JUNE 1981

LIFELINES®

$2.50

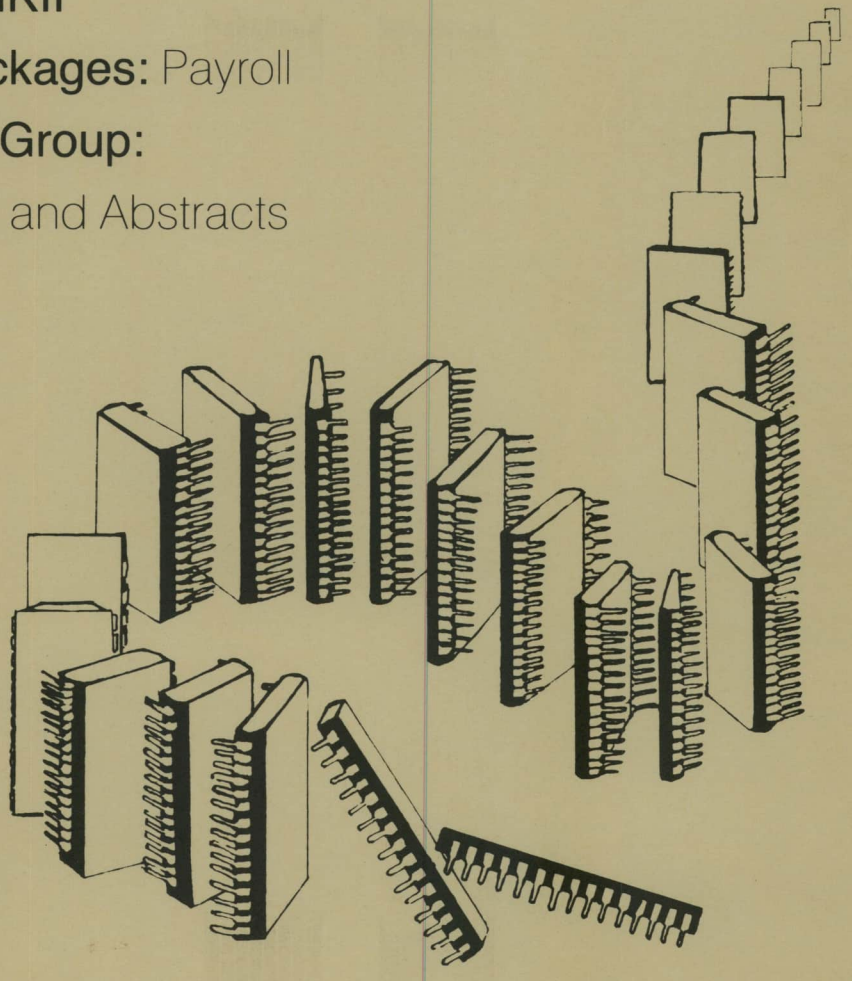
VOLUME TWO NO.1

The Software Evaluation Group:
SELECTOR IV

A Review of PLINKII

The Osborne Packages: Payroll

The CP/M Users Group:
Catalogue and Abstracts



LIFELINES®

Editor-in-Chief: Harris Landgarten
Managing Editor: Jane Mellin
Production Assistant: K. Gartner

Volume II No.1 June

CONTENTS

Lifelines, Volume II, Number 1. Published monthly. The single copy price is $2.50 domestically, including the U.S., Canada, and Mexico. The single issue price for copies sent to all other countries is $3.60. A one year's (12 issues) subscription is priced at $18.00, when destined for the U.S., Canada, or Mexico, $40 when destined for any other country. All checks should be made payable to Lifelines Publishing Corporation. Foreign checks must be in U.S. dollars, drawn on a U.S. bank; checks, money orders, VISA, and MasterCard are acceptable. All orders must be prepaid. Lifelines is published by Lifelines Publishing Corporation, 1651 Third Avenue, New York, N.Y. 10028; telephone: 212-722-1700. Please send all correspondence to the the Publisher at the above address. Postmaster send change of address to the above address. Application to mail at 2nd class postage pending at New York, N.Y.

Editorial Comments

The 1981 NCC is now history and it was as big and exhausting as ever. In general the show was dominated by hardware, with software taking a back seat. For the first time the segregation of microcomputer products from minis and mainframes was virtually eliminated. True, the two floors of exhibit space were nominally called the "Main Exhibit Area" and the "Personal Computer Exhibit Area" but the intermix of product types on both floors totally blurred the distinction. Perhaps Affips will eventually realize that computers are computers no matter how big.

I urge you to keep in mind that much of what is shown at an exhibition such as this is imaginary in terms of current availability. Many of the products introduced at last year's NCC have not yet been produced. It would be a mistake to plan on the availability of any computer product which has never been shipped to customers, no matter what the vendor promises. This applies to software as well as hardware.

The only microcomputer software vendors exhibiting at the show were Microsoft, Micropro, Lifeboat Associates, and Structured Systems. I suspect that more would come if they realized the marketing power of this 'mega-show' (80,000+ attended). In fact the gate was held down because of insufficient hotel space in the Chicago area - ironic considering that the show itself only filled half of gigantic McCormack Place. Some people who made last minute reservations were placed twenty miles away while those who came impromptu found themselves without quarters.

Microsoft demonstrated XENIX on a Z8000 computer in their booth. Some UNIX freaks I know told me it was the real thing. The release date is supposed to be in June. "Leading-edge" types had better start lining up equipment or be left behind. XENIX for the 8086 and 68000 were announced with delivery quoted for the third and fourth quarters respectively. The 68000 version will be totally reworked in order to fully take advantage of this powerful chip. No prices have been announced yet. Lifeboat did announce prices for the PDP/11 version of XENIX starting at $500 for a single user and ranging into the $5000 range for a 32 user system. It is suspected that prices for the 16 bit micro versions will be lower. For those of you who are still anxiously waiting, I am sorry to report that Microsoft APL is officially dead. The CP/M version of Microsoft's Pascal (MS/Pascal) is, however, alive and kicking. In fact it was running at the show and should be ready for distribution soon. At the moment it needs a 64k system (whatever that means) to run. Work is being done to reduce its size.

The pace of the "Chain with Common" story is picking up. Bill Gates personally assured me that a new version of the BASIC Compiler implementing this feature will be released by mid May. Greg Whitten, the man responsible for such things, concurred stating that the new compiler was indeed finished and worked beautifully. In view of this, I apologize for misleading you in last month's editorial. But wait, that's not the end of the story. Several independent people told me that they visited the Microsoft booth and were told that "Chain with Common" wouldn't be ready until the summer. To confuse matters more, I was shown a letter from Microsoft to a customer stating that the new compiler would be delivered on May 1st (oops!). For the moment we will go with the official story and hope that next month's Lifelines will contain a review of the new BASCOM.

Over at the Micropro booth WordStar 3 and SpellStar were being demoed. The new WordStar implements horizontal scrolling in user selectable increments, and a column move feature. This later feature allows you to mark vertical columns as blocks and then copy or move them as with normal block moves, facilitating manual construction of multi-column pages and other common wordprocessing tasks. Although previously alluded to, this new WordStar will not support true proportional printing with PS print wheels nor does Micropro appear to be close to implementing this admittedly difficult feature. Another thing WordStar 3 will do is interface with Spellstar, Micropro's new spelling checker. SpellStar will be delivered as an overlay file (a la MailMerge) and will only run with WordStar 3. Its operation is much like Spellguard with the added advantage of being interactively linked to its host wordprocessor after the batch type spell scan is completed. Neither WordStar 3 or SpellStar will be available for another month. A full review of these new products will be published when they are released.

Sol Libes of S100 Microsystems hosted another panel discussion, similar to the one held last month in San Francisco. Approximately fifty attended despite the fact that the session was improperly titled, erroneously scheduled, and was left out of the show guide because of NCC ineptness. Sol did a wonderful job under the extreme circumstances. Speakers included Gary Kildahl, Tony Gold, and Ward

(continued on page 17)

A Review of PLINK II

by Harris Landgarten

Every so often a piece of software appears which profoundly affects the state of the art of microcomputer programming. Examples which immediately come to mind are Microsoft's BASIC-80, Compiler Systems' CBASIC, and Micropro's Wordmaster and Wordstar programs. I predict that PLINKII, a new linkage editor by Phoenix Software Associates, will play a major role in elevating the level of application programs available to the end user. Why should I make such bold claims for a lowly linkage editor? Perhaps some historical background is in order.

The first CP/M programming tool which generated RELocatable code for later linking was Microsoft's Macro-80 assembly language development package. The general idea is to assemble separate source code modules into relocatable object code so that L80, the supplied linker, can intelligently combine only the desired modules into an executable file. The advantages of this scheme are numerous. Libraries of oft-used routines can be maintained in REL format so that the linker can automatically pick the required ones during link-time. If a bug has been discovered in one module, only that module has to be reassembled; it is then relinked with the other already-assembled modules. This scheme of generating Microsoft type REL files for later linking is followed by many compilers now on the market. Some of the more popular ones are FORTRAN-80, BASIC-80, COBOL-80, PL/I-80, Pascal/MT+, Pascal/Z and Whitesmiths' C. All of these packages include linkers which perform adequately--as long as they aren't stretched to their limits. The problem is that the constraints are uncomfortably tight for those developing large applications. The reasons for this restriction vary from linker to linker, but the common theme is an inability to properly use the host computer's memory space. A prime example is Microsoft's L80. All linkers must retain tables which are built, maintained and used during the linkage process. L80 maintains these tables in the computer's internal memory. Unfortunately, the executable form of the program is also built in memory and L80 itself takes some 8K of memory. Since no whole can be larger than the sum of its parts, we are led to the conclusion that programs linked with L80 will not be able to use all the memory available to them during run-time. In fact, users of 64k machines find that programs of approximately 45k are the largest that can be constructed with L80. Digital Research distributes LINK-80 both as part of their PL/I-80 package and separately. LINK-80 attempts to solve the memory usage problem in two ways. First, and most importantly, it allows the user to split his program into overlays. This means that routines are only in memory during run-time, when they are actually needed. At other times, other routines use the same memory space. This is accomplished by automatically swapping routines from disk. While this vastly increases the total amount of memory available for an application, it does nothing to ease the limits placed on the amount of code which can be resident at any one time. In fact LINK-80 constrains concurrent memory usage even more than L80 because the linker itself is larger. This is an important limitation because some applications require large resident root sections and do not overlay well. Furthermore, good program design dictates the limited use of overlays, both for speed and clarity of structure. Overlays provide enormous power, but should be used to overcome real memory constraints, not artificially imposed ones. Digital tries to solve this problem by providing an optional switch which causes LINK-80 to write the tables to disk during linkage while continuing to build the program in memory. This measure only buys a modicum of extra memory space at the expense of painfully slower linkage times.

PLINKII is a two-pass linkage editor. Although a Z80 microprocessor is required for operation, code for 8080 target machines can be linked. Since the output file is built up on disk, 64k programs which completely fill the address space can be created even on smaller machines. PLINKII offers complete control over the memory utilization of the program, and has the ability to create arbitrary overlay structures. When overlays are used, programs consisting of up to 8 megabytes of code may be created-when the current version of CP/M version 2 is used.

PLINKII accepts as input REL files which contain one or more named modules (program units). Files containing more than one module are called libraries. Modules are composed of segments (sometimes called relocation bases) of several types. These are usually the code segment, the local data segment, and various common blocks. Segments which form a continuous unit of memory are called sections (each separate overlay is called a section). PLINKII works by dividing the input modules into segments, regrouping the segments into sections and then organizing the sections into the output program.

In the CP/M environment, the user can choose between two types of output files: COM or PRG. COM files are directly executable and contain an exact memory image of the program with an assumed origin of 100 Hex. PRG files are not directly executable under CP/M. They contain not only the binary code itself but information about the code such as version number, date created, and the load addresses of the various code sections. A PRG file is run by using the EXECUTE utility provided with PLINKII. PRG files have the advantage of being able to hold code with discontinuous or conflicting memory assignments. In this way programs containing a large number of overlays are still held within one PRG file saving disk space and improving media portability. More advantages will be discussed later.

