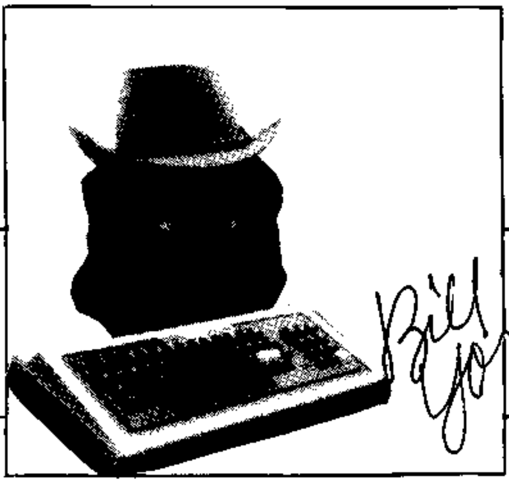
THE S-EIGHTY™

VOL. 1, NO. 4, APRIL 1980

PURE & SIMIAN



Bill York

I come to you not as a prophet............................

Expansion, growth, profit...these have become watchwords for the microcomputer industry, and they appear to be causing negative reaction and results in some areas of our industry. A company can indeed outgrow itself, overextend its capital in money and people, cut its promotional budget so as to skip a month here and there, and suddenly said company finds that it doesn't exist on a competitive scale. I don't think that many people anticipated the industry to soar as it has...no matter what some people say...as sales increased we increased our promotions, orders increased, we hired additional people to meet the workload and suddenly WE COULDN'T FIND ANY COMPUTERS!! We had finally reached the point where the manufacturer couldn't keep up with the demand, and the backorder became the order of the day. This started from the computer itself, all the way to I.C's.

With this issue, we have refused an advertiser who was offering a major hardware item at 8% over cost. We admit that the low price had a part in the decision, but upon checking with the company we did not find sufficient capitalization to cover repairs and product support which we know will cost more than the $25.00 markup...not to mention advertising costs and normal operating overhead. A deal that puts your supplier out of business is not our idea of a bargain.

A New England-based chain of computer stores is reported to be in dire straits. They started with one large store and grew to several branches, but have recently sent a group of employees packing, and are behind in payments to creditors.

There is much to be learned and earned from the micro industry, and I guess the only way is through the mistakes of others. Being overzealous (and who isn't when it comes to the almighty buck) can put your business in the red before you realize it. Good marketing strategy and support, good product support and competitive pricing are what it's all about. Once you go beyond the financial constraints of your business, you're headed for trouble. There are many informative publications to scan that are loaded with good financial and marketing information; some are Changing Times, Forbes, Consumer Reports, and many others.

RIP OFF! seems to be a popular part of the lingo of today...I'll bet George Carlin could do a couple of hours on it. We are very interested in the consumer aspect of our industry, and are considering a column devoted to exposing companies who appear to be fraudulent, and a column to praise companies who are doing business on a first-rate basis. The only way I can find out about the good companies, (I'll dig up the bad ones) is from your input. Let me know about your experiences with various manufacturers and vendors. If you have a good experience, let us know...if you've had a bad experience, let us know even faster so we can save someone else from the same debacle. We'll also provide you with hints on what recourse is available when you feel that you have been taken.

SAN FRANCISCO

I left my heart, money, appetite and a couple of unmentionables in San Francisco and never had a better time in my life! I am referring, of course, to the West Coast Computer Faire #5 which took place the 14th-16th of March. I have been an exhibitor in four of five shows and can quite honestly say that this year was far better than any in history. Almost 20,000 people filed through Brooks Hall and the Civic Auditorium and almost 20,000 people bought something along the way.

San Francisco is filled with magic for anyone who visits...from its fabulous restaurants to its beautiful *scenery...and what a place to have a computer faire.*

Most of you are familiar with George Blank, our Managing Editor here at SoftSide Publications. George had eight cases of a new publication which he thought the Faire attendees might like to see, so rather than sell them at cover price, George thought "Let's sell them at 5¢ each!". Well!...at the end of three days, we had $175.00 worth of nickles and the City of San Francisco is now sitting two inches higher in the bay. I once thought of George as insane...now I regard him as a good friend. In spite of the humor here, it perfectly describes the response of the visitors to booths as they made their way around the show.

Jim...you've done it again and my hat's off to you. A very nice job and a fabulous faire. See you next year...with a much larger crew.

CURRENT EVENTS

MITA (Microcomputer Industry Trade Association) seems to be off and running in good shape. They now have a spokesperson in Washington and are looking toward a MITA show and several other interesting and vital projects.

I normally come away from a show with at least one shocking bit of news, but this year the news seems to be much as expected...Radio Shack is falling way behind its competition. North Star*, Ohio Scientific*, Atari*, APPLE* and several others are rapidly passing Radio Shack in quality and support in all levels. Many companies that I know of...including ours...are considering dropping the TRS-80 before it's too late. Graphics, color and mainframe are taking their toll on the Model I from Fort Worth.

Spring is just around the corner! See you in May.

