ADM-31
D  Hazeltine 1500               E  Microterm ACT-IV
F  Beehive 150/Cromemco 3100    G  Imsai VIO
H  Hewlett-Packard 2621 A/P     I  Infoton I-100
J  Processor Tech Sol / VDM     K  Soroc IQ-120/140
L  Perkin-Elmer 550 (Bantam)    2  Terminal Menu #2
3  Terminal Menu #3             Z  none of the above
U  no change

PLEASE ENTER SELECTION (1 LETTER): H

HEWLETT-PACKARD 2621 terminal

OK (Y/N): Y



           *****  PRINTER MENU  *****
(More specific info is displayed after choice is entered)

A  Any "Teletype-like" printer (ie almost any printer)
C  "Teletype-like" printer that can BACKSPACE
D  DIABLO 1610/1620 daisy wheel printer
E  DIABLO 1640/1650/630/Xerox 1700 series daisy wheel printer
F  QUME Sprint 5 daisy wheel printer
G  NEC Spinwriter 5510/5520 thimble printer
I  "Half-Line-Feed" Printers
M  I/O Master / O.E.M. Printer Combination
R  C. Itoh/TEC Starwriter Printer
U  no change
Z  none of the above
```

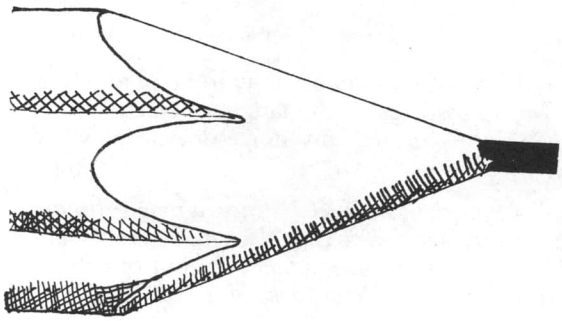
(continued next page)

```
PLEASE ENTER SELECTION (1 LETTER): C

Backspacing TTY-like printer

    This choice is for any printer that will respond to an
Ascii BACKSPACE character (code 08) as well as carriage
return, line feed, and printing characters.
    Make sure any AUTO LF or LOCAL LF switch is OFF.
    If you have a DAISY WHEEL or thimble printer shown on the
menu, use the appropriate choice in order to obtain "Micro-
justified" output and additional formatting capabilities.
    Refer to manual for instructions on patching in optional
control sequences for ribbon color change, character pitch
change, and half-line roll (for subscripts and superscripts)
if your printer has these capabilities.

OK (Y/N): Y



Most Teletype-like printers use no communications protocol.

       *****  COMMUNICATIONS PROTOCOL MENU  *****

A "Communications Protocol" is necessary with some printers
to prevent printer buffer overflow and character loss.

E  "ETX/ACK" Protocol
X  "X-ON/X-OFF" Protocol
N  NONE required (or handled outside of WordStar)
U  no change

PLEASE ENTER SELECTION (E, X, N, B, or U): N

No communications protocol


OK (Y/N): Y



With no protocol, the usual driver selection (below) is L

           *****  DRIVER MENU  *****

Or, how should WordStar send characters to your printer?

L   CP/M "List" device (LST:)
T   CP/M primary console device (TTY:)
C   CP/M secondary console device (CRT:)
P   Port Driver (direct I/O to 8-bit ports)
N   Parallel Centronics Printer Driver
Q   Serial Driver on TRS-80 Model-2
S   User-installed driver subroutines
U   no change

PLEASE ENTER SELECTION (L,T,C,P,N,P,S,B or U): L

CP/M List Output driver (LST:)


    In most systems this is a "logical" device which must be
assigned to the desired one of four "physical" devices with
the STAT command, before WordStar is invoked.

OK (Y/N): Y



ARE THE MODIFICATIONS TO WORDSTAR NOW COMPLETE?

  IF THEY ARE ANSWER YES TO THE NEXT QUESTION.
  IF YOU WISH TO MAKE ADDITIONAL PATCHES TO WORDSTAR'S
  USER AREAS, ANSWER NO TO THE NEXT QUESTION.


OK (Y/N): N


YOU MAY NOW MODIFY ANY LOCATION DESCRIBED IN THE LISTING
AT THE END OF THE USER MANUAL OR THE CUSTOMIZATION NOTES.


YOU MAY USE EITHER THE LABEL OR THE HEX ADDRESS TO SPECIFY
THE LOCATIONS YOU WISH TO CHANGE.  IF YOU USE A LABEL THEN
YOU MAY APPEND AN OFFSET TO THE LABEL (I.E. LABEL:+31). THE
LABEL ALWAYS HAS A ":" APPENDED (LABEL:). YOU MAY SPECIFY
THE NEW VALUE ONLY AS A HEX NUMBER.  A LOCATION OF ZERO (0)
WILL CAUSE THE END OF THE MODIFICATIONS


LOCATION TO BE CHANGED (0=END): 284
    ADDRESS : 0284H   OLD VALUE: 04H   NEW VALUE: 00
LOCATION TO BE CHANGED (0=END): 28B
    ADDRESS : 028BH   OLD VALUE: 04H   NEW VALUE: 00
LOCATION TO BE CHANGED (0=END): 2D1
    ADDRESS : 02D1H   OLD VALUE: 19H   NEW VALUE: 03
LOCATION TO BE CHANGED (0=END): 2D2
    ADDRESS : 02D2H   OLD VALUE: 40H   NEW VALUE: 07
LOCATION TO BE CHANGED (0=END): 2D3
    ADDRESS : 02D3H   OLD VALUE: 09H   NEW VALUE: 03
LOCATION TO BE CHANGED (0=END): 0


CONFIRM TERMINAL AND PRINTER SELECTIONS:

 HEWLETT-PACKARD 2621 terminal
 Backspacing TTY-like printer
 No communications protocol
 CP/M List Output driver (LST:)

OK (Y/N): Y

**********************************************************************

The installation is now complete and the WS.COM file will
be written to the logged in disk.

For users that wish to play with the highlighting, patch
the above locations with the following data instead of the 00's
See page number 9 of appendix E from the 3.0 manual.