The simplest way to describe PLINKII's features is to explain

(continued next page)

the usage of some of its commands.

After loading PLINKII the first thing the programmer does is specify the files to be input. This is accomplished using the FILE, LIBRARY, and SEARCH statements. The FILE statement causes all the files specified to be loaded in their entirety. The LIBRARY command causes a single pass of a REL library with only modules satisfying unresolved references loading. SEARCH is identical to LIBRARY except that multiple passes of the library are performed if undefined symbols remain after all files have been read. The library request feature available in Microsoft products is honored, making it unnecessary to explicitly specify compiler libraries such as FORLIB.REL. The INCLUDE and EXCLUDE statements work in conjunction with the just-mentioned input commands to provide a select capability. INCLUDE causes only the modules mentioned to be considered for loading, while EXCLUDE prevents the named modules from being included in the linkage edit. All action takes place before any library searches; INCLUDE and EXCLUDE cannot be used on the same file.

By default, all modules loaded by the FILE, LIBRARY, and SEARCH commands are allocated to the next available low memory with uninitialized data segments sorted to the end of the section. This latter feature can result in substantial savings in disk space, since uninitialized data areas are not stored as part of the COM file being output. One disk of mine contained 460k of compiled BASIC-80 programs which were reduced in size to 380k merely by relinking all the programs with PLINKII. This feature can be overridden with the NOSORT command for special cases.

MODULE and SEGMENT statements are provided so you can select individual modules and segments which have been read into another section, and make them part of the current section. By using this feature, you can load all of the input files in one place and then selectively place code segments into various overlay sections.

The LOCATE statement allows you to place code at any address specified. For example:

```
FILE    PROG1,PROG2
LOCATE  4000H
SEGMENT SEG1
```

will cause the modules in PROG1 and PROG2 to load at the beginning of available memory; segment SEG1 is loaded at 4000H, even if it is contained in PROG1 or PROG2. Furthermore, the use of PRG files will eliminate the customary "holes" that would be present in a COM file of this type.

The ACTUAL command permits modules to be loaded at addresses differing from their actual run address. This is useful for code which must run above CP/M such as extra I/O drivers. PLINKII's overlay capability can be used to automatically relocate such routines.

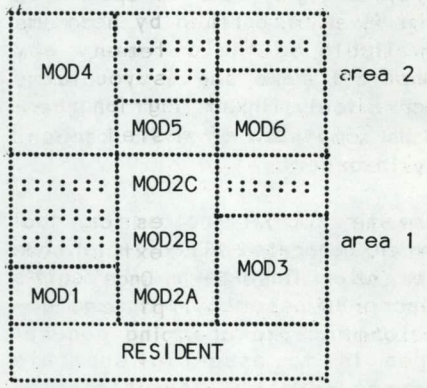
Users working in a ROM environment will appreciate the PROG and DATA statements. These commands cause all program segments or all data segments to be reorganized and grouped into separate sections. The program section can then be targeted to the ROM address by using the LOCATE command.

When a program is too large to fit in memory, overlays must be used. PLINKII allows you to construct arbitrarily complex overlay schemes. Whenever overlays are used, PLINKII includes its overlay loader in the output PRG file. (The output file must be of type PRG if it contains overlays.) Three overlay loaders are supplied; one requires a Z80, one runs with an 8080, and one contains debugger code which prints each overlay name on the console as it is called during run-time. An overlay structure containing two independent overlay areas would be constructed as follows.

```
OUTPUT TEST.PRG
 FILE F1, F2, F3, F4, F5 ,F6,F7
 BEGIN
    OVERLAY SEG MOD1
    OVERLAY SEG MOD2A,MOD2B,MOD2C
    OVERLAY SEG MOD3
 END

 BEGIN
    OVERLAY SEG MOD4
    OVERLAY SEG MOD5
    OVERLAY SEG MOD6
 END
```

Note that all of the modules named in the overlay statements are contained in the files F1-F7.

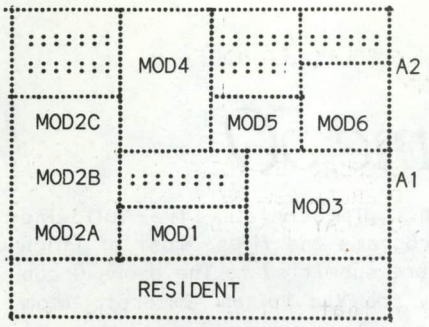


The above figure diagrams the memory usage of the the sample overlay description. Note that areas filled with ':' (colons) are unused memory areas. For special situations, when it is known in advance that certain overlays will never be used while others are in memory, the SHARE statement is provided. For example, suppose that Mods 4-6 will never be accessed when Mods 2a-2c are in memory. Memory could be saved by using the following overlay structure.

```
OUTPUT TEST1.PRG
FILE F1, F2, F3, F4, F5, F6, F7
BEGIN
 OVERLAY SEG MOD1
 SHARE
  OVERLAY SEG MOD2A,MOD2B,MOD2C
 OVERLAY SEG MOD3
END

BEGIN
 OVERLAY SEG MOD4
 OVERLAY SEG MOD5
 OVERLAY SEG MOD6
END
```

The following figure illustrates this structure. Note that when Mods 2a-2c are in memory no others may be. When Mods 1 or 3 are in memory, either 4, 5, or 6 may also be.



Hierarchial (tree-like) overlay structures may also be specified. This is accomplished by nesting the BEGIN - END statements in much the same way as you nest loops in a programming language. Up to 32 levels of nested overlays are allowed.

Linkage command files can be constructed with a text editor for use with PLINKII. Once written, a linkage script can be executed simply by typing

PLINKII @FILENAME

with the file having the default type LNK. This is particularly useful when repeatedly linking the same program, for instance during the debug cycle of a new application.

A console I/O facility is provided. Operator messages can be displayed; values can be input into any of the 50 local variables provided. Any PLINKII statement may be prefaced with a label. Both conditional and unconditional transfer of control to a label are provided. These in conjunction with other simple programming language type constructs allow linkage scripts to be written for 'menu-driven' linkages. These scripts can be stored as LNK files for future use. An example of this would be the controlled linkage of I/O drivers into a CP/M BIOS by operator input at link-time.

In actual use, PLINKII performed flawlessly. The complexity of operation varies from

PLINKII FILE PROGRAM

for the linkage of a simple BASIC program to the more complex examples given earlier for overlays

The speed of the linker is remarkably fast for a disk to disk type linkage editor. Large (50k) BASIC programs were linked in under 8 minutes on double density 8 inch floppies. Overlaid programs typically linked in under 10 minutes depending upon the size and complexity of the application. It should be kept in mind that PLINKII is heavily disk bound and that operating times will vary several orders of magnitude between hard disks and 5 inch floppies. All disk to disk type loaders are significantly slower than in memory types, making the disk speed question critical for those using PLINKII extensively. Don't be surprised by one hour or more linkage times on slow media. One interesting point of reference is that PLINKII is approximately three to four times the speed of PLINK, Phoenix Software's first disk to disk linker.

Facilities are provided which allow several memory map reports of different complexities to be produced. These reports can selectively be output to either the list device or to a disk file. No option is provided for output to the console.

PLINKII has been tested with BASIC-80, FORTRAN-80, PL/I-80, Pascal/MT+, Pascal/Z, and Whitesmiths' C. Please note that overlays are not supported by BASIC-80 because the language contains no subroutine facility. If you would like to see overlays in BASIC-80, I suggest that you write to Microsoft and ask them to include a Subroutine and Common feature in their next release of the BASIC Compiler. At this time PLINKII will not overlay COBOL-80 programs, using the section statement. A future enhancement will allow the true segmentation of COBOL-80 programs.

The documentation is very complete and reasonably well organized. It is written in an understandable manner and a sufficient number of tutorial examples are included. The documentation is clearly pointed towards someone of reasonable computer literacy (e.g. one who understands this review). No index is included; however, the table of contents should suffice since it is organized by command.

My only criticism of PLINKII is the fact that nothing is output to the console during the linkage process. Since linking can take more than a half hour for completion, it would be nice if some sort of verification of operation was given periodically. People are apt to manually abort a good link because they think the system has crashed. You can get especially nervous during the times of up to one minute when no disk activity takes place at all. Furthermore it would be helpful if you could somehow gauge how far along the process is. The author has told me that a 'verbose' option is being considered which would satisfy these objections.

PLINKII is available from Lifeboat Associates for $350. The manual is not available separately. For an additional $350 you can purchase an unlimited license to distribute the overlay handler and the execute utility as part of your application. Updates from PLINK are not available. PLINK will continue to be sold and supported. PLINKII is considered a totally new and different product, a fact borne out by its operation. This product is clearly a professional programming tool; if you use any of the languages listed above, it's a tool you can't afford to be without. Even if you don't use PLINKII yourself, you will know of its impact when you start seeing 100k programs with PRG extensions proliferating the market.

5000

Our 5000th subscriber is Thomas Bonoma of Concord, Massachusetts. We hope this is a sign of good luck for Mr. Bonoma. We are awarding him six extra issues. Thanks to all our subscribers for helping this first year to be a good one.

The CP/M Users Group Catalogue and Abstracts, Volume 50

DESCRIPTION: Bob Van Valzah's "Pascal Pascal Compiler" and a few miscellaneous programs for printing via UNIX.

NUMBER	SIZE	NAME	COMMENTS
		-CATALOG.050	CONTENTS OF CP/M VOL. 50
50.1	1K	A.OCO	Sample program, eight queens problem, ready to be linked to RTP.COM (OCO=obj code)
50.2	1K	A.PCO	Sample program pcode, ready for PFET.COM.
50.3	5K	ABSTRACT.050	Abstracts for this volume.
50.4	2K	COMPARE.COM	Vital program for compiler writers.
50.5	2K	CPMDIR.C	Prints CP/M DIR on UNIX.
50.6	2K	CRCK.COM	CRC program for validating files on this disk.
50.7	2K	CRCKLIST.CRC	CRC list of files on this disk.
50.8	3K	DISK.DOC	Bob's comments.
50.9	3K	EQ.COM	Prints all solutions to the eight queens problem.
50.10	2K	EQ.PAS	Eight queens problem.
50.11	5K	FROMCPM.C	Print CP/M file via UNIX stdout.
50.12	1K	FWD.PAS	Forward Procedure Declarations.
50.13	6K	HW5.COM	Builds optimal binary search tree
50.14	15K	HW5.PAS	and decodes a message.
50.15	1K	HW5DATA.	Sample data for above program.
50.16	6K	PASYNTAX.DOC	Syntax graphs for PPC compiler.
50.17	1K	PC.SUB	Submit file to compile a PPC program.
50.18	7K	PFET.COM	Object code of Pcode to 8080 translator.
50.19	11K	PFET.PAS	Source of above.
50.20	1K	PHONE.C	C program to print words you can spell with your phone #.
50.21	1K	PLAYDATA.	Sample data for above program.
50.22	13K	PLAYKAL.PAS	PPC program to determine best moves in game of Kalah.
50.23	2K	POPS.DOC	DOC for Pcodes used by PPC.
50.24	1K	POWTWO.PAS	PPC Program to print negative powers of two.
50.25	16K	PPC.COM	Pascal Pascal Compiler.
50.26	10K	PPC.DOC	DOC for above file.
50.27	26K	PPC.PAS	Pascal source for above file.
50.28	2K	PSTACK.DOC	DOC on run-time P-machine stack.
50.29	3K	REGEN.DOC	Notes on how to modify and compile PPC.PAS.
50.30	11K	RTP.ASM	Source for run-time package.
50.31	2K	RTP.COM	Object of above.
50.32	1K	STIRLING.PAS	Prints a table of Stirling Numbers.
50.33	4K	TESTER.PAS	Tests functionality of PPC.
50.34	1K	VALIDATE.SUB	Submit file to make sure you have a "fertile" computer.