Cover photograph by Elaine Cheever

COVER CONTEST

Win a year's worth of SoftSide on Cassettes - details on page 19

REVIEW

MICRODOS
A Different Way To Do It

by John Culleton

The advertisement was enticing: "an advanced, easy-to-use operating system...works entirely with Level II BASIC commands". The product was produced by Percom, one of the better houses in our experience. So, a check was dispatched to Access Unlimited, Inc. and, rather quickly, the diskette and a slim instruction booklet arrived. On the diskette was a single monitor program which combines the essential features of an operating system and Disk BASIC, while occupying only 7K of memory. A unique and useful set of BASIC language utilities were also included — more about that later. With one very important exception, the software supplied duplicates the major functions of Radio Shack's TRSDOS and Disk BASIC. The exception relates to disk file access and is both the chief strength and principal drawback of MICRODOS.

Radio Shack's TRSDOS is the jumping-off point for most alternative operating systems. MICRODOS emulates several TRSDOS features. However, TRSDOS and Disk BASIC operate as distinct command levels, each with its own set of directives. Frequently, the TRSDOS user switches back and forth between levels to get to the appropriate command. By contrast, MICRODOS operates on only one level, which equates to the Disk BASIC level on a TRSDOS-based system. There are additional directives of the CMD "X" form to provide for such features as formatting a diskette, writing a copy of the operating system program onto a diskette and providing an automatic message at bootstrap time. The monitor program (written in machine language, not BASIC) stays in residence at all times.

Most of the features found in Radio Shack's Disk BASIC are replicated, using similar or identical coding conventions. On the positive side, there are familiar and useful features such as DEF, FNx, MID$, the ten USR exits and so on. Most definitely on the negative side, the incredibly Mickey Mouse method of disk record formatting (remember LSET, CVD and all that jazz?) is imitated as well.

The one important difference lies in the management of disk space and the method of file access. TRSDOS manages the disk space for you, and does it rather well, thank you very much. (Nay-sayers and doubters are sentenced to spend six months struggling with the file mismanagement features of IBM's DOS, after which they will never again criticize any micro operating system!) In TRSDOS, the physical addresses of files remain invisible. Files can be accessed by name only. Space is immediately released to the pool of free sectors upon execution of a KILL command. A file can be distributed over several non-contiguous extents (granules in TRSDOS terminology). Space allocation and expansion are completely automatic.

By contrast, MICRODOS does no management of physical disk space. The user can read and write to a specified disk sector, using a sector number in the range 0-399 (assuming a 40 track drive). A BASIC program is loaded and executed a command in the form LOAD DSSS, R with D equal to the drive number, SSS equal to the sector number, and R an optional parameter that calls for immediate execution. The operating system is just smart enough to load contiguous sectors, starting with DSSS, until an end of file marker is sensed. The system proper contains no file index. There is, however, an interesting utility program written in BASIC which replicates many of the functions of the DIR command. The file index must, of course, be manually posted by the user since the system does no disk space management at all. The DIR program and the other BASIC utilities are called by menu selection. Indeed, the utilities form an extension of the monitor so that functions such as FORMAT, BACKUP and COPY can be executed.

Where does this unique product fit in the world of TRS-80 software? This reviewer sees two potential user communities. MICRODOS was originally written for the use of BASIC-oriented software houses which might wish to economize on the amounts of diskette space and main memory allocated to system overhead. The operating system (less utilities) does indeed occupy only 7K of memory and one track of the diskette. The production version of a software package is easily configured as a "load and go" diskette which relieves the consumer of all concern with program startup. The non-compatibility of MICRODOS files, diskette formats, and file access methods might serve as a minor protection feature for a vendor's software although any persistent user could LIST and rekey the program with appropriate modifications to the file access routines. Under MICRODOS, the 16K disk system becomes practical, which could be a powerful selling point.

The other potential market lies in the large number of hobbyists whose programming knowledge does not extend beyond BASIC but who, nevertheless, want to experiment with creating or modifying their own system software. The utilities provided with the system do indeed give insight into simple implementation of operating system-type functions in a high level language. They are clearly written, with none of the symptons of programming "gamesmanship" that some program authors regrettably delight in. It might indeed be good clean fun to extend and modify these functions and, in effect, customize the BASIC language part of the operating system software to suit personal preferences and prejudices. Although memory is cheap "nowadays" even the owner of a 48K maximum system might find himself running out of space for that ultimate version of "TREK". With MICRODOS, he can use 41K for his code, which should be enough for most of us.

Those of us who write in Assembler or in FORTRAN will find little use for MICRODOS. There is no facility for supporting these languages or the object code produced by their translators. MICRODOS is for BASIC freaks only. And Radio Shack's method of accessing data files, terrible to start with, is made a little worse by MICRODOS.

This writer has a variety of operating systems and variants available. Most day-to-day work is done under TRSDOS 2.3 as modified for 40 track operation by Percom's PATCHPAK program (PATCHPAK and MICRODOS are both the product of James M. Stutsman and he can play on our team any old time). MICRODOS will go on the rack next to RSM2D, DOS+ and the other special purpose software systems we use. It's unique, interesting, and may come in handy someday, although we don't put out much BASIC language code around here. It won't replace TRSDOS or the aforementioned special products which preceded it into our shop.

If BASIC is your thing, and if you need any of the special features of MICRODOS, or you have limited main memory, or you just plain want to fool around with disk utility writing, by all means give the package a try. The price (less than $30.00 retail) makes it too tempting to resist.

WHAT AN ACCOUNTANT DOES WITH A TRS-80

by James H. Sheats

One of the more commonly suggested uses for a microcomputer is the broad category of use referred to as 'accounting applications'. As an accountant and tax preparer, I thought that it might interest some of the readers to hear how I use the TRS-80 in my business.