IVON: 0284H - 04,1B,26,64,48

IVOFF: 028B - 04,1B,26,64 40
```

A Call For Manuscripts

Do you have thoughts to share on *your* experiences with microcomputer software? We're interested in hearing what you have learned, and so are other readers. Whether you've used professional packages, programming tools, word processing packages, or any *serious* CP/M-80 compatible products, we'd like you to write for us. We like to publish both long essays and those short gems which can hold so much important information.

Send us a brief resume of your software experience, and samples of your previous writing, if you have any. (Don't be shy if you're *not* an experienced writer.) Then we can talk about your work and about payment for your efforts. Write or call: Editorial Dept., Lifelines Publishing Corp., 1651 Third Ave., New York, N.Y. 10028. Telephone: (212) 722-1700.



Back Issues

Lifelines begins all new subscriptions with the upcoming issue. However, you can order back issues, as available, at the single copy price. Here is a list of the back issues and their feature articles. (Of course all issues contain our regular features: New Products, New Versions, Bugs, Tips and Techniques.) If you wish, you may use this page as an order form and check the available back issues which you would like. Or give us a call at (212) 722-1700. All orders must be pre-paid, either by check, VISA, or MasterCard. Checks must be in U.S. dollars, drawn on a U.S. bank, and made payable to Lifelines Publishing Corporation. The price for issues sent to the U.S., Canada, or Mexico is $2.50 per back issue. For copies sent to all other countries the price is $3.60. Some copies are available in Xerox copies only. These are slightly higher: single copies are $3.10 and $4.25 for foreign orders. These issues are denoted in italics.

VOLUME I

☐ *JUNE 1980 (Vol. I, #1):*
BASIC Comparisons-CBASIC by Bill Burton
The Undocumented Z80 Opcodes by Robert Halsall
DU Tutorial by Ward Christensen

☐ *JULY 1980 (Vol. I, #2):*
Special features on the CP/M Users Group, including catalogs and abstracts
BASIC Comparisons-BASIC-80 by Bill Burton

☐ *AUGUST 1980 (Vol. I, #3):*
BASIC Comparisons-BASIC- 80, part 2 by Bill Burton
Assembly Language Development Systems by Ward Christensen

☐ *SEPTEMBER 1980 (Vol. I, #4):*
BASIC Comparisons- BASIC-80 Compiler, Rev. 5.2 by Bill Burton
BSTAM-A File Transport Utility by Michael Posehn
The Software Evaluation Group by Steve Patchen
Three Pascals Are Better Than One by Michael Posehn

☐ *OCTOBER 1980 (Vol. I, #5):*
Assembly Language Development Systems, part 2 by Ward Christensen
How To Use a Data Management System by Steve Patchen
A CP/M Patch for Altair and iCOM Systems
The Software Evaluation Group Review Format by Ed Paulette and Steve Patchen

☐ *NOVEMBER 1980 (Vol I, #6):*
A Review of BSTMS from the Publisher
Printing sans LPRINT by Bill Norris
BASIC Comparisons-XYBASIC by Bill Burton
Assembly Language Development Systems part 3 by Ward Christensen
Introduction to Data Management Systems by Timothy Berla and John Lehman

☐ *DECEMBER 1980 (Vol I, #7):*
Business Application Problem Definitions by Steve Patchen and The Software Evaluation Group
Assembly Language Development Systems-SID by Ward Christensen
The CP/M Users Group Volume 46-Catalogue and Abstracts

☐ *JANUARY 1981 (Vol I, #8):*
Assembly Language Development Systems-SID Part 2 by Ward Christensen
A Patch for muMATH Version 2.02
The CP/M Users Group Volumes 44 and 45-Catalogues and Abstracts

☐ FEBRUARY 1981 (Vol I, #9):
A Review of VisiCalc and T/Maker by Steve Patchen
The Osborne Packages-Accounts Payable and Accounts Receivable by Martin McNiff
Assembly Language Development Systems-MACRO-80 and LINK-80 by Ward Christensen
The CP/M Users Group Volume 47- Catalogue and Abstracts

☐ MARCH 1981 (Vol. I, #10):
The Configurable Business System by Ed Paulette
The Osborne Packages-General Ledger by Martin McNiff
BASIC Comparisons-SBASIC part 1 by Bill Burton
The CP/M Users Group Volume 48- Catalogue and Abstracts

☐ APRIL 1981 (Vol. I, #11):
Condor by Ed Paulette and Steve Patchen
BASIC Comparisons-SBASIC part 2 by Bill Burton
A Review of PMATE by Harris Landgarten

☐ MAY 1981 (Vol. I, #12):
A Brief Review of PASM, BUG/uBUG, PLINK, and EDIT by Tom Cochran
Comments on SSSFORTRAN by Trevor Marshall
BASIC Comparisons-SBASIC Version 5.3h Part 3 by Bill Burton
The CP/M Users Group Volume 49 - Catalogue and Abstracts

VOLUME 2

☐ JUNE 1981 (Vol. 2, #1):
A Review of PLINK II by Harris Landgarten
The Software Evaluation Group: SELECTOR IV by Tim Berla and Steve Patchen
The Osborne Packages: Payroll by Martin McNiff
The CP/M Users Group Volume 50 - Catalogue and Abstracts

(continued next page)

☐ JUNE 1981 (Vol. 2, #1):
A Review of PLINK II by Harris Landgarten
The Software Evaluation Group: SELECTOR IV by Tim Berla and Steve Patchen
The Osborne Packages: Payroll by Martin McNiff
The CP/M Users Group Volume 50 - Catalogue and Abstracts

☐ JULY 1981 (Vol. 2, #2)
MDBS, Part 1 by Harris Landgarten
The CPM USERS Group Volume 51 - Catalogue and Abstracts
A Tutorial on Volume 51 by Ward Christensen

☐ AUGUST 1981 (Vol.2 #3):
Large Memory Management Comes to the Z80 by Bernard R. Wess, Jr.
CP/M Bit Map File Allocation by Kelly Smith
Ashton-Tate's dBASE II by Steve Patchen and Ed Paulette
Random Numbers for Microsoft BASIC by James R. Reinders
The CP/M Users Group Volume 52 - Catalogue and Abstracts

☐ SEPTEMBER 1981 (Vol. 2, #4)
Bringing Up UCSD Pascal Version 1.5 or 1.0 From Scratch by Kelly Smith
MDBS Part 2-QRS by Harris Landgarten
Terminal Talk by Steve Patchen
Calling SORT As A Subroutine from MBASIC SORT
8080 Programming Tutorial - Introduction and Terminology - Part 1 by Ward Christensen
Assembly Language Interface to PL/I-80 by Michael J. Karas

☐ OCTOBER 1981 (Vol. 2, #5):
8080 Programming Tutorial- Terminology by Ward Christensen
Assembly Language Interface to PL1-80 Part 2 by Michael J. Karas
ABBS, CBBS, FBBS, RBBS, ETC, Maybe you'd like to start one too? by Jim Mills
The Dentist's Office: PAS-3 and Univair by Tom Crites
Better Random Numbers by Bill Burton
The CPMUG Volumes 53 and 54 - Catalogues and Abstracts

☐ NOVEMBER 1981 (Vol. 2, #1):
8080 Assembly Language Tutorial — Park 3: Terminology and Architecture by Ward Christensen
Custom I/O Routines for PIP by Michael Karas
After the Game by Stephen Walton
A Review of Pascal MT+ by James Gagne

☐ DECEMBER 1981 (Vol. 2, #7):
8080 Assembly Language Tutorial — Data Movement & Arithmetic Instructions by Ward Christensen
A Subroutine to Check for Stack Overflow by Kelly Smith
After the Game - Part 2 by Stephen Walton
WordStar Release 3 Installation for TRS-80 Model II by James K. Korenthal
Indexing Utilities: FABS and MAGSAM by Steve Patchen
Remote System Do's and Don'ts by Dave Hardy
The Osborne I Computer by Kelly Smith
The CPMUG Volumes 65-75 - Catalogues and Abstracts

Tips & Techniques

J. N. Baraclough of Scottish Television Limited has been kind enough to share patches for ZDT which improve output formatting on 64-column screens.

- To change the number of bytes per line on the D command from 16 to 8:-

  CHANGE LOCATIONS 05C6H, 05D2H, 05EAH
  FROM 10H TO 08H

- To reduce the number of address blocks per line on E and F commands from 13 to 8:-

  CHANGE LOCATION 0561H
  FROM 0CH to 07H

- To display only the normal register set on the X command:

  CHANGE LOCATION 1005H thru 1007H
  TO 0DH, 0AH, 24H.

  CHANGE LOCATIONS 0FIBH and 0F1CH
  FROM 08H, D9H to 18H, 27H

(The only disadvantage of the third patch is that the alternate register set cannot be examined.)

Roberto Denis of Plantation, FL saw our note on problems with BASCOM and CDOS. He had this to say:

"The problem with BASCOM (Version 5.2) not operating properly under CDOS lies in BASCOM, not in CDOS. Updated versions of CDOS do have a proper system call 12, which returns the HL register containing 0(hex) to flag the fact that it is not CP/M 2.2 compatible. However, an *undocumented* feature of CP/M is that it also returns in the A register whatever is returned in the L register (and in B, whatever is in H), BASCOM tests the A register for the version instead of the L as is documented by Digital Research. CDOS returns in the A register whatever was sent to it (i.e. register contents are preserved).

It seems that Microsoft could fix this problem by simply adding the instruction MOV A,L after the system call is made.

In the meantime, users of CDOS (Versions 1.7 and above)

can use BASCOM by effecting the following patch to compiled programs:

1) After compiling and linking, find the following sequence in the .COM file (first read into memory with some Debugger):

21 22 21 (LXI H, 2122)

(21,22 hex are the read/write random system calls.) Replace with:

21 14 13 (LXI H, 1314)

(13, 14 hex are the read/write sequential system calls.)

2) Save the .COM file — it now works!

If you don't have a debugger that can do string searches, follow the code around with DDT's disassembler, starting at 100 hex; it should be no farther in than 100 or so into the code since it is part of the initialization routine."

CPMUG Volumes 76 and 77: Catalogues and Abstracts

CPMUG Volume 76

Miscellaneous Pascal Z utilities. Original material from Pascal Z Users Group Volume 6.

-CATALOG.076		CONTENTS OF CPMUG VOLUME 76
-CATALOG.ACK		Acknowledgement file
ABSTRACT.076		Descriptive contents of volume 76
SIG/M.LIB		SUBMITTAL FORM
UGFORM.LIB		SUBMITTAL FORM

VOL.#	NAME	SIZE	COMMENTS
76.1	COMPLEX.LIB	5K	Complex number utility
76.2	CURSOR.LIB	2K	Cursor control for SD Sales video
76.3	CURSOR.PAS	3K	
76.4	CURSOR.COM	7K	
76.5	FINDBAD.MAC	7K	Locate bad sectors under CP/M 2.X
76.6	FINDBAD.COM	2K	
76.7	LONGLINE.PAS	2K	Concatenation demo
76.8	LONGLINE.COM	7K	
76.9	NAD4.PAS	9K	Name address data entry
76.1	NAD4.COM	9K	
76.1	OTHELLO.PAS	6K	Othello - UCSD version
76.1	OTHELL1.PAS	8K	
76.1	OTHELL2.PAS	8K	
76.1	OTHELLIN.PAS	8K	
76.1	PTABLE.PAS	1K	Table generating demo
76.1	PTABLE.COM	8K	
76.1	PTRS2.PAS	4K	Pointer demo program
76.1	PTRS2.COM	5K	
76.1	QQSORT.PAS	4K	Pascal - Quicker Sort
76.2	QQSORT.LIB	3K	
76.2	QQSORT.COM	8K	
76.2	QSORT.COM	7K	Pascal - Sort
76.2	QSORT.PAS	3K	
76.2	QSORT.LIB	1K	
76.2	RCPDAT.XXX	2K	Recipe program
76.2	RCPDAT.YYY	1K	
76.2	RCPDAT.MST	1K	
76.2	RECIPE.COM	15K	
76.2	RECIPE.PAS	20K	
76.3	SHELL.PAS	3K	Pascal Shell sort
76.3	SHELL.COM	7K	
76.3	SHELL.LIB	1K	
76.3	SNOOPY81.CAL	5K	1981 calendar
76.3	XREF.PAS	10K	Cross reference program
76.3	XREF.COM	14K	

(continued next page)

76.3	ZCOMPAR.PAS	2K	Pascal file compare utility
76.3	ZCOMPAR.COM	7K	
76.3	F1.DAT	1K	
76.3	F2.DAT	1K	

CPMUG Volume 76

Ray Penley sent me another disk full of goodies. I sure wish I could program like he can; in fact, I would be happy if I could type in as many programs as he has. Anyway, there are lots of usable programs on this disk. The sort routines alone make it worthwhile.

These sort routines are set up so that they can be part of a library and yet be demonstrated by a COM file. So the PAS file is the source of the COM file whereas the LIB file is what you can use to pull in with your editor at the proper spot. These three sort programs are compared with each other to give you an idea of their speed. (Look in the front of the PAS file for that info.)

SHELL.COM/PAS/LIB–A Pascal Shell sort routine.

QSORT.COM, .PAS, .LIB–A Pascal Quick sort routine.

QQSORT.COM, .PAS, .LIB–A Pascal Quicker sort routine.

RECIPE.COM, .PAS, RCPDAT.MST, RCPDAT.XXX, RCPDAT.YYY–Latest version of the recipe program, many changes plus improvements for Version 3.2, you will notice it has a much smaller COM file.

NAD4.COM, .PAS, COMPUTER.NAD–Ray is proud of this one, it is from scratch. It is a NAME ADDRESS data entry program.

PTRS2.COM, .PAS–Excellent updated demo for pointers. It has some interesting ideas.
I couldn't let Ray do it all so I threw in this little extract.

LONGLINE.COM, .PAS–I extracted this out of Ithaca's manual. It is a demo on words being added together into lines.

PTABLE.COM, .PAS–I modified this one for a table generating demo. Generating a table by algorithms is quicker than looking up a table.

FINDBAD.COM, .MAC–The original program came from a mag but this has been heavily modified to run on CPM 2.2 and to locate the bad sectors completely out of your way. Everyone needs this one. An excellent utility.

ZCOMPAR.COM, .PAS, F1.DAT, F2.DAT–Bob Harsch came through again for us. He was trying to compare some programs but the first error would kill the compare. So he sat down and wrote a Pascal/Z compare that compared ALL the differences, HEX,DEC,and CHAR. To run, change the two programs to compare to F1.DAT AND F2.DAT and then type ZCOMPAR. I added these DAT files as a demo, I injected one error into the second copy. So all you have to do is type ZCOMPAR to see it work.

CURSOR.COM, .PAS, .LIB–I have a SD SALES Video board and wanted to use the XY cursor positioning feature. But when I tried to write it in assembly I run into some problems. So I said why not Pascal/Z. This is a DEMO to test the XYcontrols and the LIB is the subroutine to put in your library. It works.

XREF.COM, .PAS–Ithaca Intersystems donated this to our group. It is rough but look it over.

SNOOPY81.CAL–I thought it would be nice to have a 1981 calendar. So I updated Snoopy.

OTHELLO.PAS, OTHELL1.PAS, OTHELL2.PAS, OTHELLIN.PAS-This is a project for someone interested in advanced Pascal Programming. It is an excellent game with a superior algorithm. But, it is written in UCSD. So we need it converted.

CPMUG Volume 77

Database seed system
Multi-track system BIOS
Fixed length disk sort system
Print format program
Original material from Pascal Z Users Group volume 7

-CATALOG.077	Contents of CPMUG Volume 77
-CATALOG.ACK	Acknowledgement file

ABSTRACT.077		Comments on CPMUG Volume 77
CRCKFILE.077		CRC of CPMUG Volume 77
SIG/M.LIB		SUBMITTAL FORM

VOL.#	NAME	SIZE	COMMENTS
77.1	DATABASE.DOC	32K	Database seed program by Dr. Bowles
77.2	DBBUILDE.CPM	22K	
77.3	DBDEMO.CPM	1K	
77.4	DBUNIT.1	8K	
77.5	DBUNIT.2	19K	
77.6	DBUNIT.3	22K	
77.7	DBUNIT.4	17K	
77.8	DBUNIT.CPM	1K	
77.9	SGEN.COM	2K	BIOS for greater than 2 system tracks
77.10	SGEN.MAC	10K	
77.11	MACROZ.MAC	18K	
77.12	CLIB.MAC	9K	
77.13	SGEN.DOC	5K	
77.14	BSORT.ASM	40K	Fixed length disk sort program
77.15	BSORT.COM	4K	
77.16	BSORT.NOT	4K	
77.17	PGLST.PAS	3K	Print format program
77.18	PGLST.COM	8K	
77.19	PGLST.NOT	1K	
77.20	W3UA.LIB	6K	

CPMUG Volume 77

This volume developed fast and slow. The material came in fast but I had the newsletter to get out so it was slow. However, you will see some new names this time and some original ideas. We have a excellent start with our disks and it is easier to program with such help. But don't slow down. Keep sending those programs. If we keep it up, the day will come when most any module we need will be available FREE. Then programming will be just a logical excercise and not a coding drag.

This first group is Dr. Bowles' database seed program that he used in his classes. It is a frame to which a expanded data base can be built. I have a special interest in databases and I am glad that Dr. Bowles donated this to Public Domain. The one who gets this converted to Pascal/Z will win a free disk full of software and the gratitude of everyone else.

DATABASE	DOC	DBBUILDE	CPM	DBDEMO	CPM
DBUNIT	1	DBUNIT	2	DBUNIT	3
DBUNIT	4	DBUNIT	CPM		

Donald Killen, of Carrollton, Texas, found that as he played around with his BIOS, he ran out of track space. So he reworked things a bit. Now he has as much space as he wants. That means, of course, a different SYSGEN. So this group of programs are all about how to do that.

SGEN	COM	SGEN	MAC	MACROZ	MAC
CLIB	MAC	SGEN	DOC		

Gene Duncan, of Palo Alto, Ca, likes to do original work and has donated the following programs. His address is listed in Newsletter #5 and he would appreciate any feedback that you might want to pass on to him. It'll give him an idea of how he is doing. The BSORT group is a sort program that is fast and can't be fooled. He gives some background in the program so be sure to check that program. The PGLST group is a print format program that will save paper. It reads a text file of fixed or variable length character records, each record ending with a CR and LF, and arranges them on a print page in columns. I added the data file W3UA to show you how it works, nice utility.

BSORT	ASM	BSORT	COM	BSORT	NOT
PGLST	PAS	PGLST	COM	PGLST	NOT
W3UA	LIB				

CPMUG Errata

Ward Christensen has sent in the following information regarding Volume 46.

FILE: RETDL.COM

SYMPTOM: Disassembles "DAA" as "RLC".

FIX: Patch the 07H at address 1342, to 27H.

EXAMPLE: (Key entry in **bold**)