This disk contains the following programs and files, most of which were submitted to The Users Group by Bob Van Valzah. Excerpts from the documentation for Van Valzah's Pascal Pascal are included with the abstracts.

Notes by Bob Van Valzah:

A.OCO This sample program is the object code output of the PFET portion of the compiler package. (OCO = Object COde).

A.PCO This sample program is the P-code output of the PPC.COM portion of the compiler. Both A.PCO and A.OCO, in this case, are partially compiled outputs of EQ.PAS, the eight queens problem. Normally, if you use the PC.SUB file for compilations, A.OCO and A.PCO will both be written to disk and subsequently erased by the submit file.

COMPARE.COM program to compare two files, from The CP/M Users Group Volume 40, useful for checking differences between different versions of the same program.

CPMDIR.C This is a UNIX C program for printing a CP/M directory to STDOUT (UNIX printer).

DISK.DOC Bob Van Valzah's notes on some of the files on this disk.

EQ.COM Compiled output of EQ.PAS, the eight queens chess problem, written by Bob.

EQ.PAS Source code in Pascal to print out all solutions to the eight queens chess problem.

FROMCPM.C This is a UNIX C program to print a CP/M file to STDOUT via modem.

FWD.PAS Sample Pascal Program.

HW5.COM Sample program.

HW5.PAS Sample PASCAL program to build an optimal search tree and

decode a message.

HW5DATA. Sample data for above program.

PASYNTAX.DOC Bob's notes on PASCAL syntax.

PC.SUB Submit file for compiling from .PAS file to .COM file.

PFET.COM Part of PPC compiler package.

PFET.PAS Source code for the PFET portion of the compiler package (compiles P-code into object code).

PHONE.C UNIX C program to print out the words you can spell with your phone number.

PLAYDATA. Data for following program.

PLAYKAL.PAS Sample program to determine best moves in a game of Kalah (Anybody know how to play Kalah?)

POPS.DOC This file is Bob's documentation on the P-codes used by the compiler.

The compiler does not generate all of the p-codes given here. Some were for planned enhancements that never were finished. Similarly, the translator (PFET) will translate many p-codes that the compiler presently does not generate. There may be some p-codes it does generate that are not listed here, but this these are the bulk of the useful ones and will give you the general idea.

POWTWO.PAS Sample program to print out the negative powers of 2.

PPC.COM PASCAL PASCAL COMPILER.

PPC.DOC Documentation on the compiler.

If you have a file named DOG.PAS and you want to compile it, you'd just type

```
submit pc dog
```

The compiler will ask "LISTING?". You reply with a single character; carriage return means no listing, any other character means "yes listing". The listing will be sent to the console as the compilation proceeds. Any errors detected in the compilation are flagged in this listing. At some point (hopefully reasonably near to the point of infraction) the error number will be inserted into the listing, enclosed in ">>" and "<<". The line following an error will start with "*********" and otherwise be blank to call attention to the error. The compiler will also wait for a single character from the console before compilation continues. This is so people with CRT's can see the error. Error numbers should be looked up in Jensen and Wirth (see below). Error number 99 is pound sign ("#") expected.

The compiler should work with a 32k CP/M and might work in 24k, but there are no memory overflow checks. If it hangs or something, you probably don't have enough memory.

The program PPC.COM takes your Pascal source and makes a single pass over it, translating it to a sort of p-code as it goes. This p-code is written to disk. PFET.COM reads the p-code file on its first pass, assigning 8080 addresses to all p-code labels and storing the p-code in memory for the second pass. On its second pass, PFET reads the p-code from memory and generates the actual 8080 object code. This code is written to a disk file. The last step in compilation is to link the generated object code to the run time package. This is done by simply using PIP to concatenate the run time package and the object file from PFET to produce an executable .COM file. The compiler (PPC) is written in Pascal, as is the p-code translator (PFET). The run time package is written in assembler.

Differences from "standard" Pascal

This section will detail the ways in which PPC deviates from standard Pascal as defined in "Pascal User Manual and Report", second ed., K. Jensen and N. Wirth.

Two additional reserved words have been defined: get and put. The following words are not now considered reserved, but are in standard Pascal, so they should be avoided: file, goto, in, label, nil, packed, set, and with.

The ASCII tab character is an acceptable white space character.

Comments are begun with the sequence "(*" and ended with "*)".

Identifiers may be very long, but only the first 8 are significant.

The data type Boolean is not supported. Relational and logical operators may be used only in if statements. The boolean constant identifiers true and false are not defined. The not operator is not implemented. These are the legal relational and logical operators: =, <>, <, <=, >=, >, and, and or.

The data type integer is available. Values must be in the range -32768 to 32767. There are no standard functions such as abs, sqr, trunc, etc. The constant maxint is not defined by the compiler. The type integer is identical to type word. The following operations are defined on integers: multiply, divide and truncate, add, and subtract.

Multiplication and division are presently implemented with repeated addition and subtraction (gag!). This makes the order of the operands critical. If one operand is likely to be less than the other, put the lesser operand on the left of the multiplication symbol for best speed. Dividing a large number by one takes a long time; dividing it by zero takes forever! (It's not that I'm not aware of the shifting methods of division and multiplication, it's just that I wanted something quick and didn't feel like looking up the good routines. I've never felt the need to replace these routines with the good ones.)

Also note that there is no integer negation. If you want negative one, write it as 0-1. The types char and real are not sup-

(continued next page)

ported. The type alfa can hold eight characters. Alfas can be assigned and compared just like integers (just don't try to do math on them!). All relational operators are defined using the ASCII collating sequence. Length can't enter into the comparison because alfas are always eight characters long (it's up to you to supply padding). Alfas may be passed as parameters.

Since files are not supported, the program heading is not needed, and in fact, is not allowed. The first thing the compiler expects to see are the global constant declarations.

Goto statements are not supported; therefore label declarations are not needed and not permitted.

Constant declarations are pretty much the same as in regular Pascal, except that leading signs are not allowed and character constants can be only one character in length. A minor extension is that I put in limited compile time constant expressions to make coding the translator easier. See the syntax graphs to see where these can be used.

Variable declarations have the restriction that the type must be a type identifier and may not be a complex type.

In this implementation, functions can return only integer values. This makes it unnecessary (and illegal) to give a function return type in the function declaration.

The case statement is limited in that it cannot accept multiple case labels on the same statement. On the other hand, it has been extended to allow an else statement which is executed when none of the case labels match the expression value. See the syntax graphs in PASYNTAX.DOC for the syntax.

Single dimensional arrays of integers and alfas (the two "built-in" types) are allowed. You can also declare arrays of subrange or enumerated types, but these are treated as arrays of integers and take the same amount of storage. Of course, arrays of arrays are not allowed, as that would be more than one dimension.

If a simple alfa variable appears with a subscript after it, it is treated as though it were an array of integers. This fact can be used to get at the individual characters of an alfa variable. For example, if "a" is a simple (not an array) alfa variable, then a[0] refers to the first two characters. The least significant eight bits would contain the first character and the most significant eight bits would contain the second character.

Record types are not allowed. Therefore, there is no need for a with statement.

There is no set type. (However, it shouldn't be too hard to implement a 64-bit set type using the p-instructions already around for alfa variables...)

There are no pointer types, and consequently, no new function.

There are no files and no read or write statements. All input and output is done with the put and get statements. These are only vaguely similar to the standard Pascal put and get. GET#0 gets one character from the input file. PUT#0 sends its output to the output file. PUT#1 sends its output unconditionally to the console. The arguments to the put statements consist of a series of expressions separated by commas.

If an expression evaluates to an alfa, all eight characters of the alfa are printed. Integer expressions followed by a pound sign ('#') will print the decimal value of the expression. If no pound sign follows the expression, the low eight bits of the expression are sent as one character. The input and output files mentioned above can be either disk files or console input and output. Which is used depends on what is typed on the command line following the compiled .com file when it is executed. If the first filename following the .com file name is blank or '*', then input characters are taken from the console. If it is the name of a disk file, then input comes from that disk file. A similar rule applies to the second filename following the command and the destination of the output characters.

Var parameters are different in that if one parameter to a procedure is to be var, then all parameters must be var parameters. This is a silly restriction that should be easily removed by any talented compiler hacker. There is a also a small kludge to make the compiler's job easier; the word var must appear in the call to all procedures with var parameters, as well as in the declaration. This is very easy to forget and a real nuisance at times. Somebody please fix.

It is possible to forward declare procedures and functions, but as with var parameters, there is a minor syntactic kludge to make the compiler's life easier. The forward part is handled in the normal way except that you don't give the parameter list (the compiler never checks procedure calls against their declarations anyway!). When you actually want to declare the procedure, use the form procedure:

```
foo(<real parameter list>);
backward;
```

This gives the compiler a hint it can't miss that this procedure was forward declared earlier!

PPC.PAS Pascal Source code for the compiler.

PSTACK.DOC Documentation on the stack operations of the run-time P-machine. The runtime stack is kept on the 8080 machine stack.

REGEN.DOC Notes on how to modify and recompile the compiler. When reassembling the runtime package, do not use LOAD to create RTP.COM. Instead, you must use a debugger and do the following:

1) Assemble RTP.ASM to produce RTP.HEX. Make note of the final code address printed by the assembler. RTP.COM should go up to this address minus 1.

2) Fire up your favorite debugger (DDT will do).
3) Fill memory with 0's. 100h - 1000h should do.
4) Now you can read in RTP.HEX, starting at 100h.
5) Boot back to the CCP.
6) Save memory up to one byte below the final code address printed by the assembler. For instance if 0600 was last address, type

"SAVE 5 RTP.COM".

This procedure must be followed so that PIP can be used to concatenate the runtime package and the object code produced by the compiler. It will also make your life a lot easier when using COMPARE.COM to compare parents and children (should you ever try and extend the compiler).

If you make changes to PPC.PAS or PFET.PAS, you'll want to be sure that the new compiler is capable of compiling itself. In genetics, this would be like making sure that your children are not sterile. The file validate.sub should help make sure you don't have sterile children. It uses a "know fertile" compiler (PPC.COM, PFET.COM) to compile the new PPC.PAS and PFET.PAS. The resulting compiler is then used to compile PPC.PAS and PFET.PAS again. The results of this second compilation are compared to the results of the first. If they match, it is safe to erase the "known fertile" compiler because you now know that you have a compiler which can reproduce itself. If they miscompare, you'd better find out why and fix it before erasing the parents. You should also note that this test only guarantees that you'll be able to continue to use the compiler to compile itself. It does N-O-T guarantee that you've got a fully functional compiler.

After making any changes to the compiler, you'll probably want to make sure that you can still compile and execute TESTER.PAS. This test doesn't test all functions of the compiler either, but passing TESTER is good sign that you haven't broken anything major. By the way, it is normal to get a few type mismatch errors while compiling tester. A new version of the compiler which is smarter about type checking would prevent these messages.

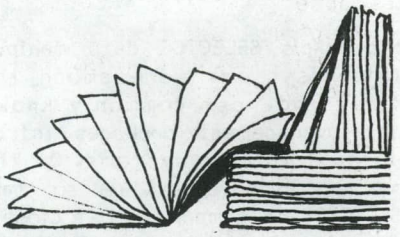
RTP.ASM Run Time Package Source code file.

RTP.COM Run Time Package Object code file.

STIRLING.PAS Sample program to generate Stirling numbers.

TESTER.PAS Sample program.

VALIDATE.SUB Submit file to verify that your computer is "fertile."



The Software Evaluation Group: SELECTOR IV

Authorship of package:
Micro-Ap Inc.
9807 Davona Dr.
San Ramon, Calif. 94583

Introduction

Micro-Ap's SELECTOR data manipulation systems were among the first and most commonly known data management packages introduced for CP/M systems. Of the systems we have reviewed so far, SELECTOR IV compares most closely to the Configurable Business System (CBS) by Dynamic Microprocessors Inc. SELECTOR is appropriately called an Application Development System in the Micro AP ads, and is listed under General Purpose Applications (along with CBS) in the latest Lifeboat ad.