My business consists of general ledger write-up and income tax work. I use an NCR magnetic ink encoder and a service bureau that provides a complete set of financial statements. I priced out an equivalent general ledger package for the TRS-80. The hardware requirement was three disk drives and a line printer with at least 96 characters per line. The software alone would cost about $2,200.00. I don't intend to purchase this right away for obvious reasons.

I have attempted to use three different computerized tax preparation services. I can still see instances where a good tax service would be useful, but a large volume of returns is necessary to justify the time spent in learning rather complicated input procedures and forms of these services. However, I may try again with an income tax package for my TRS-80.

My own TRS-80 system has a 48K memory, a line printer, and two disk drives. I have an amortization schedule program which is useful year round since I share an office with two lawyers.

I wrote two payroll programs that begin with input hours and calculate gross pay, social security, federal and state withholding tax, and the new Advanced Income Credit. One of the programs uses DATA statements to input hourly wage rate, marital status, and dependent information for each employee. The other one uses INPUT statements for this. This latter is useful to me when a customer wants to know how much to deduct from an employee's paycheck and he has mislaid his charts.

I occasionally have to prepare depreciation schedules and other special explanatory schedules for customers' tax returns and other purposes. In most cases, the same format will be used every year for this particular customer with only the figures themselves changing. This situation is a 'natural' for the Electric Pencil; I use it extensively. Each customer's schedules are stored in a separate Pencil file. I also have purchased SCRIPSIT and intend to convert some of the Pencil files during the slack season.

I do not use a checkbook balancing program. I do many reconciliations every month and I have not found it difficult to perform this task the old fashioned way. Most 'home budget' programs are not adequate for my own needs although, if a customer walked in with a printout from one, I would probably use it in preparing his return.

My plans for the future include the purchase of one or more additional disk drives, a telephone interface, and some more software, including a really good general ledger package. These plans are made based on the needs of my bookkeeping and tax service as they exist now. Should other opportunities develop, I may make more extensive use of my TRS-80 to furnish other services to my customers, but I have no definite plans in this regard at present.

SUPERMENU for NEWDOS Users

by James Garon

Are you still suffering with the problems and downright errors of TRSDOS? Have you listened with awe and envy to the users of NEWDOS and NEWDOS+? This program is not designed to make you feel better. In fact, it is a BASIC program which CANNOT be run under TRSDOS. This is because it makes use of one of NEWDOS' special features: the ability to get a DIRectory while remaining in BASIC.

Once the DIRectory is on the screen (which is actually just another 1K of memory to the Z-80 chip), the program, called SUPERMENU, reads the list of programs from the screen and into an array. Each program is assigned a symbol (@, A, B, ..., Z) allowing up to 27 programs to be run with a single keystroke — even machine language programs! This is possible because NEWDOS can execute a machine language program from BASIC (if its name ends in "/CMD"). SUPERMENU decides automatically whether to "RUN" or "CMD" a given program.

This whole process can be made even more automatic by typing (while in DOS):

AUTO BASIC RUN "SUPERMENU"

This requires, of course, that you also SAVE the program itself under the name "SUPERMEN" (The computer will ignore the "U" in the AUTO-command — but it looks nicer).

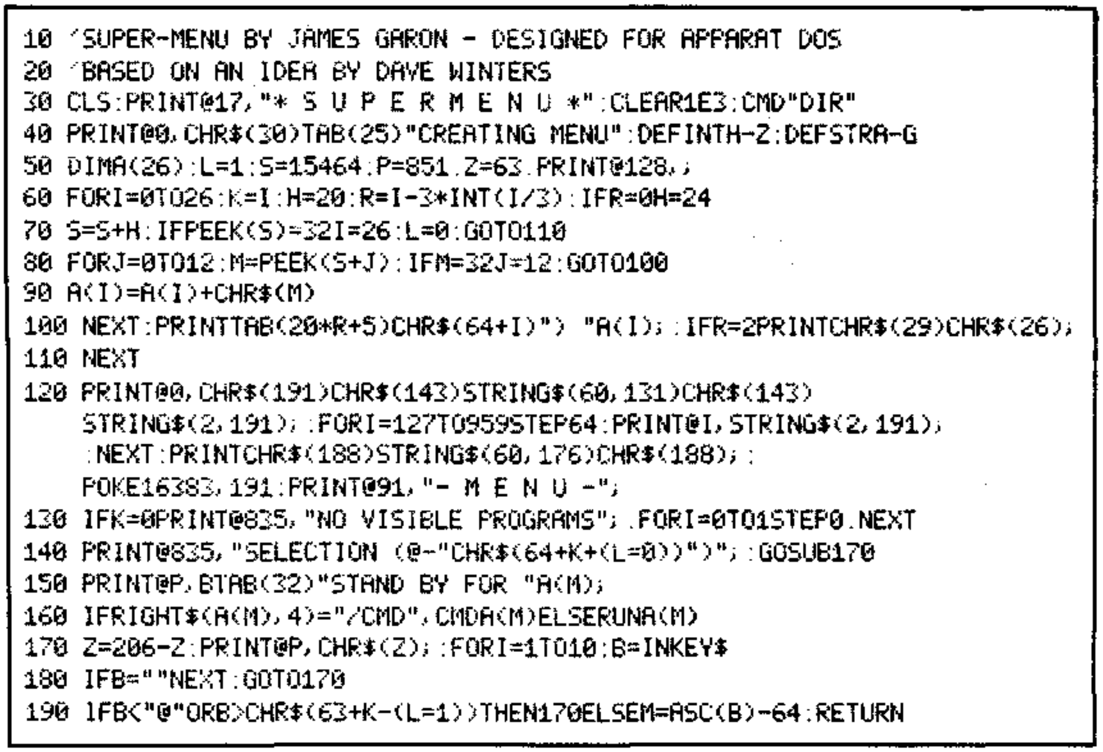
If you are willing to work a little harder, you can automate your whole program library. Take a look at one of your programs. If it has any END statements, change them to:

RUN "SUPERMENU"

If it has no END statement, then place: RUN "SUPERMENU" after the last line. Now SUPERMENU calls your program and your program returns to SUPERMENU!

Since SUPERMENU as written will create a single key DIRectory/menu for Drive 0, what do you do if you have two or more drives? Just change line 30 from ...CMD"DIR" to CMD"DIR :1" and save the new program under the name "DRIVE1". If you have the drives, save a program called "DRIVE2" with line 30 containing CMD"DIR :2" and so on (Each such program takes only one gran). Now, when you press the RESET button, you will soon see a single-key menu of the programs on Drive 0. One of them will be called "DRIVE1". Press the letter next to "DRIVE1" and, PRESTO, you are soon gazing fondly at the programs on Drive 1!

TRSDOSers, eat your hearts out!



```
10 'SUPER-MENU BY JAMES GARON - DESIGNED FOR APPARAT DOS
20 'BASED ON AN IDEA BY DAVE WINTERS
30 CLS:PRINT@17,"* S U P E R M E N U *":CLEAR1E3:CMD"DIR"
40 PRINT@0,CHR$(30)TAB(25)"CREATING MENU":DEFINTH-Z:DEFSTRA-G
50 DIMA(26):L=1:S=15464:P=851.Z=63.PRINT@128,;
60 FORI=0TO26:K=I:H=20:R=I-3*INT(I/3):IFR=0H=24
70 S=S+H:IFPEEK(S)=32I=26:L=0:GOTO110
80 FORJ=0TO12:M=PEEK(S+J):IFM=32J=12:GOTO100
90 A(I)=A(I)+CHR$(M)
100 NEXT:PRINTTAB(20*R+5)CHR$(64+I)") "A(I); :IFR=2PRINTCHR$(29)CHR$(26);
110 NEXT
120 PRINT@0,CHR$(191)CHR$(143)STRING$(60,131)CHR$(143)
    STRING$(2,191); :FORI=127TO959STEP64:PRINT@I,STRING$(2,191);
    :NEXT:PRINTCHR$(188)STRING$(60,176)CHR$(188); :
    POKE16383,191:PRINT@91,"- M E N U -";
130 IFK=0PRINT@835,"NO VISIBLE PROGRAMS"; .FORI=0TO1STEP0.NEXT
140 PRINT@835,"SELECTION (@-"CHR$(64+K+(L=0))")"; :GOSUB170
150 PRINT@P,BTAB(32)"STAND BY FOR "A(M);
160 IFRIGHT$(A(M),4)="/CMD",CMDA(M)ELSERUNA(M)
170 Z=206-Z:PRINT@P,CHR$(Z); :FORI=1TO10:B=INKEY$
180 IFB=""NEXT:GOTO170
190 1FB<"@"ORB>CHR$(63+K-(L=1))THEN170ELSEM=ASC(B)-64:RETURN
```

DEPRECIATION

by Daniel Miller, Astoria, NY

If you own your own small business, this program will help you complete Form 1040, Schedule C-2 'Profit (or Loss) From Business or Profession'.

The program calculates depreciation by the three following procedures:

(1) Straight line;
(2) Declining balance (125%, 150%, 200%);
(3) Sum of the year's digits.

Before beginning, determine the adjusted basis (your actual cost) for the item. The adjusted basis is the total of your purchase price, money borrowed and the fair market value of services or property you provided.

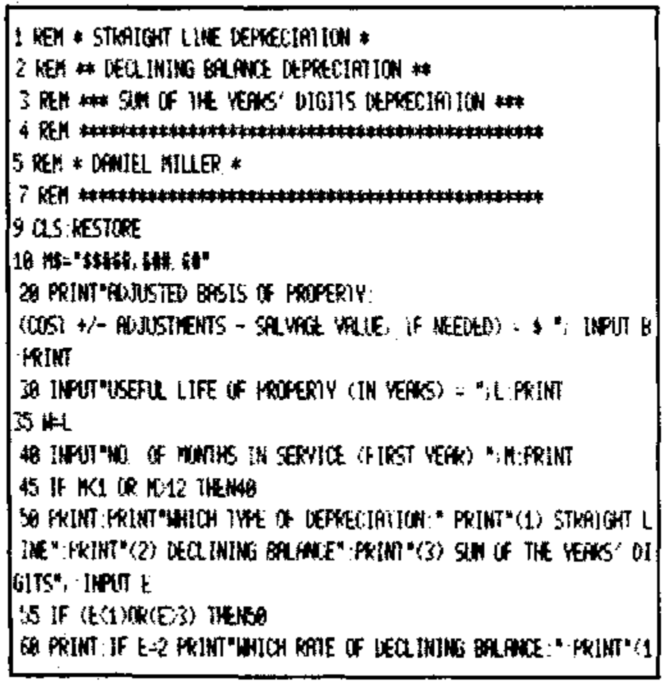
Certain adjustments, e.g. transportation and installation costs, and outlays for capital improvements,will raise the adjusted basis; others, e.g. depletion allowances and casualty losses, will lower the basis.