```
B>ddt retdl.com
DDT VERS 1.3
NEXT  PC
1B80 0100
-s1342
1342 07 27
1343 00 .
-^C
A>save 27 b:retdl.com
```

Ordering From CPMUG

Many of you have inquired about ordering volumes from The CP/M Users Group.

The complete catalog of volumes is available for $6, prepaid, to the U.S., Canada and Mexico. The price is $11 for catalogs sent to all other countries.

Software is obtainable exclusively on diskette; CPMUG does not supply printed documentation. Prepaid media and handling charges are as follows:

8" IBM format to U.S., Canada, Mexico	$8
8" IBM format to all other countries	$12
North Star format to U.S., Canada, Mexico	$8 or $12
North Star format to all other countries	$12 or $16

As you will note, software is available in 8" IBM and North Star 5¼" formats. Write for a price list of North Star format volumes, as prices vary for volumes in this format.

Payment covers the cost of the diskette(s), packaging, and shipping. Checks must be in U.S. dollars, drawn on a U.S. bank. Domestic shipping is via UPS where a full street address is given; all other orders are via U.S. Postal Service.

CPMUG receives orders by mail only; they have no phone service. Orders should be sent, with prepayment, to CPMUG, 1651 Third Ave., New York, N.Y. 10028.

Members receiving the material are reminded that software contributions are necessary if the exhange program is to prosper. Software contributions are gladly received for inclusion into the Library with the understanding that the contributor is authorized to make the material available to others for their individual non-commercial use.

Software should be accompanied by sufficient documentation in the form of internal comments or accompanying *.DOC file to permit the material to be applied and/or modified. Where appropriate, the documentation should describe any supporting software (interpreter, memory, clock, etc.) necessary to use the routine. For your convenience, a comprehensive submittal form is now included on most distributed diskettes.

Contributors are invited to request any any Library Volume in exchange for the one submitted.

PLAN80, Version 2.1— Hints & Kicks

Bill Norris

November 15, 1981

INSTALL has a default terminal defined (Soroc 120) which is useful if the terminal definition files are lost or damaged. If your terminal is one that PLAN80 has a TRM file for, it is easier to read it with the R command than it is to edit the Soroc values.

CONTROL.TRM is the *only* terminal definition file that is read by INSTALL or PLAN80, so if fortune smiles upon your CPU, there will be no need to run INSTALL. All that is necessary is to rename the TRM file to CONTROL.TRM. Complications arise, however. There already is a CONTROL.TRM file on the disk. If you read the manual and were wondering what happened to the missing Hazeltine definition file, wonder no more. Even if it is the proper terminal definition file, it should be renamed by using CP/M (on a copy, please) like so:

B»REN HAZ1500.TRM=CONTROL.TRM[VO] «c.r.»

Now PIP should be used to create a CONTROL.TRM file that is a copy of the one that is needed. If your terminal happens to be a Lear Siegler ADM-3A for instance, the appropriate CP/M command might be the following:

B»A:PIP CONTROL.TRM=ADM3A.TRM[VO] «c.r.»

WARNING: are you using a real terminal? If so, the above should work. If you are emulating a terminal in software, (this is the case if you are using the TRS-80's video board, an S-100 video board, etc.) there will be a problem if the emulating software doesn't include an erase to end of line function. This is because PLAN80 will erase the line by writing spaces to the end of the line, and if this is done on the bottom line of the screen, when the last column is written to, the screen will usually scroll up one line and PLAN80 won't realize that this has happened. If you have a TRS-80 with Lifeboat's CP/M you should do the following:

If your CP/M is version 2.25B or later, just install the ADM31 terminal definition file. If it is an earlier version, install the ADM3A file, and edit it by defining the EOL function. The value that erases to the end of line is 23.

If your emulator can't do this, there are two temporary fixes. When PLAN80 messes up the screen, just redraw it with the P command. This can be annoying because it is slow (it has to redraw the screen twice). A faster method is

to use INSTALL to define a dummy EOL sequence. Up to 5 characters may be used, and until you are familiar with what is happening, it is best to use them all to print a warning on the screen. If you use the values 33 101 111 108 33, the screen will display !eol!. As this will shift everything on that line 5 characters, you will probably want to change it to one character eventually.

INSTALL supposedly allows you to redefine the screen height/width from the standard 24x80. If it worked, the above fixes would not be needed. All you would have to do is redefine the screen to 23x80 or 24x79. Unfortunately, while INSTALL works with the new values (it looks fine with the T command), PLAN80 still will scroll the screen up. Also, after testing the new row and column values, any odd values selected are reduced by one.

PLAN80 lets you temporarily redefine the screen to new height and width values, but this won't solve the problem. (the command line still get placed on line 24 and spaces are written out to column 80).

NULLS are not needed for most video boards and may be deleted from your CONTROL.TRM file, but if left in the file, will only slow up the display by a small amount (probably unnoticeable).

Macros Of The Month

Michael Olfe

Ward Christensen and Tom Cochran have expressed interest in establishing contact with other people using PMATE. If you share this interest, please write to me c/o Lifelines. And if you don't want this to be the last "Macros of the Month", **send in those macros.**

The sorted directory macro in the manual is torturously slow, and the built-in directory listing is hard to read. This one lists directory entries of one filetype in a four-across format, does not alphabetize, and is fairly fast. It puts the filenames in the main text buffer, and deletes them when you press a key.

```
; Uses:          Buffer 0,"T", Variable 0
; Invoke:        If macro is in buffer 1,
;                to list all files with type "COM", type ".1com$^["
;
t 15ye lqa                ; tag the location, set new tabs,
                          ;  get ftype from caller
bk be i^AA$ a i*.$        ; clear buff 0, insert ambiguous filename
bte$ x1^A@0$ @cv0 #v1 t   ; insert filenames in buffer "T",
                          ;  count insertions
[es
$@e!(@c>@0)_-m            ; make format four-column
@x>50{m^}                 ;
r       $]                ;
gpress key$#@1d8ye%       ; wait for keypress, delete directory,
                          ;  and restore tabs
```

Put the cursor anywhere on a label name, invoke the macro, and it will position the cursor at the label. I use this for "tracing" jumps and calls in assembly language programs, but it could be turned into a generalized searcher by not appending a colon to the word in buffer 0. Moving the cursor with the left hand and hitting the escape key with the right (while the macro is still on the command line) is a fast way of following jumps and calls through a source file.

```
; Uses:          Buffers 0,"T"
; Invoke:        If macro is in buffer 1, and cursor is on "BEGIN"
;                in line reading "JRZ BEGIN", type ".1". Macro
;                will place cursor on label "BEGIN:" if it is
;                anywhere in the file.
;
-s^W^[mt                  ; find beginning of word and tag
s^W^[-m#bc          ; find end and move word to buffer 0
bei:$                ; put a colon after word to indicate label
btees^A@0$@e=0%      ; set error supress flag,
                     ; search forwards for the label in buffer 0
                     ; stop if found, else
-s^A@0$              ; search backwards for it
```

PMATE is useful for batch. It can run through a file and translate one kind of mneumonics to another, or one dialect of Pascal to another. This macro automates search and replace. You load a buffer with a list of search/replace words or phrases. When the macro is invoked it takes search-replace pairs from that buffer and does the replacements in the text, pair by pair.

(continued next page)

The search-replace buffer can be saved in a file and used repeatedly.