The distinction between General Purpose Applications and true Data Management Systems, such as Condor or dBASE II, is an important one for anyone considering the purchase of either kind of package. The single most important difference is that the General Purpose Applications are designed for a situation where the application developer sets up a group of fixed procedures for users. These fixed procedures are generally linked to a menu, so that a user can set each in motion by specifying a number or letter on the menu. This means that all procedures must be anticipated by the application developer, because users are not expected to know how to carry out a new task--such as printing a report that isn't pre-defined. For this situation, a General Purpose Application package may be perfectly adequate.

In the case of a Data Management System, the goal is to gain a substantial degree of flexibility. The package should allow users to formulate new reports and answers to queries without calling in a specialist (such as the application developer) and without previously anticipating the exact information needs. The system should also allow users to reformat data when requirements change. The advanced capabilities and added flexibility of a true DMS require more sophistication on the part of the day-to-day user in order to be useful at all. Because of these differences in their goals, General Purpose Applications and Data Management Systems are suited to different individuals.

The Data Model

Like CBS, SELECTOR does not really conform to a well-defined data model, but since SELECTOR can't really represent too many relationships, it can be called a hierarchical data model. The system supports alphanumeric, numeric (with up to 14 significant digits), and dates (in either yymmdd, yyyymmdd, ddmmyy, or ddmmyyyy format) as its data types. Any field can also be designated as a key field (either unique or non-unique) and the indices generated from the key values can be used to speed searching and updating; they can also connect records from different data sets during report generation and batch updating. The ability to set up several indices on one data set can be very useful when accessing large data sets in several different orders.

SELECTOR requires the use of an indexing step to update the indices to a data set when records are added, deleted, or when key fields changed. The system gave strange results when a non-current index was used. It also allowed the creation of several identical keys in a field which had been defined as a unique key. (This happened only while editing in Recall/Update mode. Uniqueness was enforced in entry mode.) The automatic index maintenance of CBS was much smoother.

Documentation and Menu Organization

The preliminary SELECTOR manual is designed around the SELECTOR screen menu system. It is a fixed tree of menus and is not user adjustable like the CBS menus are. The menus are easy to follow and make the system fairly simple to learn. The chapters covering the menu-selected operations begin with two sections explaining the module discussed in the chapter and how to get to the module through the menu system. At the beginning of each chapter the user is also told what is required before using the module and how to leave the module. The manual we received earlier this year was marked "Preliminary" on every page and dated 27 July 1980. While we were preparing this review a new manual arrived.

The most notable thing about the new manual is the absence of the index promised in the preliminary manual. Promises, promises! Otherwise it is completely reorganized and re-written. It has sections added to help with the initial set up of the system and new applications. It also has a section on Digital Research's CP/M operating system, to help those who are not already familiar with it.

To our surprise it was easy to get SELECTOR to talk to both of our terminals (an IBM 3101 and a VT-52 compatible terminal). Getting started is straightforward. There are enough examples in each section to make the reading go easily and you can experiment as you read along. All the alternate choices at each point are discussed and you are referred to pertinent sections for each choice. Each entry required is well-explained.

Data Entry, Update, and Retrieval

All screen interactions with data are performed using a fixed format screen layout. The field names are displayed down the left side of the screen and the values are inserted in a fixed column (another similarity to CBS-see figure 1). There are two modes for dealing with data on the screen. In Entry mode the ope-

by Tim Berla & Steve Patchen

rator fills in each field and then types <ESC (the escape key) to add the record to the data file. In Recall mode the operator can search and move sequentially forward and backward through a data set based on any of the previously defined keys--provided the index used is completely up to date. Records which are displayed in Recall mode can be updated field by field.

SELECTOR performs some data checking and verification. Numbers entered are forced to fit the specification noted at definition time; there is little warning to the operator if s/he enters too many digits, or too many digits after the decimal point. For example, in a field formatted for a number with five digits before the decimal point and two digits after, an entry of 23456 was changed to 23456.00. Entering 1.2345 yields 1.23. The changed version is displayed, but there is no error message or bell sounded, so it is easy to miss an error (an error such as having the most significant digit removed). When the cursor gets to the last allowed position in a field, it remains in that position and any additional characters are typed over the last character. For example, typing "SELECTOR IV" in an eight-character field will yield "SelectoV". Because the only feedback from this kind of reformatting is the user's observation of the data as it is reformatted in the field, it is necessary to watch the screen quite a bit while typing, at least until you know what to expect.

Report Generation

The most impressive features of SELECTOR are its report generators. There are two distinct report packages. The first does tabular reports and mailing labels. These labels can be eight across; up to 99 can be printed for each record. The second report generator can fit file information onto a pre-printed form. It allows you to specify the exact row and column on which to print information. Records from different files can be merged into the report. The report generators permit a field in one file to retrieve a record from another file. This automatic retrieval process can continue from one file to the next, allowing a total of six files to be used at once. However, only one such record can be accessed this way for each secondary file. This is a very distinct limitation on the usefulness of these report generators. In many situations, it is desirable to print a report which has certain types of information printed once per page, and other types of information printed many times per page. A good example of this type of report is an invoice. Typically the name and address of the invoiced party, the date, and the invoice number are printed once on each invoice page, while the elements of the line items (description, cost, etc.) are printed many times. Neither of SELECTOR's report generators can handle this type of invoice. (It should be noted that not one of the systems we have reviewed so far is capable of producing such a report without the writing of an application program.)

Data Conversion

There are two data conversion modules. One allows conversion between SELECTOR III-C2 and SELECTOR IV data files and between any CBASIC and SELECTOR IV data files. The other conversion program allows you to redefine a file and change the fields by copying the old version into a new file with a new definition. You can also split one file into two new files, with each record of the original file split into two smaller records. An additional feature of this package is its ability to load a particular field in selected records with a new value.

Batch Updating

SELECTOR offers a Batch Update/Delete option that is very useful for systems where batches of information (such as payments or orders or any kind of transactions) are entered into a file and then processed to update information in other files (such as inventories or balances). The user must employ the Batch Update Enter module. At this point, the manual advises one to plan the batch update on paper before trying to enter it into the computer. The first step is to specify the fields that will be used to get records from the additional "Satellite" files (figure 2). Multiple records are fetched for report generation in the same way. Next, one must specify up to 80 "steps" for SELECTOR to follow for each record in the main file. These steps actually constitute a program inside of SELECTOR for carrying out a complex updating function. The steps are carried out through specifications given describing which fields from which files are to be used. For example, to specify the third field in the second file, the user types /3 to the appropriate prompt. SELECTOR then echoes that as "FIELD NAME" of "FILE NAME" (as in QUANTITY of TRANSACT). The steps in figure 3 (taken from the SELECTOR manual) could be paraphrased as, "For all transactions with quantity greater than , subtract the quantity from the inventory, multiply quantity times unit price to give total price, and add the total price to the customer's year-to-date balance. If the transaction date is after March 1, 1980 then delete it."

Figure 4 is an example I constructed to demonstrate a cumbersome but usable way to allow printed invoices. (The technique is neither debugged nor guaranteed.) The idea is to have a file called INVOICE with repeating fields into which the lines of the invoice are stored. The INVOICE file should have one record for each customer to be invoiced. Step 1 updates the

(continued next page)

count of the lines used on the invoice so far. (It should be set to zero before this procedure is used.) Steps 2-6 will put the first transaction for this invoice into the fields PART NUMBER1, DESCRIPTION1, QUANTITY1, PRICE1, and TOTAL PRICE1. Lines 7-11 repeat the same functions only for the second transaction on this invoice. The five steps must be duplicated for as many items as are anticipated on any given invoice. Since the total number of steps allowed in a BATCH UPDATE is 80, this technique is limited to about 14 or 15 lines per invoice. Its would also be possible to add some steps to produce totals for the whole invoice. The INVOICE file could then be sent through a page report to generate the actual invoices.

The programs can get pretty hairy and are somewhat difficult to edit, so it's necessary to write everything out ahead of time. If you make it through the many somewhat obscure questions required to create a Batch Update/Delete, this method can be a powerful tool in a turnkey application package. It does not solve the problem of permitting invoice information to update an inventory. The restriction permitting only one secondary record for each primary record processed makes this impossible.

Review Summary

Good Points

SELECTOR IV is a well-documented package that worked smoothly and without errors. Its powerful Batch Update/Delete, Line Report, and Page Report modules allow an application designer to handle many important business processes without writing application programs.

Bad Points

Because SELECTOR does not use any particular data model, it would be difficult to utilize for relatively complex problems. The definition modules are hard to comprehend, requiring most definition steps to be planned ahead of time on paper. SELECTOR lacks the flexibility to handle ad hoc queries or reports.

Recommendations and potential application:

If you are already using an earlier version of SELECTOR you might benefit from the page report generator or other improvements. Likewise, if you are using CBASIC data files and require a good report generator you might find this package useful. Unless the reporting capabilities are the most critical and the data manipulation requirements are simple the novice may find SELECTOR IV too unwieldy.

```
      Page 1 of 1          Left ## is field number          11
Fields Enter:<or,> or.to go back or fwd 1 item, ^F to display edit-menu

                          ESC to write record
 ## Name          Data

 1   EMP#          _____
 2   FIRST-NAME    ________________
 3   LAST-NAME     ________________
 4   STREET-ADDR   ____________________________
 5   CITY          ________________
 6   STATE         __
 7   ZIP           _____
 8   PHONE         ___________
 9   SOC-SEC#      _________
 10  CITIZENSHIP   ____________________________
 11  EDUC-LEVEL    __
```

[Figure 1]

```
                        BATCH UPDATE/DELETE
                    Update Files - Transaction is #1
              Files are: A:TRANSACT A:CUSTOMER A:INVNTORY
      For each file to be update, enter the calling File#/Field#,
                  and the to-be-called File#/KEYfield#.
              Enter the Call# to edit, or 0 to quit . . . 1
 Enter Calling File# _ Calling Field# _ Called File# _ Called Field# _

                             Batch Update Calls
     Call# Using: FileName - FieldName, Call: FileName - FieldName
 1:          1 TRANSACT    1 CUSTOMER #       2 CUSTOMER    9 CUSTOMER
 2:          1 TRANSACT    2 PART NUMBER      3 INVNTORY    1 PART NUMBER
```

[Figure 2]

```
                        BATCH UPDATE/DELETE
                    Update Files - Transaction is #1
          Files are: A:TRANSACT A:CUSTOMER A:INVNTORY
   Enter: 'SKIP', <cr>, or 'DELETE(#),<CR>, or DEST.
File#/Field# _<RET>

1    IF QUANTITY of TRANSACT EQ 0 then . . .
 SKIP
2    IF DATE of TRANSACT GT 800301 then . . .
 DELETE 1
3
 TOTAL PRICE of TRANSACT=[QUANTITY of TRANSACT]*[SELLING EACH of INVNTORY]
4
 YEAR-TO-DATE of CUSTOMER=[YEAR-TO-DATE of CUSTOMER]+[TOTAL PRICE of TRANSACT
5
 ON-HAND of INVNTORY=[ON-HAND of INVNTORY]-[QUANTITY of TRANSACT]
```

[Figure 3]

```
1
  TRAX COUNT of INVOICE=[TRAX COUNT of INVOICE]+1
2  IF TRAX COUNT of INVOICE=1 then . . .
  QUANT1 of INVOICE=[QUANTITY of TRANSACT]
3  IF TRAX COUNT of INVOICE=1 then . . .
  PART NUMBER1 of INVOICE=[PART NUMBER of TRANSACT]
4  IF TRAX COUNT of INVOICE=1 then . . .
  DESCRIP1 of INVOICE=[DESCRIPTION of INVNTORY]
5  IF TRAX COUNT of INVOICE=1 then . . .
  PRICE1 of INVOICE=[PRICE of INVNTORY]
6  IF TRAX COUNT of INVOICE=1 then . . .
  TOTAL PRICE1 of INVOICE=[PRICE of INVNTORY]*[QUANTITY of TRANSACT]
7  IF TRAX COUNT of INVOICE=2 then . . .
  QUANT2 of INVOICE=[QUANTITY of TRANSACT]
8  IF TRAX COUNT of INVOICE=2 then . . .
  PART NUMBER2 of INVOICE=[PART NUMBER of TRANSACT]
9  IF TRAX COUNT of INVOICE=2 then . . .
  DESCRIP2 of INVOICE=[DESCRIPTION of INVNTORY]
10 IF TRAX COUNT of INVOICE=2 then . . .
  PRICE2 of INVOICE=[PRICE of INVNTORY]
11 IF TRAX COUNT of INVOICE=2 then . . .
  TOTAL PRICE2 of INVOICE=[PRICE of INVNTORY]*[QUANTITY of TRANSACT]
                          .
                          .
                          .
                          .
```