(For a complete description of basis and depreciation, obtain I.R.S. Publication 17, 'Your Federal Income Tax'.)

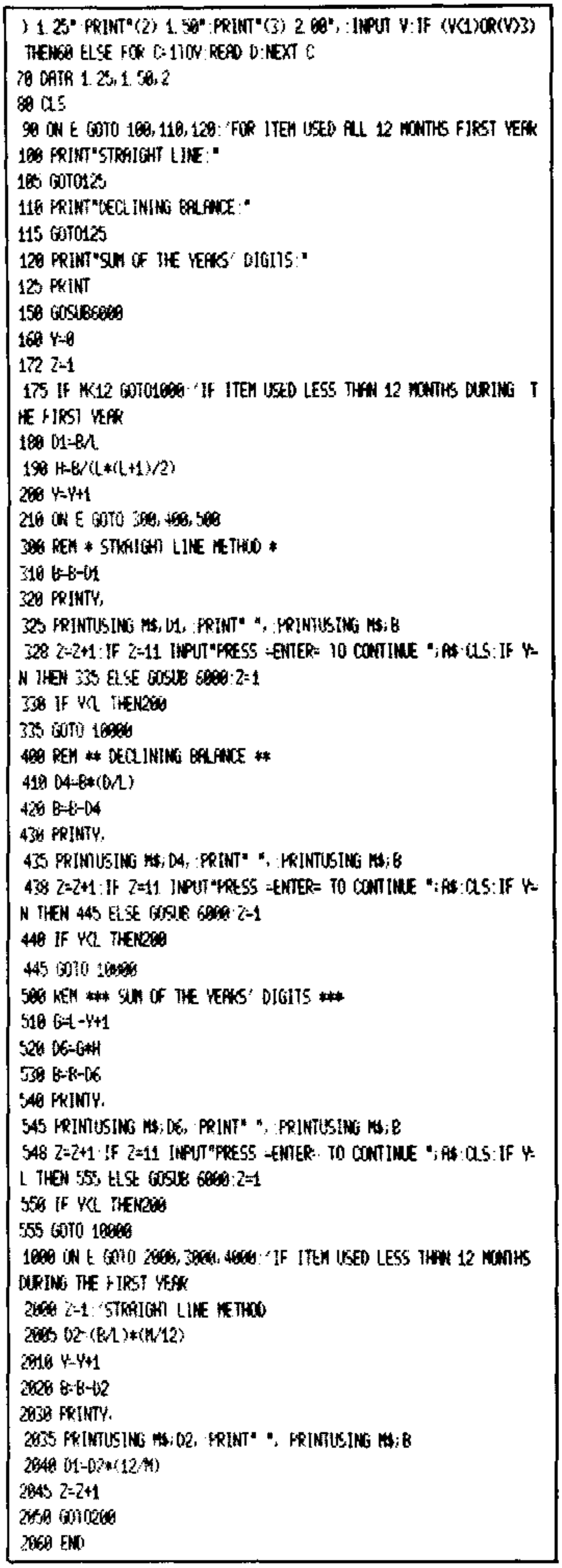
If you use either the straight line or sum of the year's digits method, subtract from the basis the salvage value, i.e. the amount you believe you could get at the end of the item's useful life. This figure, the adjusted basis, will be used in the program.

Do not depreciate the item past its salvage value.