```
; String-getting macro (from PMATE manual) in buffer 7
;  gets an escape-terminated string into buffer 9
;
b9k
[bte              ; do good, avoid evil buffer errors
g^A@9$@k=27_
b9e
@k=127[-d][@ki]]
b9k


;       Constructs buffer 0 search and replace strings
;         Use the "Break" instant command key to exit
;
; Uses:          Buffer 0,7
; Invoke:        If macro is in buffer 1,
;                 type ".1" and alternately type
;                 search-phrases and replace-phrases,
;                 using "ESC" to end each phrase, break to exit
;
bkbe              ; clear buffers, insert header
iSearch strings                  Replace strings
$
[.7be             ; get search string into buffer 9
  i~$b9gi~$       ; move to buffer 0, delimit
  .7be
  i     ~$b9g     ; get replace string
i~
$]gDone$

;  The "~" symbol is used in this macro for string quotation due
;   to a quirk of the "W" command but more importantly to allow
;   multiple-line search and replace strings and a nearly-free
;    format in the buffer.
;
; Uses:          buffer 0,"T",8,9
; Invoke:        If macro is in buffer 2,
;                 buffer 0 is filled with search-replace pairs,
;                 and buffer "T" has text, type ".2"
bea bte                    ; begin at the beginning
[be es~$@e_ts~$-m#b8cm     ; search-phrase in buffer 8
 s~$ts~$-m#b9cm            ; replacement in buffer 9
 btea                      ; start search at beginning of "t" buffer
 [ec^A@8$@9$@e_]      ; replace all
]
btegDone: Press key$
```

One thing PMATE lacks is "split-screen" capability. This macro gives a one-line window onto another buffer and thus allows you to scroll through two files line-by line. The cursor points to the current line in the "T" buffer, and the current line in buffer 9 is displayed at the top of the screen. A carriage return scrolls both buffers one line.

```
;       File compare macro
;
; Uses:         Buffers 0,"T", and 9          Value register 0
;Invoke:        If you have two files to compare, and one is in
;                the "T" buffer and the other is in a disk file
```