[Figure 4]

REFERENCE SECTION:

See the introductory articles for details on terminology and the evaluation format used. Lifelines, Volume 1 No.4 p.7; Volume 1 No.5 pp.4-6; Volume 1 No.6. pp.12; Volume 1 No.7 p.2:

Send suggestions for software evaluations and other correspondence to:

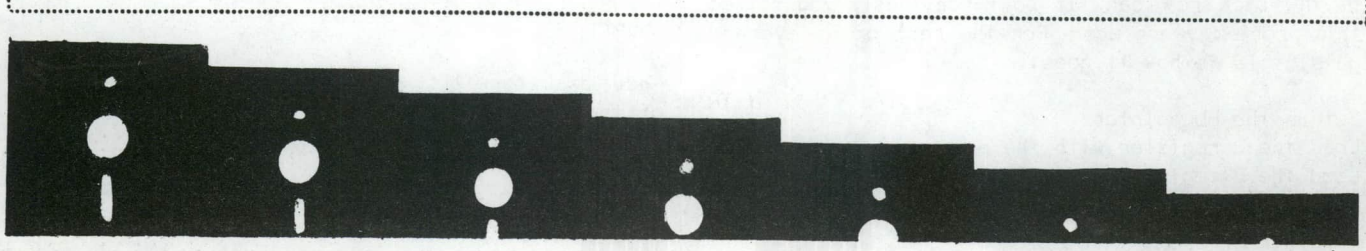
The Software Evaluation Group
c/o Lifelines
1651 Third Ave.
New York, NY 10028

Steve Patchen is a data engineer for Lab Data Systems in Pontiac, Michigan.

Tim Berla is a consultant specializing in the design and use of information systems for microcomputers.

TABLE I
Facts & Figures

Package or Version name: SELECTOR IV Ver 2.10	Record size & type limits: numeric fields are limited to 14 significant digits, character fields are limited to 255 characters record and file sizes are limited by your particular CP/M system.
Price: $440	
Systems available for: CP/M	
Required supporting software: CBASIC 2.05 OR HIGHER	Portability: Good across CP/M systems and between CBASIC file structures.
Memory requirements: 48K RECOMMENDED	
Disk capacity required: 2 drives-160k or more This would allow only a minimum system with small data files	User skill level required: within the capability of a novice. Because source files are provided, a programmer could make some adjustment for special requirements.
Utility programs provided: DATA CONVERSION In addition to SELECTOR III-C2 to IV conversion, the data conversion program can convert any CBASIC data type to SELECTOR IV format and any SELECTOR IV data file can be converted to a standard CBASIC sequential type data file.	System upgrade policy: upgrade from SELECTOR III-C2 $400 upgrade within 90 days of purchase of SELECTOR III-C2 $235 new revisions of SELECTOR IV $25



(continued next page)

TABLE II
Qualitative Factors

	Rating*
Documentation	
organization for learning	6
organization for reference	5
readability	5
includes all needed information	5
Ease of use	
initial start up	6
conversion of external data	4
application implementation	3
operator use	4
Error recovery	
from input error	3
restart from interruption	4
from data media damage	4
Support	
for initial start up	
for system improvement	

(I called the author, Bob Goodman to be sure I understood the limitations placed on invoicing and orders as discussed in the article. Although they plan to have multiple key accesses on the next release, they have no immediate plans to provide a means for the page report generator to be able to access more than one record per secondary file as referenced by the primary file record fields.)

* Ratings in this table will be in a 1-7 scale:
1 = clearly unacceptable for normal use
4 = good enough to serve for most situations
7 = excellent, powerful, or very easy depending on the category

TABLE III
Data Management Capabilities

A. Underlying Data Model:
1. Data Types-alphanumeric, numeric, date
2. Relationships-
Supports one-to-many relationships adequately via Batch Updating and Report generators with "Satellite" files.

B. Functions Provided:
1.a. Data dictionary maintenance-
maintained as part of each file-file definitions can also be copied for use in additional files
b. Data reorganization & conversion-
file re-definition is allowed and conversion between other CBASIC files is possible
2.a. Data entry and editing-
the entry format is fixed and data type checking is poorly handled
b. Report generation-
excellent tabular and pre-printed form reporting modules are provided
3.a. Data selection by predicate-
Good support of complex predicates-somewhat difficult to use.
b. Data joining & relating multiple data sets-available in a very restricted form
c. Calculations on data-
several of the modules have some calculation capability-Batch Update allows a calculation to be carried out on each record of a file
4.a. Data independent application interface-Files are CBASIC compatible, but no data independence.

Tips&Techniques

Here is the June winner of our tip contest, a device enabling one to use the Z80 block input or output instruction when the counter has a maximum of 255. (A buffer is often 256 or a multiple thereof.) The contributor is John Wilson of the Department of Systems Design, University of Waterloo, Waterloo, Ontario, Canada. He has this to say:

"Well, it can be done by fooling the Z80 instruction execution mechanism. You clever guys in the back row can sit down--obviously you solved this problem years ago. For the rest of us average guys, this is how it goes.

Set up the HL pointer
Load the C register with the port
Load the B register with ZERO
and the next statement is INIR (or whichever you need)

Let's set it down formally:

```
LD    HL,ADDR
LD    C,PORT
LD    B,0
INIR          ;will transfer 256 bytes
INIR          ;willtransfer next contiguous 256
INIR          ;...and so on                    bytes
```

The key is that the Z80 mechanism seems to carry out the first byte transfer BEFORE it decrements the counter, and it seems to decrement the counter BEFORE it checks for zero. So, by starting with a value of zero, we get the effect of a 256 byte count.

Additional benefits are that the terminating value of the B register is also zero, so successive block move instructions will move successive contiguous blocks without any further setup for the registers. Insofar as speed is concerned, this is all taking place in micro-code so you will find, if you check the cycles, that there is at least a 2:1 advantage over the equivalent user code with a 16-bit counter."

The Osborne Packages: Payroll by Martin McNiff

Of the three Osborne software systems available to you through The CP/M Users Group, the Payroll with Cost Accounting system is by far the most detailed and powerful. With this claim comes a straight caveat: you're going to have to change the software. The changes imposed by the government are perennial, especially report requirements for 941 and W2, along with state and local agencies. This system as written will process a bi-weekly California payroll...note emphasis on as written. Several firms are offering services to convert the state tax calculations, to allow variable pay periods and so forth.

On the positive side, this system maintains detailed records of every paycheck written in the current year, allows unlimited miscellaneous deduction and pay details which print on paychecks, prints a clear audit trail from entry to final payroll calculation, keeps track of job expenses; it details job activity by employee and employee activity by job, prints federal forms W2 and 941, prints an absentee report, insurance report, distributes job overheads, prints paychecks and check registers, prints detailed or summarized payroll journals.....but I am not a marketeer. Simply stated, this is not some pencil-necked, pantywaist piece of software. It does a lot, and one complaint I hear is "that Osborne stuff is too complex. I don't get it". Well, buster, here's your chance. Strap in and get ready for another whirlwind tour.

System Overview

The Payroll system has 25 application programs, and the Cost Accounting system contains 9. Each system is detachable from the other. There are 12 data files in the entire system; 10 are used for the Payroll system exclusively. Some files are abridged versions of others. Some files are created and updated automatically. The following files are worthy of note as they help describe the system's capability.

Employee Master File--This file is randomly accessed; i.e., each employee's number is its relative record position. This file contains the employee's name and address, birth date, tax statuses, pay rate and cumulative pay and deduction amounts for the current month, quarter and year. Up to 999 employee master file records can exist, but each record is 1150 bytes long. We've recommended a practical maximum of 100-300 employees, depending on the speed and capacity of your system.

Employee History File--Every paycheck written in the current year is detailed in this file. The file includes the payee, net pay, detailed tax deductions, total deductions and miscellaneous pay.

Miscellaneous Deductions/Pay File-- This is a nice file to have around! Aside from normal payroll deductions for taxes, many companies will pay commissions, bonuses, or other special pay or deduct for United Fund, garnishments, union dues or whatever. More than one record can exist for an employee. The following data describe when, how and how much to alter the pay amount in each miscellaneous deduction/pay record:

--Record type (misc. pay, deduction, additional state or federal withholding).

--Deduction priority (i.e., take this deduction first or last if more than one deduction exists).

--Taxable pay/nontaxable pay (denotes that this miscellaneous pay record should be taxed).

--Frequency (this payroll only, every pay period, alternate pay periods, first or last pay period in the month).

--Ten-character description of pay or deduction, which prints on an employee's paycheck.

--Deduction rate, a percentage of net pay.

--Amount of miscellaneous pay or deduction.

Transaction Files: Entry, Summary and Job Posting-- The entry file creates the summary and job posting files. Pay data are stored here for every employee. Employees can have more than one transaction record per pay period. The pay type--regular hourly or salaried pay, overtime pay, holiday pay (similar to overtime), vacation pay (which checks remaining vacation time for the employee), piecework and comp time are stored. An employee's time on a specific job can also be recorded, as input to the Cost Accounting system. Just before calculating pay, the Payroll system splits the entry file into the summary file (an abridged version of the entry file) and the job posting file. The job posting file accumulates records until you want to run any cost accounting programs.

Tax Files-- Two files--one for Federal, one for State-- store the tax tables used to calculate withholding and other government deductions. The Federal tax file contains rates and cutoffs for withholding, FICA and unemployment taxes as in IRS circular E. The state tax file contains California tax information. California payroll is the most tortuous payroll going, and it will need to be changed for other states. Additionally, the state likes to change its calculation methods to keep us software types on our toes. These files, and the associated software used to calculate withholding, are somewhat volatile. They simply can't be generalized, or be correct forever.

Job File-- This file contains a "header" record for each job, and a "detail" record for each employee working on the job. This cost accounting system uses labor costs only at present. Allowing for future expansion to include material costs and budgeting, the Osborne Account Payable and Accounts Receivable systems have reserved space and software for entering job postings, but no

(continued next page)

software is out as yet to interface to the Cost Accounting system. Job numbers can be up to six digits long. The first five digits contain the job number, and the last digit contains the task number. A task is actually treated as a separate job, but one report will total all tasks together for a job.

To illustrate, job 120100 might be "McGillicutty Contract--Negotiation". Job 120101 could be "McGillicutty Contract--Word Processing", job 120102 "McGillicutty Contract--Travel and Research", and so on. Each is a separate task for a job--in this case, the McGillicutty contract. More about tasks later.

Month-to-date and job-to-date totals are stored for each different job number. Hours, straight cost, and a special cost (depending on the type of pay for an employee), are stored, as well as payroll overhead, general overhead, personnel overhead and other overhead costs.

The general processing cycle for the Payroll system is flexible, but it normally operates as follows:

--Enter pay transactions using the Payroll Transaction Entry program.

--Print the transactions using the Transaction Report; change any errors with Transaction File Maintenance program.

--Run the Transaction Summary program. This appends concentrated transaction entry data to the Summary file, and posts Cost Accounting data to the Job Posting file.

--Enter, change or delete any miscellaneous pay or special deductions using Deduction File Maintenance. Print out the miscellaneous pay/deduction data on the Deduction Report.

--Run the Payroll Accumulate program, which assigns gross pay and any miscellaneous pay to employees who have pay transactions.