An item put into service by the fifteenth of the month is considered in use for the entire month. Otherwise it is not considered for depreciation until the following month. If an item has been used for the entire first year, enter '12'; if not, enter the appropriate number of months.



```
1 REM * STRAIGHT LINE DEPRECIATION *
2 REM ** DECLINING BALANCE DEPRECIATION **
 3 REM *** SUM OF THE YEARS' DIGITS DEPRECIATION ***
 4 REM ***********************************************
5 REM * DANIEL MILLER *
 7 REM ***********************************************
9 CLS:RESTORE
10 M$="$$##,###.##"
 20 PRINT"ADJUSTED BASIS OF PROPERTY:
 (COST +/- ADJUSTMENTS - SALVAGE VALUE, IF NEEDED) = $ "; INPUT B
 :PRINT
 30 INPUT"USEFUL LIFE OF PROPERTY (IN YEARS) = ";L:PRINT
35 N=L
 40 INPUT"NO. OF MONTHS IN SERVICE (FIRST YEAR) ";M:PRINT
 45 IF M<1 OR M>12 THEN40
 50 PRINT:PRINT"WHICH TYPE OF DEPRECIATION:" PRINT"(1) STRAIGHT L
 INE":PRINT"(2) DECLINING BALANCE":PRINT"(3) SUM OF THE YEARS' DI
GITS", :INPUT E
 55 IF (E<1)OR(E>3) THEN50
 60 PRINT:IF E=2 PRINT"WHICH RATE OF DECLINING BALANCE:":PRINT"(1
```



```
 ) 1.25" PRINT"(2) 1.50":PRINT"(3) 2.00", :INPUT V:IF (V<1)OR(V>3)
 THEN60 ELSE FOR C=1TOV:READ D:NEXT C
70 DATA 1.25,1.50,2
80 CLS
 90 ON E GOTO 100,110,120:'FOR ITEM USED ALL 12 MONTHS FIRST YEAR
100 PRINT"STRAIGHT LINE:"
105 GOTO125
110 PRINT"DECLINING BALANCE:"
115 GOTO125
120 PRINT"SUM OF THE YEARS' DIGITS:"
125 PRINT
150 GOSUB6000
160 Y=0
172 Z=1
 175 IF M<12 GOTO1000:'IF ITEM USED LESS THAN 12 MONTHS DURING  T
HE FIRST YEAR
180 D1=B/L
 190 H=B/(L*(L+1)/2)
200 Y=Y+1
210 ON E GOTO 300,400,500
300 REM * STRAIGHT LINE METHOD *
310 B=B-D1
320 PRINTY,
 325 PRINTUSING M$;D1, :PRINT" ", :PRINTUSING M$;B
 328 Z=Z+1:IF Z=11 INPUT"PRESS =ENTER= TO CONTINUE ";A$:CLS:IF Y=
N THEN 335 ELSE GOSUB 6000:Z=1
330 IF Y<L THEN200
335 GOTO 10000
400 REM ** DECLINING BALANCE **
 410 D4=B*(D/L)
420 B=B-D4
430 PRINTY,
 435 PRINTUSING M$;D4, :PRINT" ", :PRINTUSING M$;B
 438 Z=Z+1:IF Z=11 INPUT"PRESS =ENTER= TO CONTINUE ";A$:CLS:IF Y=
N THEN 445 ELSE GOSUB 6000:Z=1
440 IF Y<L THEN200
 445 GOTO 10000
500 REM *** SUM OF THE YEARS' DIGITS ***
510 G=L-Y+1
520 D6=G*H
530 B=B-D6
540 PRINTY,
 545 PRINTUSING M$;D6, PRINT" ", :PRINTUSING M$;B
 548 Z=Z+1:IF Z=11 INPUT"PRESS =ENTER= TO CONTINUE ";A$:CLS:IF Y=
L THEN 555 ELSE GOSUB 6000:Z=1
550 IF Y<L THEN200
555 GOTO 10000
 1000 ON E GOTO 2000,3000,4000:'IF ITEM USED LESS THAN 12 MONTHS
DURING THE FIRST YEAR
 2000 Z=1:'STRAIGHT LINE METHOD
 2005 D2=(B/L)*(M/12)
2010 Y=Y+1
2020 B=B-D2
2030 PRINTY,
 2035 PRINTUSING M$;D2, PRINT" ", PRINTUSING M$;B
 2040 D1=D2*(12/M)
2045 Z=Z+1
2050 GOTO200
2060 END
```

```
3000 Z=1 'DECLINING BALANCE METHOD
3005 D5=((D/L)*B)*(M/12)
3010 Y=Y+1
3020 B=B-D5
3030 PRINTY,
3035 PRINTUSING M$;D5, PRINT" ", PRINTUSING M$,B
3040 D4=D5*(12/M)
3045 Z=Z+1
3050 GOTO200
3060 END
4000 Z=1 'SUM OF THE YEAR'S DIGITS METHOD
4001 H=B/(L*(L+1)/2)
4002 Q=(L*(L+1)/2)
4003 Y=Y+1
4005 G=L-Y+1
4010 D7=(G*H)*(M/12)
4020 B=B-D7
4030 PRINTY,
4035 PRINTUSING M$;D7, PRINT" ", PRINTUSING M$;B
4040 D8=((D7*(12/M))*(12-M))
5000 Z=Z+1
5005 Y=Y+1
5010 D9=(D8*(W/Q)*((12-M)/12))+(D8*((W-1)/Q)*(M/12))
5020 B=B-D9
5030 PRINTY,
5035 PRINTUSING M$,D9, PRINT" ", PRINTUSING M$,B
5038 IF Z=10 INPUT"PRESS -ENTER- TO CONTINUE ",A$ CLS IF Y=L THE
N 5050  ELSE GOSUB 6000 Z=1
5040 W=W-1
5045 IF Y<L THEN5000
5050 GOTO 10000
6000 PRINT"END OF YEAR","DEPRECIATION","PRESENT VALUE" PRINT RET
URN
6050 END
10000 PRINT INPUT"ARE THERE ANY MORE ITEMS TO DEPRECIATE ",Q$
10010 IF LEFT$(Q$,1)="Y" THEN CLS RESTORE GOTO 20
10020 PRINT END
```

PEEK and POKE

From "Learning Level II" by David Lien

PEEK and POKE are BASIC words that allow us to do non-BASIC things. They provide the means whereby we can PEEK into the innards of the computer's memory, and if we wish, POKE in new information.

It is not our purpose here to become an expert in machine language programming, or even on how the computer works. We have to approach this and related topics a little gingerly lest we fall over the edge into a computer abyss (or is it an abysmal computer?).

We do know, however, that computers do their thing entirely by the manipulation of numbers. Therefore, when we PEEK at the contents of memory, guess what we'll find? Numbers? Very good! (Ummmyass).

INSERT FIGURE 1.