```
;                named "file2.asm", and this macro is in buffer
;                  3, type ".3file2.asm".
;
;
qab9kbk                    ; get filename from caller
                           ; clear buff 9 and 0
b9exi^Aa$a                 ; put file in buffer 9
btea                       ; got to top of buffer 9 and "T" buffer
0v0                        ; current line number in buffer 9
[b9ea @01 bd bte           ; move next line from buffer 9 to buff 0
                           ; goto buffer "T"
g^A@0$ bk                  ; display buff 0 line on top
@k=69{-1va0}               ;  "E" : back one line in buffer 9
@k=88{va0}                 ;  "X" : forward one line in buffer 9
@k=82{-21va0}              ;  "R" : back 21 lines (one screen)
@k=67{21va0}               ;  "C" : forward 21 lines (one screen)
@k=13{1va0}]               ;  <CR> moves both buffers forward one
                           ;    line
```

Miscellaneous:

0)None of these macros use the disk-scrolling capabilities of PMATE version 3. They act only upon the text in memory.

1)Version 3 has a @b variable which contains the current buffer number. This solves only half of a problem. The macro cannot use that information to choose an active buffer, since variables in buffer statements cause errors, e.g. b@0e.

2)Why does the "W" command treat tabs as characters rather than as white-space? This does seem odd.

3)There is no warning in the manual that if-then-else statements may not be broken by carriage returns. The logic goes haywire if the "then" and "else" clauses are not on the same line.

(continued from page 2)
Saville, who had most graciously provided among other things an open invitation to contact him for assistance. Without a microsecond's hesitation I picked up the phone; a few moments later I found myself engaged in a lively discussion with Wink, who took me through a step-by-step introduction to STOIC.

Today STOIC is not in widespread use, though thought by some to be far superior to FORTH. Had MIT and Harvard seen fit to place it in the public domain without qualification, it might well have enjoyed considerable commercial success, as FORTH recently has.

The key to longevity and widespread use of software is the establishment of official standards and unencumbered as well as unimpassioned disclosure of the basic architecture and definitions upon which it is based. Those who, to maintain commercial advantages, treat fundamental software (such as languages and operating systems) as totally proprietary and subject it to frequent revision, have the potential for more harm than good and do much to impede the development of our industry as a whole.

In summary, it is in our own best interests to support any and all efforts to develop and maintain true standards in the microcomputer industry; this task must be met with enthusiasm and active participation by each of us if we are to realize the full potential of the microcomputer. Be wary of those who claim to protect our industry by covertly developing software "standards". It's not by keeping us in the dark that they best contribute to our industry, but rather by development of and adherence to industry standards in the full light of day. We don't want *de facto* but rather *de jure* standards.

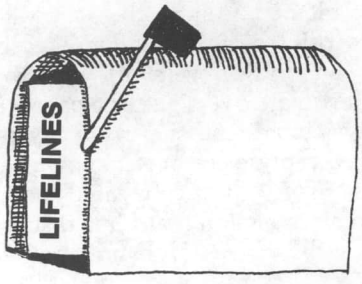
Let us encourage those members of our industry who are reluctant to participate openly and enthusiastically in any and all efforts to develop standards. Let us urge them to take a broader view and act in the collective interests of this industry. IBM has opened the door to a new era and the time to act is at hand ...

Edward H. Currie

Renew

Did your subscription begin in February 1981? If it did, you've probably received two reminders from us and you should know that renewal time is here. If you don't renew now, you'll almost surely miss our big February issue, which will include some exciting reviews and product information.

Don't miss out on your sole source of information on what's really happening with CP/M-80 software. Call or write the *Lifelines* Subscription Dept., 1651 Third Ave., New York, N.Y. 10028. Telephone: (212) 722-1700.



Zoso

Hello and Happy New Year.

Speaking for myself, I find it most depressing to face a brand new year, grumbling about all the money which I spent on seasonal gifts. As if that isn't bad enough, the cruelest part of a Northeastern winter is almost upon us. Wait, it gets worse; I am beset by some virulent strain of imported flu, aggravated by a hangover so brutal that for years to come I will turn lime green at the mere remembrance. I guess the main thing which keeps me going is knowing that 1981 has at last and forever become a droplet in the fathomless oceans of time...

I confess; you caught me. Except for the part about winter's gathering gloom, I made it all up. Naturally, if this magazine appears in your mailbox around the first of January, this must have been written some time earlier (December 3rd to be precise). I should have known that you're far too alert to be fooled by such a pitiful ploy and I should never have tried. However, be warned that others may be giving you less credit than I do. I'll explain shortly.

You may recall that a few month's ago, I waxed caustically about Disk Doctor (since greatly improved) and that the author of that program, John Holland, let me have it back with both barrels in a subsequent issue. Anyway, I wrote a lengthy and detailed reply to Mr. Holland. It had been my intention to have [all of] that reply included in the last column. However, it was decided that the best interests of this publication would be served by putting a lid on the old topics and proceeding with the new ones, so only the last bit of my letter to Mr. Holland was printed. (Mr. Holland received the unabridged version). Due to an unfortunate oversight, the pages (and it was pages) missing from my published reply were not acknowledged. This may well have created the mistaken impression that I had dismissed the bulk of Mr. Holland's rejoinder without comment, while offering only cavalier reply to points of my own choosing. Such was not the case and I want everyone to know.

This month's main attraction concerns my impromptu visit to COMDEX (the show for computer dealers and OEMs which was held in Las Vegas, November 19-23). Since I am neither a dealer nor an OEM, I really had no business attending but what the heck, I was about due for a visit to Vegas.

Regarding this desert paradise, let's understand this much from the outset; anyone endowed with even a scintilla of taste or discernment would have to agree that Las Vegas is an ideal way to experience 'Apocalypse Now'. If you were to remove the bright lights and the one-sided wagering which paid for them all, you would be left with a singularly mean hick town, distinguished only by garish architecture and a trend-setting crime rate. But what's the use of pretending? Vegas does have the bright lights and it does have acres and acres of gambling. Do you like the kind of crowds one finds in a place like this? I don't! Still, for all its failings, there is one thing about Vegas which never fails to warm my heart. It's one of the last bastions of the closely knit American family. Where else can one still see so many concerned fathers escorting their pristine daughters back to the hotel rooms?

The evening I arrived, I went to scout up some familiar faces at the MGM Grand Hotel. I had no luck (not the best of omens) so I sat down for a few hands of 21 (as the insiders call it). Within moments, a rather weathered looking cocktail waitress came by to take drink orders. I asked her for a Dewar's (my favorite Scotch whisky, better known as White Label most places outside the United States). When I sipped from the glass which she brought me I almost perished -YUCCH! I politely informed the waitress that she must have made a mistake. My guess was that she had served my Dewar's to the person who had ordered nail polish remover. Off she sped to make amends. A few minutes later she returned. You guessed it! She brought me more of the same. Determined to get a proper drink, I left my seat at the blackjack table and went to the main bar where I again requested a Dewar's. Now pay attention! This is where it gets bizarre. The bartender reached for a metal clad hose which had a keypad with roughly a dozen buttons near the free end. He suspended the nozzle of this hose above a glass and pushed the lower right hand button. I now had my third glass of mystery solvent but this time I had to pay for it (because I was no longer gambling). I complained to the bartender only to be curtly informed that the lower right hand button did indeed select Dewar's. Well, I didn't buy that for a second. Something like 50 to 100 buttons would be needed to 'punch up' the selections normally available in a properly stocked bar. Hell, even if there had been enough buttons, the drinks would still have been wretched. Since all the booze passes through the same nozzle, every drink has to be tainted with a few drops of whatever was dispensed last. Technology run amok if you ask me! Nonetheless, as a responsible reporter, I felt it my duty to dispel any lingering doubts. During the next hour, I made two more forays to the same bar. The first time I asked for a Chivas Regal, the second time I asked for a Cutty Sark. Needless to say, both were selected, with the lower right hand button. There is a point to this story. If you are concerned that your liver might outlast the rest of your body, help is at hand and now you know just where to look.

When I last visited Las Vegas, microprocessor based slot machines were a rarity. Now they abound! Some of the newer one-armed banditos have even forsaken the familiar spinning reels in favor of full color emulations on a CRT. These machines are quieter, play faster and provide running commentary in the form of snotty messages which only disappear when you

insert more money. I dropped fifty bucks into one of these space age thingies and got less than ten bucks back. No surprise there!

I closed out the first evening back at the tables where I won a few hundred. It never fails; just as soon as you start to win, those pit vipers (pit bosses - the guys in the shiny suits who loiter malevolently behind the dealers and croupiers) start to stare holes right through you. I always find this unsettling. Let's just say any underlying distrust is mutual.

The next day I headed over to the show. I ended up waiting in a very long line because I was one of hundreds (or maybe it was thousands) whose pre-registration had never gotten processed. So much for computers simplifying our existence.

I passed the time exchanging technical badinage with a couple of guys who were stuck in the same line. I could tell they were both computer prodigies; the scuffed shoes and ill fitting suits were a dead giveaway. Some time later, my ID badge fell from the stamping machine and I headed into the exhibit hall.

Now for those of you who may not have attended shows like COMDEX, there are usually lots of promotional freebies. Generally, one is expected to only take one or two of each item, but with minimal practice, you can take as many as I do. In this regard, COMDEX was disappointing. There were really only two items worth grabbing in quantity...

"FREE - TAKE ONE", proclaimed the sign above the large bowl which was filled with complimentary microcassettes and so I did, several dozen times. I'd love to give these generous folks a free plug but they forgot to have the name of their company printed on the cassettes and I was too busy collecting to notice. Whoever you guys are, Thanks. I mean it. I ended up with a like number of nifty little keychain knives courtesy of the Graphic Company. Again, heartfelt thanks... As always, I helped myself with abandon at the Intertec exhibit. Whenever I need to make certain friends in the know chortle uncontrollably, I festoon myself with Intertec's lapel buttons and parade about with a straight face. It gets them every time. So much for the interesting freebies at COMDEX. Take it from this old pro; if you really want to load up on some really neat stuff, pay a visit to CES, NCC or ELECTRO instead.