--Run Payroll Calculate program, which deducts payroll taxes from gross pay and updates the Employee Master file with current, quarter-to-date and year-to-date running pay totals.

--Run Deduction Calculate, which takes out any miscellaneous deductions.

--Print a Payroll Journal report to print all Employee Master file data after the payroll calculation phase.

--Print checks using Check Writer program. Miscellaneous pay and deductions are itemized on the check stubs.

--Print a check register.

--Reset the Miscellaneous Pay/Deduction file with the Deduction Reset program.

--Run the Payroll History Update, which saves check details for the current year for every employee paid.

--If the Cost Accounting programs are used, run the Job Posting Update program which merges payroll data into the Job file.

This is the general cycle. On an as-needed basis, the following programs are run:

--Job File Maintenance: Create, alter or delete jobs.

--Costing Report: Print a detailed cost accounting report by job number.

--Employee Activity Report: Print what jobs every employee is working on, organized by employee number.

--941 and W2: Print Federal forms.

--Overhead Distribution: Calculate and distribute job costs for personnel, payroll and other overhead.

--State and Federal Tax File Maintenance: Change tax tables as needed.

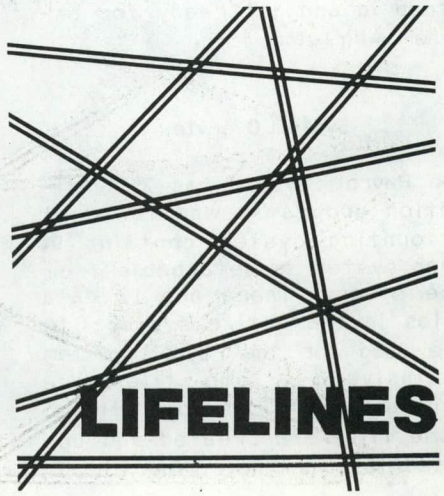
--Payroll History Report: Print check details for every employee, with employee totals for the current quarter and year. If check totals do not agree with the Employee Master File's totals, the report notifies the reader of a discrepancy.

In this series of articles, I have really tried to keep from hawking books. If you want to know more about the system, you should really buy the book Payroll With Cost Accounting--CBASIC from Osborne/McGraw-Hill. It's hard to summarize 222 pages of a large-format book(not including 142 pages of source listings), 35 programs and roughly a dozen data files in one article. Chapter Six of the book will explain what programs you have to change in order to accomodate other state payrolls, variable pay periods, other hardware such as CRTs, etc., etc. The software itself won't do that.

To paraphrase Picasso (I think), someone does it first and then someone else does it pretty. You have heard (and will, no doubt, for some time hear) about the difficulties people have in implementing some Osborne software. These problems will result if you are a basement bit-basher, but sometimes they can result from the software. We publish errata for the software and we will send copies to Lifelines as they come up.

We at Osborne have maintained a list of reliable companies who take the software, soup it up here and there and re-sell it. They offer support to end users, and the prices they charge are minimal because they didn't have to write the software from scratch. We don't pay people to do this, and they don't pay us. They sell software and make money doing that. We sell books and make money doing that. You don't have to pay through the snoot for applications software, and you wind up with a better value.

I would like to receive your comments. Send them to Martin McNiff, c/o Osborne/McGraw-Hill, 630 Bancroft Way, Berkeley CA 94710.



(continued from page 1)

Christensen. John Burns of Balcones Computer Corp spoke briefly about user oriented programming techniques. John's company, which authored a new CP/M accounting package called "The Boss"(tm), has developed some interesting programming techniques for the CP/M environment. I predict that they will be a major influence in future business software.

Another rumored feature of CP/M 3 has surfaced. A success/failure type program status will be returned to the operating system via a BDOS call. This in conjunction with a new improved Submit will allow Unix-like command sequences to be processed. Digital Research wishes to clarify the status of CP/NET. They consider CP/NET to be "finished" and not, as I said last month, a "back burner" project. (I wonder which is worse.) CP/NET86 is however, a true "back burner" project.

On the hardware side, there were new computers, new printers, new hard disks, Sony's 3 inch floppy, and much more. I'll try to cover the highlights next month. You're not going to believe how many big brand names will be on CP/M computers by this time next year. See you next month.

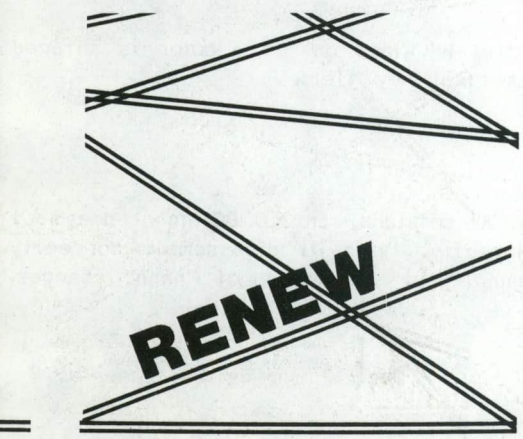
Harris Landgarten

p.s.
Here are some last minute bugs which came my way post-NCC:

PL/I-80 Version 1.3 has a bug in the WRITE FROM statement. In the KEY FROM(k) part of the statement, where k is the record number of the record to write, k cannot be replaced by a function call as in KEY FROM(GETADDR(X)). Currently, the only solution is to assign the k parameter before executing the WRITE FROM statement.

When using the Microsoft BASIC Compiler, the DATA statement will generate bad code if compilation is attempted with the /C switch.

For reasons unknown, CP/NET will not work with Microsoft BASIC random files.



OOPS

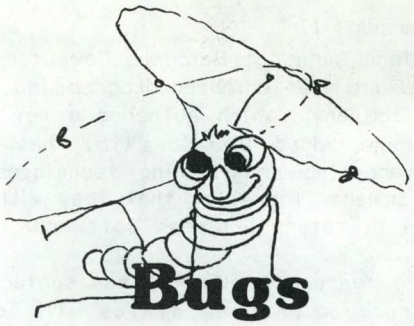
Two errors were noted in the May issue, Volume I, No. 12. They are as follows:

Under Tips & Techniques, page 18, column two, there was an error in the twelfth line from the bottom of the FN.SIZE listing. It should read, "IF FCBADR%<0 THEN POKE FCBADR%+40,(FCBADR% XOR OFFFFH)/256 XOR OFFH".

In the article SBASIC, Part 3, on page 10, the fifth line of the second column is incorrect. The copyright symbol (©) should be replaced with a "^".

Remember..

If your subscription began in July with our second issue, it's time to renew. In fact it's <u>past</u> time to renew; if you don't send in your renewal form right away you'll miss next month's issue. If you've lost your subscription form, send us one of your mailing labels, or a facsimile of it; this will help us to more efficiently update your subscription.



Despool
Version 1.1A
This version, as sold by Lifeboat Associates, is actually Digital Research's Version 2.0.

Pascal MT+
This product does not run on Apple, due to a problem in CP/M itself. Programs compile, but they do not link.

PMATE
Version 2.06
When Lifeboat's CP/M version 2.24A is used, the 'CLEAR TO END OF LINE' function should not be added to the ADM3A.CNF file (23). If the CL-EOL function is employed, PMATE will hang after the first character is typed. There are two exceptions to this problem: TRS-80 CP/M 2.24A implements the function correctly; PMATE doesn't crash if the TVI terminal is used.

The '/' key should shift (per page I-3 of the manual); it does not.

^U and ^J advance 21 lines, not the 6 lines stated on page II-3 of the manual.

Postmaster
Version 3.3
The manual states that Postmaster has the 'H' (HELP) command in its PMTEXT module; however, it does not.

Selector IV
Version 2.12
The index file is wiped out if a colon is entered into the numerical key field.

ZSID
Version 1.4
Under ZSID "A" command, an "LD BC,(nnnn) does not assemble correctly. It will disassemble correctly as LD SP,(nnnn)--but the value of "nnnn" changes. Example:

```
A300
0300      LD    BC,(0FFF)
#0300
 0300     LD    SP,(0FFF)
```

New Versions

BDS C Compiler
Version 1.44
Listed below are the bug fixes in this update:

1-In version 1.43A the character sequence \\ at the end of a quoted string caused the preprocessor in CC1 to stop stripping comments. This bug has been fixed.

2-The "qsort" function didn't work when the data array being sorted was larger than 32 bytes. This has been corrected.

3-When executed the "rename" library function would zero out the first three bytes of code following it; that is, the first jump instruction of the next function in memory would be destroyed. This bug has been corrected.

4-In version 1.43 "ccl" stopped stripping comments when it encountered the double quote character in single quotes (""'); this occurred while reading in the source code from disk. The problem has been rectified.

5-When the name of a function and the type information for its definition were placed on separate lines, the compiler would lose a line of code. After that point errors would be reported with incorrect line numbers. This has been fixed.

These new features have been added:

1-The "setfcb" function requires that the buffer which holds the resulting "fcb" be at least 36 bytes long.

2-A new library function called "execv" has been added (source in DEFF2.ASM); it permits chaining to another COM file with a variable number of command line parameters.

3-An new package permits directed I/O and pipes. DIO.C and DIO.H allow direct input, directed output, and pipes on the command lines.

4-A "standard error" buffered I/O stream number now is recognized by the "putc" buffered output function. An "iobuf" value of 4 causes the character given to be written to the CP/M console output; an "iobuf" value of 1 caused it to be written to the standard output.

5-String constants may now contain zero bytes within them. This makes it easier to initialize a table of strings of the same size.

6-CC1, CC2, and CLINK may be aborted during execution, with a ctl. C.

7-A new "-f" CLINK option, when specified before the name of a CRL file to be searched, forces all functions in that file to be loaded into the linkage.

Datebook
Version 2.03
The new features to be found in this version are as follows:

1-Nine groups of three people can now be served.

2-Storage is more efficient, permitting a longer range to fit on a disk drive.

3-A conference can be scheduled among any subset of the twenty people served.

4-Disk access and appointment displays are faster now.

5-Case is disregarded when looking for a person's appointment.

6-Individual appointment schedules and schedules for any range of days can now be printed.

7-Data entry now permits backtabbing through data fields. Dates can be entered with specifications like "+90d", for "90 days later".

8-For greater security, appointments can be logged as they are arranged.

9-Terminal selection is in a separate CONFIG program.

10-LEARNHOW illustrates data entry.

11-The installation program can be re-run, allowing the addition of new groups or name changes.

A bug occurred in conference scheduling; it has been rectified. If the user removed the names of some conference attendees and changed the meeting time, the S(chedule) option would then disappear and a line of dots would appear.

Magic Wand
Version 1.11
Problems with paragraph indentation, spooling, tabbing, superscripts and subscripts have been remedied; improvement has been made in the conditional hyphen function. A problem with the "start printing at specified page" function has been corrected. The print line length is now 255 columns.

These changes in the User's Manual have been made:

1-On page 29, this Section I should follow Section H: "User of the Tab Key function should be avoided when creating text to be justified. The hard space character or the Tab command should be used instead."

2-On page 109, sentence 11 in the second column should be replaced with this passage:"To define a fixed data file, type FILE Fn, where n is the number of character in each record but may not exceed 128."

In the Supplemental User's Manual the first sentence of Section 3 should be: "The speed at which text is displayed with SCREEN ON may be selected by typing a number from 9 (the slowest speed) to 0 (the fastest). (e.g. slow =9876543210=fast)".

Micropolis Microdisk
Version 1.92
The new version of this hard disk module corrects a bug which caused "format" (in some cases) not to initialize the directory platters properly.