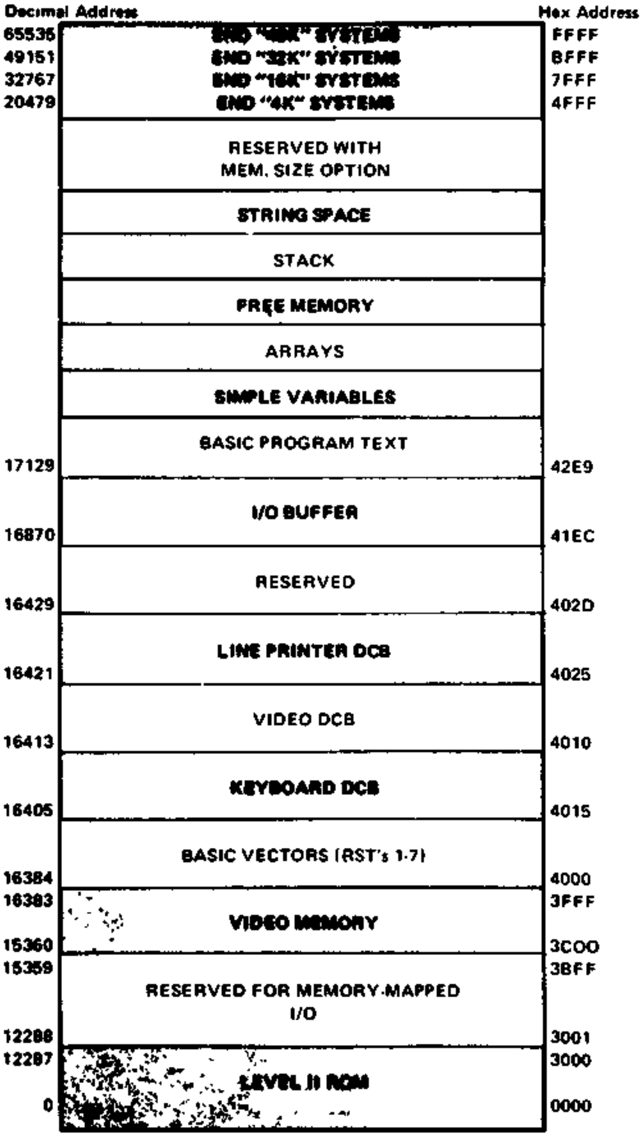


Figure 1. Level II Memory Map

As you can easily see from the Memory Map in Figure 1, large chunks of the computer's memory are set aside, or "mapped" for very specific uses. (Oh, you can't see it easily...? Had your eyes checked lately?) The Level II ROM for example, uses byte address 0 through 12287. All numbers we talk about here are decimals, not hex, octal, binary or Sanskrit.

Type in this program:

```
20 N=0
50 PRINT N, PEEK(N), CHR$(PEEK(N))
60 N=N+1
70 GOTO 50
```

Let's analyze the program before RUNning it.

Line 20 sets the beginning address where we want to start PEEKing. As Figure 1 shows, there are lots of good places to go spelunking, and we can change line 20 to start wherever we want.

Line 50 prints three things:

A. The address — that is, the number of the byte, the contents of which we are PEEKing at.

B. The contents of that byte, expressed as a decimal number between 0 and 255.

C. For convenience (and some value), the contents of that address converted to its ASCII character. (Many of the ASCII characters are not printable — we warned you it was a ribald novel.) Go back to the chapter on ASCII if your memory has grown dim.

OK, now RUN the program, being ready to stop it with SHIFT@ if you see something interesting. It can also be stopped at any time with the BREAK key, and restarted with CONT without having to start all over again with N at 0.

Didn't see anything interesting? What did you find starting at address 261??? You have to be able to read vertically as the letters swish by.

When the letters jump to double width, hit STOP, then CLEAR, then CONT, as they are too hard to read when so large. Change N to start at different places in memory and PEEK to your heart's delight. You can't goof up anything by just PEEKing. It's indiscriminant POKEing that gets you into trouble.

The command level is very handy for resetting the starting address. Change the value of N by just typing:

```
N=5000
```

for example, then

```
CONT
```

instead of RUN

When done PEEKing with this program and having seen far more information than can possibly be absorbed, rework line 50 to read simply.

```
50 PRINT CHRl (PEEK(N));
```

and RUN.

It PRINTs only the ASCII characters, horizontally, and is the ideal program to RUN when friends visit. Just act casual about the whole display and avoid any direct questions. Makes a great background piece for a science fiction movie.

When you find an interesting spot, hit BREAK, then

```
PRINT N
```

at the command level to find out where in memory you are PEEKing. (Don't you wish you could explore the corners of your mind as easily?)

CONTinue on when ready.

Having moved from PEEKing to leering, it's time to see what else we can do.

Careless POKEing can leave holes...

Before POKEing, we'd better see that we're not POKEing a stick into a hornet's nest. It's with the greatest of ease that we destroy a program in memory by POKEing around where we shouldn't.

Obviously there is no use POKEing in the ROM area since ROM stands for Read Only Memory. It's not changeable. The rest of the "Memory mapped" area, from 12288 through 17129 is reserved for specific things, so best not to POKE in there while we're just bungling around. Anything above 17129 should be available memory, unless taken up with our BASIC program or required for processing. With such a short program as ours we surely can't goof anything up. Can we?