Some new hardware was introduced (or reintroduced) at COMDEX. I was especially impressed by the new Malibu printer which offers either dot-matrix (fast) or near letter quality (not so fast). I'm happy to report that Malibu seems to have given up on the ersatz wood grain finish which made some of their earlier models too ugly for me to consider seriously. I remain impressed as ever by the Datasouth printer. It seems that Datasouth has been showing essentially the same equipment at the major computer shows for the last few years. This by no means suggests stagnant technology! Datasouth products are so well designed and so carefully built that they remain state of the art in terms of reliability and performance.

There were some interesting new terminals. Unfortunately, the one which clearly represented the best price/performance ratio had problems. I wanted one from the moment I saw it, but two of three samples were dead by the end of day two. I'm not going to tell you any more just yet because I am confident that the manufacturer will get the glitches out very soon. Another terminal which I really liked was the TAB. I first saw one of these beauties at the mini-COMDEX held in New York last May. Why aren't there more TAB terminals in the field? A systems integrator whose judgment is sound beseeched me not to mention Volker-Craig terminals (because he already has enough trouble getting delivery). Fair enough, I won't breathe a word.

I loved the color plotters from Houston Instrument and Hewlett-Packard. If you're rich the former will do quite nicely. If you're even richer then spring for the latter.

Although there was lots of great hardware to be seen, software is my principal interest. For this reason, I enjoyed the Lifeboat exhibit most of all. It featured some truly marvelous products, demonstrated by the authors themselves. Can you imagine my delight at not only seeing state of the art programs being put through their paces but also conversing at length with the gifted individuals who created them? Graftalk, BOSS, SalesPro, Guardian and Angel were amongst the items which I first saw in action at the Lifeboat booth. Splendid packages all!

On the third day of the show, a friend handed me a copy of Infoworld (the COMDEX issue - November 30, 1981). I think they were giving them out but I really don't know (since I'm already a subscriber and therefore had no reason to stop by their booth). Some time back, in these very pages, I praised Infoworld to the skies. Back then, there were only 26 issues a year, all aimed at a sophisticated audience. Now, it's a weekly, bigger and glossier than ever. However, unlike the good old days, it usually winds up in my waste bin about ten minutes after I turn the first page. More is less, to put it bluntly. A few of the editorials are still interesting, but there the similarity ends. The pages which used to be devoted to objective and informative reviews of serious CP/M based application packages are now mostly filled with overly charitable evaluations of trivia written for Apples and TRS-80s. Valuable insights once provided by Adam Osborne and John Craig have been supplanted by the disjointed maunderings of someone who calls herself(?) Minnie Floppy. Maybe that's where the money is these days, but when my subscription lapses, the cost of renewal will remain in my pocket.

To backtrack just a bit, the opening paragraph of this column was inspired by Infoworld's COMDEX issue (which was printed before COMDEX began). Whereas no claims were made that real news about COMDEX '81 could be found therein, little effort was spared in [mis]guiding the casual reader toward the conclusion that Infoworld was scooping the competition with near zero lead times. A brief disclaimer of sorts appears on page 48 of the subsequent issue. (The horse has fled, let's lock the barn)! But I digress.

The real reason my friend went out of his way to make sure I got a look at Infoworld's COMDEX issue was to alert me to the existence of a column pseudonymously written by someone calling herself GIGO. My friend's feeling

(continued next page)

was that GIGO had tried to take my show on the road. Since then, others have told me more or less the same thing. My comment, for the record: "Good God, I sure hope not -GIGO sounds positively dysmenorrheic!". Still one must wonder just how derivative the concept for GIGO's column really was...

For those who missed it, GIGO did an especially thorough hatchet job on 'The Last One', postulating that it did not exist, most likely because the authors had belatedly realized that they would need to learn BASIC in order to write it. (Note: 'The Last One' is a program generator which has been heavily advertised during the last half year or so).

GIGO, let me give you some valuable advice: An inherent trait (or flaw perhaps) of human beings is an abiding urge to perceive things in the best possible light. What this means is that you will rarely have to check your facts, just as long as you deal in flattery. Even when praise is halfwitted and unresearched, those who disagree tend to blame themselves for being incapable of a positive outlook. Quite the opposite applies when one is critical. Should you choose that route, you either research your facts very carefully (in which case a fair number of people will still dislike you), or you don't (in which case the picture grows considerably bleaker). I mention this because 'The Last One' does exist. I have seen it! When you finally do get your hands on a copy, open the white cardboard jacket with the dark blue printing and remove the square, custom molded styrofoam box. Look inside and you will see some Apple-size manuals and a disk which has a label resembling a miniature English banknote. I mention these details so you will know I have not made this any of this up. I don't yet know if 'The Last One' is any good but then again you couldn't have the slightest idea either. Face it kiddo, you got caught with your skirt up the very first time out.

Three days of COMDEX were plenty. I decided to split a day early. It's never easy getting a flight out of Las Vegas when shows like this are in town. But after a few hours of waiting at the airport, with my name on the standby list for every eastbound flight, I fin[illegible] hit the one Jackpot that counted. Bye, bye Las Vegas. I may well [have to] be back for CES in January.

Finally, I did promise a second contest this month. It has been pushed back. Look for it next time. I'll tell you this much; it will be of great help to reread all the articles about random numbers and card shuffling which have appeared in Lifelines over the last few months.

Must run - have fun,

Zoso

Operating Systems

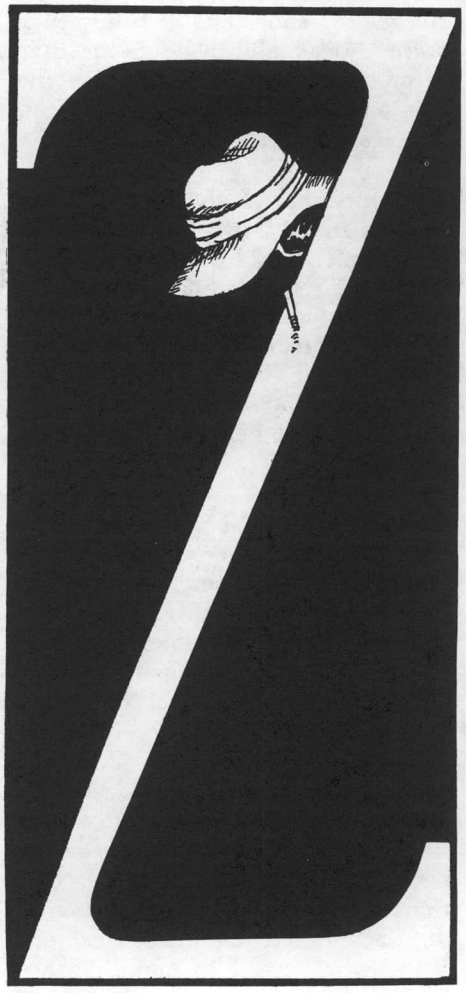
Description	Version

These operating systems are available from Lifeboat Associates, except where otherwise mentioned.

CP/M-80 for:	
Apple II w/Microsoft BASIC	2.20B
Datapoint 1550/2150 DD/SS	2.2
Datapoint 1550/2150 DD/DS	2.2
Datapoint 1550/2150 DD/SS w/CYN	2.2
Datapoint 1550/2150 DD/DS w/CYN	2.2
Durango F-85	2.23
Heath H8 w/H17 Disk	1.43
Heath/Zenith H89	2.2
iCOM 3812	1.42
iCOM 3712 w/Altair Console	1.42
iCOM 3712 w/IMSAI Console	1.42
iCOM Microfloppy (# 2411)	1.41
iCOM 4511/Pertec D3000 Hard Disk	2.22
Intel MDS Single Density	1.4
Intel MDS Single Density	2.2
Intel MDS 800/230 Double Density	2.2
MITS Altair FD400, 510, 3202 Disk	1.41
MITS Altair FD400, 510, 3202 Disk	2.2
Micropolis Mod I - All Consoles	1.411
Micropolis Mod II - All Consoles	1.411
Micropolis Mod I	2.20B
Micropolis Mod II	2.20B
Compal Micropolis Mod II	1.4
Exidy Sorcerer Micropolis Mod I	1.42
Exidy Sorcerer Micropolis Mod II	1.42
Vector MZ Micropolis Mod II	1.411
Versatile 3B Micropolis Mod I	1.411
Versatile 4 Micropolis Mod II	1.411
Horizon North Star SD	1.41
Mostek MDX STD Bus	2.2
Ohio Scientific C3	2.24
Ohio Scientific C3-B/74	2.24B
Ohio Scientific C3-C'(Prime)/36	2.24B
Ohio Scientific C3-D/10	2.24A
Ohio Scientific C3-C	2.24A
Sol North Star SD	1.41
North Star SD IMSAI SIO Console	1.41
North Star SD MITS SIO Console	1.41
North Star SD	2.23A
North Star DD	1.45
North Star DD/QD	2.23A
Processor Technology Helios II	1.41
by Lifeboat/TRS-80 5 ¼"(Mod I)	1.41
by Lifeboat/TRS-80 Mod II	2.25A
by Cybernetics/TRS-80 Mod II	2.25

Hard Disk Modules

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



New Products

The products described below are available from their authors, software houses, computer stores, and software publishers.

Angel
Guardian
Time Management Software

These packages are designed to aid scheduling of all kinds. Angel is basically an expanded version of Guardian, which will be described first.

The user stores information in the form of memo entries, which can be a maximum of 49 characters long. With each memo entry are a series of key dates; these dates are used to determine when the user wishes a reminder of the memo item. Memo entries are private to specified individual users of the package.

Recurring memo items are distinguished according to whether reminders are required on a daily basis, every so many days, monthly, yearly, weekly, or one time only. When the user signs on, s/he is reminded of these items according to the intervals described above, and according to how far in advance the user has specified that s/he desires information on an upcoming event or deadline. The user can be reminded of an item from zero to 99 days in advance; this is referred to as a "prompting interval".

Angel has an extra feature in addition to those described above. It can be utilized to schedule a series of events leading up to a desired goal. Using PERT, this Project Management facility allows the user to define a project as a sequence of tasks which need to be completed in order to achieve a project goal. The tasks can be described by the user in up to 49 characters, along with a projected duration. Each task may have more than one prerequisite, and one task may be a prerequisite for several following tasks. Some tasks may have no prerequisites. Only those tasks immediately preceding or following another must be ordered by the user; the overall planning is done by Angel. The tasks do not have to be entered in any specific sequence; they may be added or changed after the original entry, as may the time relationships between them.

Angel notifies the user of any conflicts that may be accidentally entered by the user. Angel allows estimates (high, low, and typical) to be entered describing the time each part of a project will require. Angel will then arrive at an overall time frame for the project, and will come up with start and due dates. Due date, latest start date, and earliest start date are kept for each item; completions are flagged.

42K of available user memory is required for these packages, along with a cursor-addressable terminal.

RBTE-80
(Remote Batch Terminal Emulator)
Winterhalter and Associates

This program allows a microcomputer to emulate either the IBM 3780, IBM 2780, IBM 3741, or IBM 2770 terminals. These terminals transmit data from disk files or unit record devices (printers, card punches), at speeds of 200-9600 baud, over telephone lines. Data is transmitted in blocks using the Bisynchronous Protocol.

The RBTE program can emulate any of the IBM remote batch terminals by modifying the existing configuration. It runs unattended; no operator interaction is required once the program is initialized. Once line connection has been established, the RBTE will automatically begin a sending/receiving cycle until all pertinent files have been transmitted and received, or until the line is disconnected. Status messages indicate the progress of the transmission.

ASCII and binary data may be transmitted or received with transparent or non-transparent transmission modes and ASCII or EBCDIC transmission codes. Binary codes can only be sent in transparent mode. Data received in non-transparent mode is translated to ASCII and copied to ASCII files. Data received in transparent mode under operator control may be either translated and copied to an ASCII file or copied to a binary file.

A special feature of RBTE-80 displays a diagnostic trace of the received data link control characters, the received message, and the transmitted messages. The diagnostic trace may be viewed on the operator's console or remotely on a spare RS232 port connected to a modem.

The bisync I/O access method is customizable, permitting the programmer to design custom bisynchronous terminal emulations not already provided. The diagnostic trace can be used to trace the communications manager's commands and statuses, for debugging custom emulations.