Selector IV
Version 2.13A
Using an editor you can make the following alteration in REPORT.BAS, so that Line Report Definition report definition names and titles can be correctly printed.

```
1000 PRINT FN.CLR$
                              <--deleted
     KFILL12$=""
```

(continued next page)

```
1710 IF SERR% THEN 900
     PRINT FN.CTR$(ERR LINE%,"Press P to print");
     DUMMY%=FN.GET.A.MATCH%("P")
     LPRINTER
     FOR I%=1 TO 3
          PRINT
     NEXT
     PRINT USING "& LINE REPORT DEFINITION as of &";\
               REPORT.FILE$,DISPLAY.DATE$ <--(FILE$ vs TILE$)
```

This change to REP1EX.BAS ensures that a space will be printed between fields printed on the same line of labels after the first field is filled to its length limit.

```
DEF  FN.STRIP$
     B$=CFIELD$(F.BIN%,J%)
     IF MATCH ("N",I.TYPE$(F.BIN%,J%),1) THEN RETURN
     M1%=MATCH("  ",B$,""
     IF LEN(B$) AND RIGHT$(B$,1)<>" " THEN B$=B$+" " <--(added)
     RETURN
```

In SELECT.BAS, make this change:

```
26000      PRINT FN.ERR$("Merging...",0)
           MAJOR.CYCLES%=FN.IMOD%((CYCLE%-2),7)+1 REM <--(chng)
```

In BATCH.BAS, on the 14th and 245th physical lines of the source code, change the variable DEF$ to DF$.

Then recompile these programs to create new INT files.

Patch For RAID

This patch allows RAID to support more than two drives. (As issued, it knows only A and B, which has proven inconvenient for some hard disk users.)

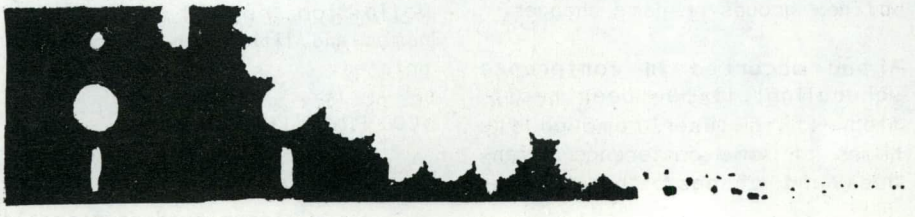
For RAID Version 4.7.3:

RAID.COM

27A4 02--> # drives desired

For the Floating Point version:

```
A>RAID
GF;RAID.COM <cr>
O;27A4 <cr>
dd <cr>
DA;100;3400 <cr>
PF;RAIDX.COM
BY
```



The programs listed below are available from their authors, software distributors, and computer stores.

JRT Pascal
by JRT Systems, Inc.

This product is a pseudo-code compiler and run-time package. It is designed to be more compact than machine code compilers and therefore more efficient for a high level language such as Pascal. No macro assembly is necessary with this compiler.

Random access by relative byte address or record number is provided for input/output. Data can be stored either in ASCII or binary form.

A "dynamic storage" region contains pointer variables, external procedures and buffers. These components are fully accessible. Automatic storage compression and free list are maintained; a DISPOSE procedure allows released storage blocks to be placed on this list. Stored material (linked lists, binary/multiway trees, rings, etc.) may be created, modified, and deleted while in the "dynamic storage" region.

External procedures, designed for modular and structural programming, can be separately compiled and stored on disk in relocatable format; they are automatically loaded into "dynamic storage" when first referenced. If storage space becomes inadequate, the procedure used least recently will be purged. These external procedures can call themselves and each other. A COMMON area may be declared to store variables shared among the procedures. A main program and its external procedures are joined by a linker utility.

The floating point package permits fourteen digit precision in

binary coded decimal format. Strings (arrays of characters) and structured variables are supported, as is concatenation. The value returned by a function may be string, array, record, Boolean, set, integer, real, char, or pointer.

CASE statements may be full expressions; the optional ELSE clause is executed when no earlier conditions are met.

Assembler routines and the CP/M operating system may be called directly using the ASM procedure. Other functions permit direct interfacing with 8080/Z80 hardware ports. Hex integer constants can be used in source, in console, or in disk input; any variable can be converted to hex characters.

JRT Pascal requires a 56K system and one drive.

Microstat

by Ecosoft

This statistical testing package includes a Data Management System for manipulation of files. It is designed for academic and corporate research.

The Descriptive Statistics program calculates commonly used measures, using "correction vector" algorithms to handle large sums of squares which might result in overflow or inaccuracy. The Frequency Distributions program generates histograms and frequency distributions for quantitative and qualitative data sets. In addition, Microstat performs hypothesis (mean) tests, analyses of variances, scatterplots, regression analyses, correlation matrices, and time series analyses.

Ten non-parametric tests can be effected: the Wald-Wolfowitz Runs test, the Wilcoxon Rank-Sum test for two groups, the Kruskal-Wallis one way analysis of variance by ranks, the Kolmogorov-Smirnov goodness of fit and two group tests, the Wilcoxon signed-rank test, absolute normal scores, the Friedman test, and the Kendall coefficient of concordance, and sign-test.

Crosstab and Chi-Square statistics programs generate contingency tables and calculate chi-square statistics for them; a goodness of fit test calculates chi-square, given observed and expected values.

Factorial, permutations, and combinations programs calculate factorials with several methods and under varying conditions; values up to 49 are calculated directly. Factorials from 50 to 300 use an accumulation of logarithms. Microstat performs 1,000,000 factorial, utilizing Stirling's approximation with an extended precision multiplication routine.

The following discrete and continuous probability distributions are supported: binomial, hypergeometric, Poisson, exponential, normal, F distribution, student's t, and Chi-square.

In addition, this package performs hypothesis testing, allowing two proportions from independent groups, or proportion versus hypothesized value.

Microstat requires a 16 by 64 CRT, BASIC-80 (version 5.03 or higher), a 48K system, and CP/M or NorthStar DOS. Two drives are recommended for use of Microstat. The package is 205K in size; it may be split on two or three disks.

Mince

by Mark of the Unicorn

This product is a screen-oriented text editor. Features include: single-key commands, block moves, and global editing (string replacement). Editing takes place in one mode. The text appears on the screen as it will when printed.

More than one file can be edited at a time, and two files can be displayed on screen at once. Deleted words, lines and paragraphs are saved in a special buffer.

Mince requires a 48K system, a CRT with clear-screen and cursor addressing, and a 120K drive if a single drive is used; multiple drives of at least 60K each can be used alternatively.

Professional Time Accounting

by ASYST Design Services

This menu-driven package is intended for use by professional offices where billing is computed on the basis of time. Data is entered into Employee Time and Expenses and Client Billing files. These are sources for Work In Progress, Employee History, Job Detail, and Client Master files.

This package maintains billings, current charges, transfers, hours and dollars accumulated per quarter by each employee.

Professional Time Accounting can be used with an 8080 or Z80 processor. It requires a 48K system and a disk capacity of 241K; two drives are needed. Clear screen and addressable cursor functions are also required; CBASIC2 and CP/M are necessary.

New Publications

Basic Business Software

by E.G. Brooner

This book is designed both for businessmen in search of software and for programmers who wish to learn more about business software. It examines methods of writing such programs; in addition, it contains several complete programs.

144 pp; $9.95.

Computer Language Reference Guide

by Harry L. Helms, Jr.

BASIC, ALGOL, LISP, PASCAL, PL/1, COBOL, and FORTRAN are explained and compared in this handbook. Guidance is offered for those outside their "native" computer, who wish to understand a program in a "foreign" language. A keyword dictionary is included, as are reserved word lists for each language.

112 pp; $6.95

VERSION LIST

PRODUCT NAME	S	M	OS	P	MR	$	
Accounts Payable/Cybernetics	3.1		CP/M	Z80	64K	500	Needs RM/COBOL
Accounts Payable/Graham Dorian	1.11	1.11	CP/M	8080	48K	805/40	Needs CBASIC2
Accounts Payable/Structured Sys	1.3B		CP/M	8080	52K	840/40	
Accounts Payable/Peachtree	10-10-80		CP/M		48K	530/60	Needs BASIC-80 4.51
Accounts Receivable/Cybernetics	3.1		CP/M	Z80	64K	500	Needs RM/COBOL
Accounts Receivable/Graham Dorian	1.08	1.08	CP/M	8080	48K	805/40	Needs CBASIC2
Accounts Receivable/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
Accounts Receivable/Structured Sys	1.4C		CP/M	8080	56K	840/40	
Address Mngemt. Sys.	1.0		CP/M	8080		150	Requires 2 drives
ALGOL/60	4.8C		CP/M	8080	24K	250	
ANALYST	2.0		CP/M	8080	52K	250/20	Needs CBASIC2,QSORT/VSORT
APL/V80 Compiler	3.2		CP/M	Z80	48K	500	Needs APL terminal
Apartment Management	1.03	1.03	CP/M	8080		805/40	Needs CBASIC2
Automated Patient History	1.2		CP/M	8080	48K	175	
BASIC-80 Compiler	5.24	5.24	CP/M	8080	48K	360/35	
BASIC-80 Compiler		5.1	TRSDOS		64K	400/25	TRS-80 Model II only
BASIC-80 Interpreter	5.2	5.2	CP/M	8080	40K	335/35	W/Vers. 4.51,5.2
BASIC Utility Disk	2.0	2.0	CP/M	8080	48K	75	
BSTAM Communication System	4.5	4.5	CP/M	8080	32K	200	
BSTMS	1.2	1.2	CP/M	8080	24K	200	
BUG / uBUG Debuggers	2.03		CP/M	Z80		129/25	
*BDS C Compiler	1.44	1.44T	CP/M	8080	32K	150/30	
Whitesmiths' C Compiler	2.0		CP/M	8080	60K	630/30	
Cash Register	2.0	2.0	CP/M	8080		805/40	
CBASIC2	2.07P	2.17P	CP/M	8080	32K	125/20	
CBS Applications Builder	1.2		CP/M	8080	48K	395/40	CDOS version too
CIS COBOL Standard	4.3,1		CP/M	8080	48K	850/50	
CIS COBOL Compact	3.46	3.46	CP/M	8080	32K	650/50	
FORMS 1 CIS COBOL Form Generator	1.06	1.06	CP/M	8080		150/20	
FORMS 2 CIS COBOL Form Generator	1.1,6	1.16	CP/M	8080		200/20	
*COBOL-80 Compiler	4.01A	4.01A	CP/M	8080	48K	710/35	
COBOL-80 PLUS M/SORT	4.01		CP/M	8080	48K	845/65	
CONDOR	1.10		CP/M	Z80	48K	695/35	
CREAM (Real Estate Acct'ng)	2.3		CP/M	8080	64K	250	CBASIC needed
*DATASTAR Information Manager	1.101		CP/M	8080	48K	350/60	
*Datebook	2.03		CP/M	8080	48K	295/30	Needs 80x24 terminal
DESPOOL Print Spooler	2.0		CP/M	8080		80	
DISILOG Z80 Disassembler	4.0	4.0	CP/M	Z80		110	Zilog mnemonics
DISTEL Z80/8080 Disassembler	4.0		CP/M			110	
EDIT Text Editor	2.06		CP/M	Z80		129/25	
EDIT-80 Text Editor	2.0		CP/M	8080		99/25	
*ESQ-1	2.1A		CP/M	8080		1495/50	Needs CBASIC2
FABS	2.4A		CP/M	8080	32K	195/25	
FILETRAN		1.2	TRSDOS		32K	99/20	1-way TRS-80 Mod 1,TRSDOS to Mod 1 CP/M
FILETRAN		1.4	CP/M		32K	149/20	2-way TRS-80 Mod 1,TRSDOS & Mod 1 CP/M
FILETRAN	1.5		CP/M		32K	99/20	1-way TRS-80 Mod II,TRSDOS to Mod II CP/M
Financial Modeling System	2.0		CP/M		48K	300	
FORTRAN-80 Compiler	3.42	3.42	CP/M	8080	36K	435/35	
FORTRAN Package	3.38		TRSDOS			80/25	
*FPL	2.0	2.0	CP/M	8080	48K	695/30	
General Ledger/Cybernetics	1.3C		CP/M	Z80	48K	500	Needs RM/COBOL
General Ledger/Graham Dorian	1.09	1.09	CP/M	8080	48K	805/40	Needs CBASIC2
General Ledger/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
General Ledger/Structured Sys	1.4C		CP/M	8080	52K	840/40	
*General Ledger II/CPaids	1.1A		CP/M	8080	48K	450/30	Needs BASIC-80 4.51
GLECTOR Accounting System	2.0		CP/M	8080	56K	350/25	Use w/CBASIC2,Selector III C-2
HDBS	1.04		CP/M	+	52K	300	
IBM/CPM	1.1		CP/M	8080		175	
Inventory/Graham Dorian	1.0	1.0	CP/M	8080		555/40	Needs CBASIC2
Inventory/Peachtree	10-10-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
Inventory/Structured Sys	1.0C		CP/M	8080	52K	640/40	
Job Costing	2.02	2.02	CP/M	8080	48K	805/40	Needs CBASIC2
*JRT Pascal			CP/M	8080	56K	225/25	
LETTERIGHT Text Editor	1.1B		CP/M	8080	52K	200/25	
MAC	2.0		CP/M	8080	20K	120/25	
MACRO-80 Macro Assembler Package	3.40	3.40	CP/M	8080/Z80		159/25	
*Magic Wand	1.11		CP/M	8080	32K	395/40	
MAGSAM III	4.2		CP/M	8080	32K	145/25	For CBASIC/MBASIC
MAGSAM IV	1.1		CP/M	8080	32K	295/25	Needs CBASIC
MAILING ADDRESS Mail List System	12-2-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
MailMerge	2.26		CP/M	8080		150/25	Needs same version WordStar
Master Tax	1.0-80		CP/M	8080	48K	995/30	
MDBS	1.04		CP/M	+	48K	900/35	
MDBS-DRS	1.02		CP/M	+	52K	300	
MDBS-QRS	1.0		CP/M	+	52K	300	
MDBS-RTL	1.0		CP/M	+	52K	300	
MDBS-PKG			CP/M	+	52K	1500/60	W/all above MDBS products