Let's PEEK around 20000 and see if anything is going on there. Change two program lines to:

```
20 N = 20000
50 PRINT N; PEEK(N),
```

and RUN

```
20000    255  20001    255   2002    255  20003    255
20004    255  20005    255  20006    255  20007    255
20008    255  20009    255  20010    255  20011    255
20012    255  20013    255  20014    255  20015    255
20016    255  20017    255  20018    255  20019    255
20020    255  20021    255  20022    255  20023    255
20024    255  20025    255  20026    255  20027    255
20028    255  20029    255  20030    255  20031    255
20032      0  20033      0  20034      0  20035      0
20036      0  20037      0  20038      0  20039      0
20040      0  20041      0  20042      0  20043      0
20044      0  20045      0  20046      0  20047      0
20048      0  20049      0  20050      0  20051      0
20052      0  20053      0  20054      0  20055      0
20056      0  20057      0  20058      0  20059      0
20060      0  20061      0  20062      0  20063      0
```

What we see are the address numbers and their contents, in easy-to-read parallel rows. Unless you've been messing around with other programs since power-up, you should just see nice rows of 255's and 0's. The memory at this location is not being used.

Great! Let's change our program and POKE in some information and do something with it. Make it read:

```
10 REM * POKE PROGRAM *
20 N = 20000
40 READ D
50 POKE N,D
60 N = N+1
70 IF N = 20011 END
80 GOTO 40
100 DATA 80,60,69,75,45,65,45,66,79,79,33
```

Before RUNning, let's analyze it.

Line 20 initializes the starting address at 20000

Line 40 READs a number from the DATA line

Line 50 POKEs the DATA "D" into address "N"

Line 60 increments the address number by one

Line 70 ENDs execution when we have POKEd in all 11 pieces of DATA

Line 80 sends us back for more DATA

Line 100 stores the DATA we are going to POKE into memory.

Now = RUN

Well, that was sure fast. I wonder what it did? How can we find out? Should we PEEK at it? Yes, but let's leave the old program in and just start a new one at 200.

```
200 REM * PEEK PROGRAM *
210 FOR N=20000 TO 20010
220 PRINT N, PEEK(N)
230 NEXT N
```

and RUN200

```
20000  80
20001  69
20002  69
20003  75
20004  45
20005  65
20006  45
20007  66
20008  79
20009  79
20010  33
```

How about that. We really did change the contents of those memory locations. We shot the numbers from our DATA line right into memory. Now if we only knew what those numbers stood for. Wonder...if we changed them to ASCII characters, would they tell us anything?

Add:

```
205 CLS
220 PRINT@470+N-20000,CHR$(PEEK(N));
```

to print at a certain location on the screen and RUN200

Print The Results Here

And that's how PEEK and POKE work.

BOOK REVIEW — TRS-80 PROGRAMS

by A.A. Wicks

Published under the aegis of Radio Shack, a beginner-oriented book, **'TRS-80 Programs'**, provides 32 BASIC programs written especially for the TRS-80 computer, and, as the preface states, 'They will work!'

The authors, Tom Rugg and Phil Feldman, have provided a most thorough and interesting text, in addition to the program. Following some explicit instructions on how to use the book in order to gain the most usefulness from it, each section has a preface and each program has a detailed preamble. The latter covers the purpose of the program, how to use it, and a sample run. this is followed by at least one video presentation of the program in operation. In most instances, typical sample runs are also provided. In the event that the user's display does not reflect the one shown, instructions are given as to what steps to take to correct the problem.

The book has been written so that no knowledge of programming or BASIC is required--merely type in the programs, and 'run'. However, because of the extensive text that follows each program listing, anyone wishing to learn more about programming in BASIC will be helped considerably.

Sections titled 'Easy Changes', and then 'Main Routines', which provide a break down by line number of the program logic and what each major portion of the program accomplishes, are excellent. In addition, a list of the Main Variables is included for reference. One more section for each program, 'Suggested Projects', provides a challenge for anyone wishing to go further, learning more about BASIC as they go.

And what of the programs? In general, a good selection of not less than four, and not more than six each, in the groupings of Applications, Educational, Games, Graphics Displays, Mathematics, and several short miscellaneous programs. A very few of some have been released commercially but otherwise have not been published, although there are similarities to other programs, e.g., 'Biorhythm'. The programs are designed to run on a 16K level II machine, but most will load within 4K without modification. Suggestions for restricted-memory loading are included. Wherever possible, full advantage is taken of the TRS-80 graphics, and tabulated displays are neat. None of the programs is extremely complex either in programming or subject matter, and their greatest value probably is in their potential for programming instruction. Nevertheless, the other side of instruction, the educational programs, are ideal for grade school (or pre-school!) guidance. And the business and mathematical programs can be very useful. The mathematics programs primarily cover higher mathematics and would be useful for students, engineers, and statisticians, with extensive use being made of graphs.

The printing quality of the listings is excellent--a credit to the production department and the printer--dilithium Press, of Portland, Oregon. The listings are of dot-matrix origin, but are sharp, well-inked black, and, for instance, leave no doubt regarding commas or periods. Print size is about 10-point typewriter size, and slightly larger than the text, although this may be an impression gained from the different type style of the text.

The cost of the book is reasonable--$5.95; it is available at Radio Shack stores, Catalog No. 62-2064. It is recommended for good entertainment, practical applications use, and programming skills improvement.