RBTE-80 requires an 8080, 8085, or Z80 processor, memory for a 30K user program; the system must have a real time clock and serial port capable of full synchronous RS-232, capable of creating interrupts. Lifeboat Associates, the sole distributor, has information on supported computers.

Sales Pro
Winters Associates

This specialized database package is designed to assist sales representatives in maintaining their contacts. A user-definable database permits customization. Basic information such as prospects' addresses, types of business, phone numbers, are recorded. Twelve user-definable questions can be created during initialization; these will be asked later as each new prospect is entered. Nine remark areas for each client accept up to 121 characters apiece; these remark areas act as scratch pads and are automatically dated.

A Closing Formula permits the user to determine how much time should be spent pursuing each prospect. The formula is based on a variety of factors, such as: yearly sales goal, dollar volume of the potential sale, and the percent chance the prospect will buy the product being offered. This feature is intended to prevent the sales person from wasting time trying to sell the wrong prospect.

A number of other reports are included with Sales Pro. They include the prospects sorted by the percent chance they will buy, the days allocated to sell each prospect, days used, and days left. A two page history can be printed for each prospect on file, to allow the sales person to organize their notes on each potential sale.

Sales Pro requires an 8080 CPU, 48K

(continued next page)

TPA, one or two disk drives and a total disk capacity of 430K, a terminal with an 80x24 CRT, a clear screen function and relative cursor movement. A 132 column printer is also necessary.

Publications

Analog Instrumentation Fundamentals
Vincent F. Leonard, Jr.
This book concentrates on basic analog instruments, how they work, are designed, and how to use them. Chapters cover fundamental measurement concepts, analog and digital signals, analog meter movements. DC ammeters, DC voltmeters, ohmmeters, rectifier-type AC voltmeters, peak-type AC voltmeters, DC bridges, transducers, passive RC filters, and attenuators. Lab- type experiments and examples accompany the discussions. A fundamental knowledge of electronics and elementary algebra is required to understand this book.

Micro Publications In Review
Vogeler Publishing, Inc.
This is a monthly quick reference to titles of articles in 70+ micro and mini publications. It reprints the tables of contents and provides a subject index consisting of twenty-six major disciplines with each having from six to forty classifications.

The Index
William H. Wallace
This index lists by subject information printed over the last six years in computer magazines; there are more than 30,000 entries included. Separate indices list subjects by thirteen different machines to which certain articles can be applied.

New Versions

CP/M-80 on TRS-80 Model II
Version 2.25B

Version 2.25B incorporates a number of changes which users have indicated that they would like to see. They are:

1- Fix of a bug in the ADM-31 delete/insert line and character routines. The system no longer fails during an insertion in the last line.
2- The parallel printer routines now have an end of line wraparound function. Instead of the system discarding characters at the end of a physical line, the user has the option, through CONFIG, of allowing printing to continue on the next line.
3- Certain parallel printers produce automatic line feed only when there is a character in the buffer, thus ignoring carriage returns meant only to advance the line. Parallel printers can now be CONFIGed for these semi-auto/lf printers. A typical symptom that you have a semi-auto/lf printer is the way the printer responds to a BASIC LPRINT statement. If an LPRINT prints the line without a line feed, but LPRINT " " prints a line with line feed, then the printer is the semi-auto/lf type and this option should be selected in CONFIG. An example of such a printer is the CENTRONICS 701.
4- In version 2.25A the CONFIG options for AUTO/NON-AUTO parallel printers were reversed; that is, selecting a printer as AUTO/LF set the system up for a NON-AUTO/LF printer. Version 2.25B corrects this.
5- The disk drivers have been modified to further improve error recovery and reduce the possibility of BDOS errors with drives of different stepping rates.
6- GETFILE is not supplied with versions of CP/M higher than 2.24A, and will not function with higher-numbered versions due to changes in the CP/M.

FABS-II
Version 4.1

This update allows compatibility with MP/M and MP/M II.

FPL
Version 2.6

The following changes have been made in this update:

1- Control-C has been disabled.
2- Pagination under the C option has been fixed.
3- EXIT in Editor now does not delete files under any circumstances.
4- VERSION now gives an error message if anything other than Y(es) or N(o) is entered.
5- Worksheets consisting of only rows or only columns can now be saved and relocated in the Editor.
6- A release number has been added to the nomenclature.

New manual pages reflect these changes and the addition of four new features.

One new feature is a program called PATCHER; it can be executed by certain users of "non-standard" versions of CP/M 1.4 who otherwise would be unable to use FPL. The entire fileset of FPL must be resident on the logged-in disk when PATCHER is run.

A new HELP feature explains where you are in the system and what your options are. When prompted by a keyword, it will also describe the functions of that keyword.

When the READ filename or WRITE filename feature of RULES or CALC is in use, the user may direct FPL to query the operator for the name of the file to be READ or WRITTEN.

A Q command will now halt printing and return the user to the DSS command mode. Printing may not resume where it left off, but must resume anew. Similarly, the Q feature may be used to halt the screen's display.

PLAN80
Version 2.1A
This update includes improved terminal handling for terminals lacking CLEAR TO END OF LINE and CLEAR TO END OF SCREEN functions. Highlighting control is improved for terminals using attribute characteristics, such as the Cromemco 3101.

T/MAKER II
Version 2.4

The maximum number of digits in the example of a number has been increased from ten to 13 in this new version. In addition, T/MAKER will now accept characters in the ASCII range from 128 to 254 as well as the standard 32 to 126. this change will only affect users with nonstandard terminals and nonstandard versions of CP/M-80.

OTHERS MAY SEE THE ERRORS OF YOUR WAYZ.

ONLY MICROSPELL CORRECTS THOSE WAYS.

Lifeboat Associates, the world's foremost source for microcomputer software, proudly presents MicroSpell,™ the first program that not only isolates spelling errors in your text, but actually corrects them.

MicroSpell works with your word processor. And if you have the best word processing system, why settle for anything less than the *best* spelling corrector?

Goes Beyond The Competition

Other spelling programs function primarily as spelling checkers, merely pointing out words with suspect spelling. It's left up to you to determine the correct spelling and then type it in.

MicroSpell, the only spelling program that knows how to "spell," corrects the error automatically. Here's how it works: MicroSpell will read your text, carefully looking for words that might be spelled incorrectly. When it comes across a word that it's not absolutely sure of, it stops and shows you that word, along with its context. Then it searches through its own built-in dictionary and presents a list of guesses which it "thinks" might be correct. All you have to do is press a key and the misspelled word is corrected. There is no need to bother with your own dictionary, or even to type in the change. MicroSpell will do it all for you!

The Most Complete Built-In Dictionary

MicroSpell uses word stems and suffix stripping routines, so its dictionary of 25,000 word-parts can deliver over 150,000 words to you. And if that's not enough, MicroSpell will let you add thousands of additional words, so you can create and store specialized dictionaries of technical terms, unusual expressions, even acronyms. And you can let MicroSpell know just when any of these special dictionaries are wanted.

MicroSpell is highly interactive and designed to complement word processing systems that create ASCII text. It requires minimum disk storage capacity of 70K per drive.

So why settle for a program that merely finds your *mispellings* when you can get the one that corrects your *misspellings*? MicroSpell. It corrects the errors of your ways.

MicroSpell is brought to you exclusively and supported completely by Lifeboat Associates. Call or send us the coupon below.

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

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

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

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

Lifeboat Associates GmbH
Hinterbergstrasse
Postfach 251
6330 Cham, Switzerland
Tel: 042-36-8686
Telex: 865265 (MICO CH)

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

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

Lifeboat Associates
The world's foremost software source

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

☐ Please send me more information on MicroSpell.
☐ Please send me a free Lifeboat catalog.

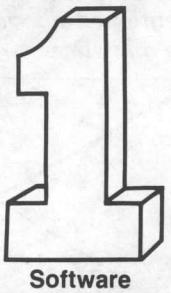
Name_____ Title_____

Company_____

Street_____

City_____ State_____ Zip_____

MicroSpell is a trademark of Bob Lucas.



Software With Full Support

Letters

December 1, 1981

Dear Mr. Gagne,

I read with interest your review of Pascal/MT+ as contained in the November issue of Lifelines. I do, however, take exception to the reputation of MT+ as the "best CP/M-based Pascal system". We are the manufacturers of the Pascal/Z software package, and we feel that our product is by far superior to the Pascal/MT+ system. This feeling is supported by many of our customers, among them Peter Grogono, the author of *Programming in Pascal*, who wrote to us:

> ". . .I am very pleased with Pascal/Z and have used it extensively in my recent work. To the best of my knowledge it is the highest quality Pascal compiler available to users of microprocessors."

I have enclosed copies of recent benchmarks which we conducted in response to those published by MT Microsystems.

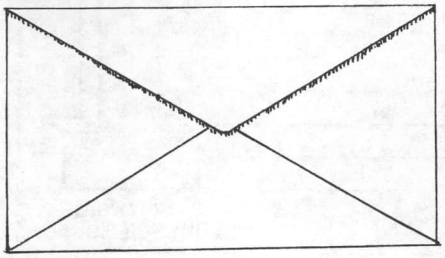
We hope that this information will be of sufficient interest to you that you would write a similar review on Pascal/Z for inclusion in a future issue of Lifelines. Your article seems to be very extensive and objective, and we would welcome your examination of our product.

Please feel free to contact me if you are interested or if you have any questions.

Sincerely,

Laurie Hanselman
Software Products Manager
Ithaca Intersystems

Editor's Note: Readers who wish to receive a copy of the benchmarks mentioned above may request them from Lifelines.



December 15, 1981

Dear Dr. Currie,

In your "Notes on dBASE II" in your November, 1981, issue you referred to the commands as being "BASIC-like", but it's a lot easier to handle if you realize that it's actually a lot closer to Pascal and PL/I. In fact, compared to even these higher-level languages, it's closer to pseudo-code because you can program in concepts rather than in bit-handling codes.

The return of a previous value of a string when you do not enter a new value is similar to the way PL/I-80 operates when you enter a comma as a separator rather than a new value.

There's also an easy fix for the problem you had with the blanks in long fields. The reason for the problem is that you had the searchstring and string backwards. The command operates like this:

@(<searchstring>, <string>).

(I don't understand how you could have made the command:

store @(oldstring," ") to pos

work for you, as this returns a numeric value of zero, since the oldstring is not found in the series of blanks.)

The code that you originally wanted to use works once this change is made. The following was captured using "set alternate on" in my dBASE II system:

```
. ? name
someone
. ? len(Name)
  20

. store $(Name,1,@('  ',Name) - 1)
      TO New
. ? new
someone
. ? len(new)
   7
```

I've been using dBASE II for over a year now and must say that it's one of the smoothest pieces of microcomputer software that anybody has ever written.

Sincerely,

Hal Pawluk
Los Angeles, CA

VERSION LIST

December 11, 1981

The listed software is available from the authors, computer stores distributors, and publishers. Except in the cases noted, all software requires CP/M-80, SB-80, or compatible operating systems.

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

S Standard Version
M Modified Version
P Processor
MR Memory Required

Product	S	M	P	MR	
ACCESS-80	1.0		8080/Z80	54K	
Accounts Payable/Cybernetics	3.1		Z80	64K	Needs RM/COBOL
Accounts Payable/MC	1.0		8080/Z80	56K	For CP/M 2.2
Accounts Payable/Structured Sys	1.3B		8080	52K	w/It Works run time pkg.
Accounts Payable/Peachtree	07-13-80			48K	Needs BASIC-80 4.51
Accounting Plus			8080/Z80	64K	
Accounts Receivable/Cybernetics	3.1		Z80	64K	Needs RM/COBOL
Accounts Receivable/MC	1.0		8080/Z80	56K	CP/M 2.2
Accounts Receivable/Peachtree	07-13-80		8080	48K	Needs BASIC-80 4.51
Accounts Receivable/Structured Sys	1.4C		8080	56K	w/It Works run time pkg.
Address Management System	1.0		8080		Requires 2 drives
ALDS TRSDOS		3.40	8080	32K	Needs TRSDOS. TRSDOS Macro-80
ALGOL 60	4.8C		8080	24K	
ANALYST	2.0		8080	52K	Needs CBASIC2,QSORT/ULTRASORT
APL/V80	3.2		Z80	48K	Needs APL terminal
Apartment Management (Cornwall)	1.0	1.0	8080		Needs CBASIC2
ASM/XITAN	3.11		Z80		
Automated Patient History	1.2		8080	48K	
BASIC Compiler	5.3	5.3	8080	48K	
BASIC-80 Interpreter	5.21	5.21	8080	40K	w/Vers. 4.51,5.21
BASIC Utility Disk	2.0	2.0	8080	48K	
BOSS Financial Accounting System	1.08		8080	48K	Needs 2/3- drives w/min 200k each, & 132-col. printer
BOSS Demo	1.08		8080	48K	
BSTAM Communication System	4.5	4.5	8080	32K	
BDS C Compiler	1.44	1.44T	8080	32K	w/'C' book
Whitesmiths' C Compiler	2.0		8080	60K	
BSTMS	1.2	1.2	8080	24K	
BUG / uBUG Debuggers	2.03		Z80	24K	
CBASIC2 Compiler	2.08		8080	32K	w/CRUN(2,204P, & 238)
CBS Applications Builder	1.3		8080	48K	Needs no support language
CIS COBOL Compiler	4.4,1		8080	48K	
CIS COBOL Compact	3.46	3.46	8080	32K	
FORMS 1 CIS COBOL Form Generator	1.06	1.06	8080		
FORMS 2 CIS COBOL Form Generator	1.1,6a	1.16	8080		
Interface for Mits Q70 Printer					CP/M 1.41 or 2.XX
COBOL-80 Compiler	4.01	4.01	8080	48K	
COBOL-80 PLUS M/SORT	4.01		8080	48K	
CONDOR II	2.06		8080	48K	
CREAM (Real Estate Acct'ng)	2.3		8080	64K	CBASIC needed
Crosstalk	1.4		Z80		
DATASTAR Information Manager	1.101		8080	48K	
Datebook-II	2.03		8080	48K	Needs 80x24 terminal
dBASE-II	2.02A		8080	48K	
dBASE-II Demo	2.02A		8080	48K	
Dental Managememt System 8000	8.7A		8080	48K	Needs CBASIC
Dental Management System 9000	1.07		8080	48K	Needs CBASIC
DESPOOL Print Spooler	2.1A		8080		
DISILOG Z80 Disassembler	4.0	4.0	Z80		Zilog mnemonics
DISTEL Z80/8080 Disassembler	4.0		8080/Z80		Intel mnemonics,TDL extensions
Documate/Plus	**1.0**		**8080**	**36K**	
EDIT Text Editor	2.06		Z80		
EDIT-80 Text Editor	2.02	2.02	8080		
FABS-I	**2.6**		**8080**	**32K**	
FABS II	**4.10**		**8080/Z80**	**48K**	
FILETRAN	1.20			32K	1-way TRS-80 Mod I,TRSDOS to Mod I CP/M
FILETRAN	1.4			32K	Needs TRSDOS. 2-way TRS-80 Mod I,TRSDOS & Mod I CP/M
FILETRAN	1.5			32K	1-way TRS-80 Mod II,TRSDOS to Mod II CP/M
Financial Modeling System	2.0			48K	
Floating Point FORTH	2		8080/Z80	28K	
Floating Point FORTH	3		8080/Z80	28K	
FORTRAN-80 Compiler	3.43	3.43	8080	36K	
FPL 56K Vers.	**2.6**		**8080**	**56K**	
FPL 48K Vers.	**2.6**		**8080**	**48K**	
General Ledger/Cybernetics	1.3C		Z80	48K	Needs RM/COBOL
General Ledger/MC	1.0		8080/Z80	56K	Needs CP/M 2.2 or MP/M
General Ledger/Peachtree	07-13-80		8080	48K	Needs BASIC-80 4.51
General Ledger/Structured Sys	1.4C		8080	52K	w/It Works Package

VERSION LIST

Product	S	M	P	MR	
General Ledger II/CPaids	1.1		8080	48K	Needs BASIC-80 4.51
GLECTOR Accounting System	2.02		8080	56K	Use w/CBASIC2,Selector III
GLECTOR IV Accounting System	1.0		8080		Needs Selector IV
HDBS	1.05A		+	52K	
IBM/CPM	1.1		8080		
Insurance Agency System 9000	1.06		8080		Needs CBASIC
Integrated Acctg Sys/Gen'l Ledger			8080	48K	Needed for 3 pkgs. below
Integrated Acctg Sys/Accts Pyble			8080	48K	
Integrated Acctg Sys/Accts Rcvble			8080	48K	
Integrated Acctg Sys/Payroll			8080	48K	
Interchange			Z80	32K	
Inventory/MicroConsultants	5.3		8080/Z80	56K	Needs CP/M 2.2
Inventory/Peachtree	07-13-80		8080	48K	Needs BASIC-80 4.51
Inventory/Structured Sys	1.0C		8080	52K	w/It Works Package
Job Cost Control System/MC	1.0		8080/Z80	56K	Requires CP/M 2.2
JRT Pascal System	1.4		8080	56K	
LETTERIGHT Text Editor	1.1B		8080	52K	
LINKER			Z80		
MAC	2.0A		8080	20K	
MACRO-80 Macro Assembler Package	3.43	3.43	8080/Z80		
Magic Typewriter	3		Z80	48K	
Magic Wand	1.11		8080	32K	
MAGSAM III	4.2		8080	32K	
MAGSAM IV	1.1		8080	32K	Needs CBASIC
MAILING ADDRESS Mail List System	07-13-80		8080	48K	
Mail-Merge	3.0		8080		
Master Tax	1.0-80		8080	48K	
Matchmaker			8080	32K	
MDBS	1.05A		+	48K	
MDBS-DRS	1.02		+	52K	
MDBS-QRS	1.0		+	52K	
MDBS-RTL	1.0		+	52K	
MDBS-PKG			+	52K	w/all above MDBS products
Medical Management System 8000	8.7a		8080		Needs CBASIC
Medical Management System 9000	1.07		8080		Needs CBASIC
Microcosm			Z80		CP/M 2.X or MP/M
Microspell	4.3		8080	48K	Needs 150K/drive
Mince	2.6		8080	48K	
Mince Demo	2.6		8080	48K	
Mini-Warehouse Mngmt. Sys.	5.5		8080	48K	Needs CBASIC
Money Maestro		1.1	8080/Z80	48K	CP/M 1.4 or 2.2
MP/M-I	1.0				
MP/M-II	2.0		8080	48K	Needs MP/M
MSORT	1.01		8080	48K	
Microstat	2.04		8080	48K	Needs BASIC-80, 5.03 or later
Mu LISP-80/Mu STAR Compiler	2.10	2.12	8080		
Mu SIMP / Mu MATH Package	2.10		8080		muMATH 80
NAD Mail List System	3.0D		8080	48K	
Nevada COBOL	2.0		8080	32K	
Order Entry w/Inventory/Cybernetics			Z80		Needs RM/COBOL
Panel	2.2			44K	Also for MP/M
PAS-3 Medical	1.77		8080	56K	Needs 132-col. printer & CBASIC
PAS-3 Dental	1.63		8080	56K	Needs 132-col. printer & CBASIC
PASM Assembler	1.02		Z80		
Pascal/M	4.02		8080	56K	
PASCAL/MT Compiler	3.2		8080	32K	
PASCAL/MT+ w/SPP	5.5		8080	52K	Also has SuperBr'n & 32K ver., Needs 200K/drive
PASCAL/Z Compiler	4.0		8080	56K	
Payroll/Cybernetics, Inc.			Z80		Needs RM/COBOL
Payroll/Peachtree	07-13-81		8080	48K	Needs BASIC-80 4.51
Payroll/Structured Sys	1.0E		8080	60K	w/It Works run time pkg.
PEARL SD	3.01		8080	56K	w/CBASIC2,Ultrasort II
PLAN80 Financial Package (Z80/8080)	**2.1a**		**8080**	**56K**	**Z80/8080**
PL/I-80	1.3		8080	48K	
PLINK I Linking Loader	3.28		Z80	24K	
PLINK-II Linking Loader	1.10A		Z80	48K	
PMATE	3.02		8080	32K	
PRISM/ADS	2.0.1		8080	56K	Needs CBASIC, 2.06 or later & 180K/drive
PRISM/IMS	2.0.1		8080	56K	Needs CBASIC, 2.06 or later & 180K/drive
PRISM/LMS	2.0.1		8080	56K	Needs CBASIC, 2.06 or later & 180K/drive
POSTMASTER Mail List System	**3.5**	**3.5**	**8080**	**48K**	
Professional Time Acctg	3.11a		8080	48K	Needs CBASIC2
Programmer's Apprentice			8080/Z80	56K	needs Basic-80
Property Management Program (AMC)	4.2		Z80	48K	Needs CBASIC 2.07+, CP/M-80 2.0+
Property Management System	07-13-80		8080		Needs BASIC-80 4.51

(continued next page)

VERSION LIST

Product	S	M	P	MR	
Property Manager	1.0		8080	48K	Needs CBASIC
PSORT	1.2		8080		
QSORT Sort Program	2.0		8080	48K	
Real Estate Acquisition Programs	2.1		8080	56K	Needs CBASIC
Remote	3.01		Z80		
Residential Prop. Mngemt. Sys.	1.0		Z80	48K	
RM/COBOL Compiler	1.3C		8080	48K	w/Cybernetics CP/M 2
RAID	**5.0.2**	**5.0.2**	**8080**	**28K**	**Modified for TRS-80 Model-I only!**
RAID w/FPP	**5.0.2**	**5.0.2**	**8080**	**40K**	
RECLAIM Disk Verification Program	2.1		8080	16K	
SBASIC	5.4		8080	48K	
Scribble	1.3		8080		
SELECTOR-III-C2 Data Manager	3.24	3.24	8080	48K	Needs CBASIC
SELECTOR-IV	**2.17**		**8080**	**52K**	**Needs CBASIC**
Shortax	1.2		Z80	48K	TRSDOS,MDOS too, needs BASIC-80 5.0
SID Symbolic Debugger	1.4		8080		N/A-Superbr'n
SMAL/80 Programming System	3.0		8080		For CP/M 1.x
Spellguard	2.0		8080/Z80	32K	Needs Word Processing Program
Standard Tax	1.0		8080	48K	Needs BASIC-80 4.51
STATPAK	1.2	1.2	8080		NeedsBASIC-80 4.2 or above
STIFF UPPER LISP	**2.6**		**8080**	**48K**	
STRING BIT FORTRAN Routines	1.02	1.02	8080		
STRING/80 bit FORTRAN Routines	1.22		8080		
STRING/80 bit Source	1.22		8080		
SUPER SORT I Sort Package	1.5		8080		Max. record=4096 bytes
SELECT			8080/Z80	40K	
T/MAKER II	**2.4**		**8080**	**48K**	**Avail. for CDOS**
T/MAKER II DEMO	**2.4**		**8080**	**48K**	
TEX Text Formatter	2.1		8080	36K	
TEXTWRITER-III	3.6	3.6	8080	32K	
TINY C Interpreter	800102C		8080		
TINY C-II Compiler	800201		8080		
TRS-80 Customization Disk	1.3B		8080		
ULTRASORT II	**4.1B**		**8080**	**48K**	
Lifeboat Unlock	1.3		8080		Use w/BASIC-80 5.2
VISAM	2.1		8080	48K	
Wiremaster			Z80		Needs 180K/drive
Wordindex	3.0		8080	48K	Needs WordStar
Wordmaster	1.07A		8080	40K	
WordStar	3.0		8080	48K	
WordStar w/MailMerge	3.0		8080	48K	
WordStar Customization Notes	3.0		8080		
XASM-05 Cross Assembler	1.05		8080	48K	
XASM-09 Cross Assembler	1.07		8080	48K	
XASM-51 Cross Assembler	1.09		8080	48K	
XASM-F8 Cross Assembler	1.04		8080	48K	
XASM-400 Cross Assembler	1.03		8080	48K	
XASM-18 Cross Assembler	1.41		8080		
XASM-48 Cross Assembler	1.62		8080		
XASM-65 Cross Assembler	1.97		8080		
XASM-68 Cross Assembler	2.00		8080		
XYBASIC Extended Interpreter	2.11		8080		
XYBASIC Extended Disk Interpreter	2.11		8080		
XYBASIC Extended Compiler	2.0		8080		
XYBASIC Extended Romable	2.1		8080		
XYBASIC Integer Interpreter	1.7		8080		
XYBASIC Integer Compiler	2.0		8080		
XYBASIC Integer Romable	1.7		8080		
ZAP-80	1.4		8080		Needs 50K/drive
Z80 Development Package	3.5		Z80		N/A-Magnolia,Superbr'n,mod.CP/M
ZDM/ZDMZ Debugger	1.2/2.0		Z80		For N'Star,Apple,IBM 8"
ZDT Z80 Debugger	1.41	1.41	Z80		N/A-Superbr'n,mod.CP/M
ZSID Z80 Debugger	1.4A		Z80		N/A-Superbr'n,mod.CP/M

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

WIN AN IBM* PERSONAL COMPUTER FROM THE NEW MAGAZINE THAT FEATURES THEM.

ANNOUNCING PC™ THE INDEPENDENT GUIDE TO IBM PERSONAL COMPUTERS.

If you're interested in the new IBM Personal Computer, you'll be *very* interested in new *PC* magazine.

Packed with vital information, *PC* will help you keep up with the ever-expanding variety of uses for IBM Personal Computers. *PC* will provide hundreds of helpful tips on adapting Personal Computers for your own needs, and keep you up-to-date on all new developments affecting IBM "PCs" and the people who use them.

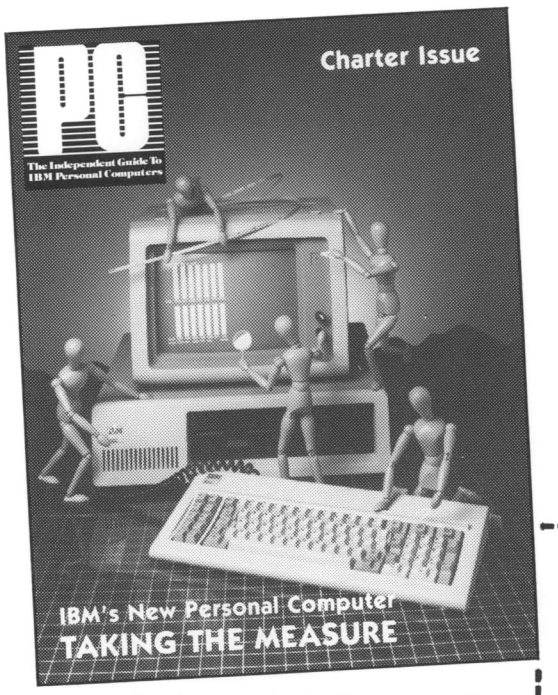
PC even has a special department called PC-Lab™ to evaluate hardware, software and supplies that can be used with IBM Personal Computers—helping you choose those that best suit your needs.

When IBM introduced its Personal Computer, an IBM executive said, "If you could choose one word to describe this system, it would be 'quality.' " That will also be the guiding light for *PC* magazine—in its easy-to-understand writing, its colorful design, and particularly in its devotion to you, the reader.

PC's premiere issue will be out in January, and no one who wants to know more about IBM Personal Computers should be without it—or any of the informative issues to follow. So if you're interested in IBM Personal Computers, be a *PC* Charter Subscriber.

OUR NO-RISK TRIAL OFFER

PC's special offer for Charter Subscribers is the first six issues for just $12, a $6 saving from the single-copy price. And if you're not satisfied with your first issue for any reason, you can cancel and get a refund. Simply write "I want a refund" on the mailing label of your first copy, send it back to us within 10 days, and *PC* will refund your payment in full plus 20¢ to pay for your stamp.



ANNOUNCING THE PC "PC" GIVEAWAY.

PC wants to know who's interested in the new IBM "PC," so we're giving one away to help gauge public interest. You could be the one to win our prize—an IBM Personal Computer system unit (16K) with keyboard and color/graphics monitor adapter. (Available IBM products of equal value may be substituted if you prefer.)

You don't have to subscribe to enter our Giveaway drawing; just fill out and mail the "Giveaway" end of the double coupon below. But why not use the other half of the coupon to enter your subscription at the same time. After all, with our no-risk trial offer, you can't lose. And you'll be sure not to miss out on *PC's* information-filled Premiere Issue.

GIVEAWAY RULES

☐ Mail entries postpaid to the address on the entry blank. Entries must be postmarked before midnight, March 1, 1982. Drawing will be held, and the winner notified by April 1, 1982. Prize delivery date will be subject to IBM product availabilities. ☐ Entries must be on an official entry form or its exact copy. ☐ You may enter as many times as you wish, but each entry must be mailed in a separate envelope. ☐ Employees of Software Communications, Inc., its affiliates, dealers, distributers, advertising agencies and media not eligible. ☐ Void where prohibited, taxed or restricted by law.

SUBSCRIBE TO PC NOW, AT NO RISK.

☐ **YES I want to be a Charter Subscriber to PC.** My charge information or check for $12 made out to **PC is enclosed.**

PC
1239 21st Avenue
San Francisco, CA 94122

Name _____

Address: _____

_____

City _____

State_____Zip_____

☐ Visa ☐ MasterCard ☐ Check Enclosed

Acct. # _____

Exp. Date _____ Bank # (MC only) _____

Signature _____

ENTER THE PC "PC" GIVEAWAY.

☐ **YES Enter my name in PC's drawing to give away an IBM Personal Computer.**

Area Code/Phone#_____

I now have:☐ an IBM Personal Computer
☐ another personal computer
☐ no personal computer

*PC is an independent journal, not affiliated in any way with International Business Machines Corporation. IBM is a registered trademark of International Business Machines Corporation. *PC* and PC-Lab are trademarks of Software Communications, Inc.

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

Second Class Postage Paid
At New York, N.Y.