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

VERSION LIST

PRODUCT NAME	S	M	OS	P	MR	$	
Microspell	4.1		CP/M	8080	48K	249	
Mini-Warehouse Mngmt. Sys.	5.5		CP/M	8080	48K	650	Needs CBASIC
MP/M Operating System	1.1		MP/M	8080	32K	300/50	
MSORT	4.01		CP/M	8080	48K	160/20	Needs COBOL-80
Mu LISP-80 Compiler	2.03		CP/M	8080	24K	210/25	
Mu SIMP / Mu MATH Package	2.03		CP/M	8080	48K	260/30	muMATH 80
NAD Mail List System	3.0D		CP/M	8080	48K	115/25	
Nevada COBOL	1.404	1.404	CP/M	8080	32K	149/25	
Order Entry w/Inventory/Cybernetics			CP/M	Z80		500	Needs RM/COBOL
*PAS-3 Medical	1.74		CP/M	8080	56K	995/25	Needs 132-col. printer
*PAS-3 Dental	1.62		CP/M	8080	56K	995/25	Needs 132-col. printer
PASM Assembler	1.02		CP/M	Z80		129/25	
Pascal/M	3.2		CP/M	8080	56K	175/25	Also for CDOS
PASCAL/MT Compiler	3.2		CP/M	8080	32K	250/30	
PASCAL/MT+	5.2		CP/M	8080	52K	500/30	For Z80 too
PASCAL/Z Compiler	3.3		CP/M	Z80	56K	395/25	
Payroll/Cybernetics, Inc.			CP/M	Z80		500	Needs RM/COBOL
Payroll/Peachtree	11-7-80		CP/M	8080	48K	530/60	Needs BASIC-80 4.51
Payroll/Structured Sys	1.0E		CP/M	8080	56K	840/40	
PL/I-80	1.3		CP/M	8080	48K	500	
PLINK Linking Loader	3.25P		CP/M	Z80		129/25	
*PMATE	2.06		CP/M	8080	32K	195	
POSTMASTER Mail List System	3.3	3.3	CP/M	8080	48K	150/20	Needs CBASIC
Property Manager	10-10-80		CP/M	8080	48K	925/60	Needs BASIC-80 4.51
Property Mngemt. Sys.	1.0		CP/M	8080	48K	650	Needs CBASIC
*QSORT Sort Program	2.0		CP/M	8080	48K	100	
Real Estate Acquisition Programs	2.1		CP/M	8080	56K	500	Needs CBASIC
Residential Prop. Mngemt. Sys.	1.0		CP/M	Z80	48K	650	
RM/COBOL Compiler	1.3C		CP/M	8080	48K	750	w/Cybernetics CP/M 2
RAID	4.7.3	4.7.3	CP/M	8080	28K	250/25	
RECLAIM Disk Verification Program	2.1		CP/M	8080		80	
*SBASIC	5.3h		CP/M	8080		295/35	
SELECTOR-III-C2 Data Manager	3.24		CP/M	8080	48K	295/25	Needs CBASIC
*SELECTOR-IV	2.13A		CP/M	8080	52K	550/35	Needs CBASIC
SID Symbolic Debugger	1.4		CP/M	8080		120/25	N/A-Superbr'n
SMAL/80 Programming System	3.0		CP/M	8080		75/25	For CP/M 1.x
Standard Tax	1.0		CP/M	8080	48K	495/30	
STATPAK	1.2	1.2	CP/M	8080		495/30	Needs BASIC-80 4.2 or above
STRING BIT FORTRAN Routines	1.02	1.02	CP/M	8080		75/25	
STRING/80 bit FORTRAN Routines	1.22		CP/M	8080		95/25	
STRING/80 bit Source	1.22		CP/M	8080		295	
SuperSort I Sort Package	1.5		CP/M	8080		225/40	Max. record=4096 bytes
*T/MAKER II Data Calculator	2.1		CP/M	8080	48K	275/25	
TEX Text Formatter	1.1		CP/M	8080	36K	105/15	
TEXTWRITER-III Text Formatter	3.6	3.6	CP/M	8080	32K	125/20	
TINY C Interpreter	800102C		CP/M	8080		105/50	
TINY C II Compiler	800201		CP/M	8080		250/50	
TRS-80 Customization Disk	1.2		CP/M	8080		75	
ULTRASORT II	3.1	3.1	CP/M	8080	48K	195/25	
VisiCalc	1.37		Apple	8080	32K	150	
WHATSIT? Data Manager	2.04		CP/M	8080	44K		
Wordindex	3.0		CP/M	8080	48K	195	Needs WordStar
WordMaster	1.07A		CP/M	8080	40K	145/40	
WordStar	2.26	2.20A	CP/M	8080	48K	445/60	
WordStar w/MailMerge Overlay	2.26		CP/M	8080	48K	575/85	
*XASM-05 Cross Assembler	1.04		CP/M	8080	48K	200/25	
*XASM-09 Cross Assembler	1.04		CP/M	8080	48K	200/25	
*XASM-51 Cross Assembler	1.07		CP/M	8080	48K	200/25	
*XASM-F8 Cross Assembler	1.02		CP/M	8080	48K	200/25	
*XASM-400 Cross Assembler	1.02		CP/M	8080	48K	200/25	
*XASM-18 Cross Assembler	1.40		CP/M	8080		200/25	
*XASM-48 Cross Assembler	1.60		CP/M	8080	48K	200/25	
*XASM-65 Cross Assembler	1.96		CP/M	8080	48K	200/25	
*XASM-68 Cross Assembler	1.99		CP/M	8080	48K	200/25	
XMACRO-86 Cross Assembler	3.40		CP/M	8080	48K	275/25	
XYBASIC Interpreter Extended	2.11		CP/M	8080		450/25	
XYBASIC Interpreter Extended CP/M	2.11		CP/M	8080		550/25	
XYBASIC Interpreter Extended COMP	2.0		CP/M	8080		450/25	
XYBASIC Interpreter Extended ROM	2.1		CP/M	8080		450/25	
XYBASIC Interpreter Integer	1.7		CP/M	8080		350/25	
XYBASIC Interpreter Integer COMP	2.0		CP/M	8080		350/25	
XYBASIC Interpreter Integer ROM	1.7		CP/M	8080		350/25	
Z80 Development Package	3.5		CP/M	Z80		130	N/A-Magnolia,Superbr'n,mod.CP/M
*ZDM Debugger	1.2		CP/M	Z80		45	For Micropolis,N'star,Apple,IBM 8"
*ZDMZ Debugger	2.0		CP/M	Z80		45	For Micropolis,N'star,Apple,IBM 8"
ZDT Z80 Debugger	1.41	1.41	CP/M	Z80		50	N/A-Superbr'n,mod.CP/M
ZSID Z80 Debugger	1.4		CP/M	Z80		130	N/A-Superbr'n,mod.CP/M

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

* Indicates a new version or new product

+ These products are available for Z80 or 8080, in the following host languages: BASCOM, COBOL-80, FORTRAN-80, PASCAL/M, PASCAL/Z, CIS COBOL, CBASIC, PL/1, and BASIC-80 5xx.

Operating Systems

Description	Version
CP/M for:	
Apple II w/Microsoft BASIC	2.0
Cromemco System 3 8"	1.4
Cromemco System 3 8"	2.2
Durango F-85	2.23
Heath H8 w/H17 Disk	1.43
H89 Heath/Zenith by Magnolia	2.2
ICOM 3812	1.41
ICOM 3712 w/Altair Console	1.41
ICOM 3712 w/IMSAI Console	1.41
ICOM Microfloppy (# 2411)	1.41
ICOM 4511/Pertec D3000 Hard Disk	2.22
Intel MDS Single Density	2.2
Intel MDS 800/230 Double Density	2.2
MITS Altair 3202 Disk	1.41
Micropolis Mod I - All Consoles	1.4X
Micropolis Mod II - All Consoles	1.4X
Micropolis Mod I	2.2
Micropolis Mod II	2.20A
Compal Micropolis Mod II	1.4
Exidy Sorcerer Micropolis Mod I	1.4X
Exidy Sorcerer Micropolis Mod II	1.4X
Vector MZ Micropolis Mod II	1.4X
Versatile 3B Micropolis Mod I	1.4X
Versatile 4 Micropolis Mod II	1.4X
North Star SD	1.4X
Mostek MDX STD Bus	2.2
Sol North Star SD	1.4
North Star SD IMSAI SIO Console	1.4
North Star SD MITS SIO Console	1.4
North Star DD	1.4
North Star SD	2.2
*North Star DD/QC w/Corvus	2.22
North Star DD/QD	2.22
Ohio Scientific C3	2.23
*Ohio Scientific C3-B	2.23
*Ohio Scientific C3-C'(Prime)	2.23
Processor Technology Helios II	1.41
by Lifeboat/TRS-80 5 1/4"(Mod I)	1.41
by Lifeboat/TRS-80 Mod II	2.24A
by Cybernetics/TRS-80 Mod II	2.25

OASIS for Altos,Bell Ctrls.,Billings, CA Computer Systems, Compucorp, Cromemco, Delta, Digital Microsys., Dynabyte, Godbout, GRI, Index Comp. Sys., IBC, Intertechnique, Kontron, Media Sys. Corp., Micromation Doubler, Morrow Thinker Toys, NNC Elect., Onyx, Quay, S.D.Sys.,Teletek, TRS-80 Mod.II,Vector Gr., Vorimex, Zilog

Hard Disk Modules

Description	Version
*Corvus Module	1.7
APPLE-Corvus Module	1.1
*KONAN Phoenix Drive	1.8
*Micropolis Microdisk	1.92
Pertec D3000/iCOM 4511	1.6
*Tarbell	1.5

A Plea From the U.K.

17/04/81

"May I use your excellent publication to make an appeal to the software producers of the World to consider that in the United Kingdom we expect a date to be in the format dd/mm/yy and not mm/dd/yy.

Some products allow the European user to specify that a comma is to be used, as a decimal point instead of a period, but most do not allow for the UK requirement to handle the date as dd/mm/yy."

G.M. Dearlove
Sutton Coldfield

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

NEW YORK N.Y.

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

FIRST CLASS MAIL