CODE BLEU IN COMPUTER ROOM!

HURRY UP, DOCTOR!

It's anti-drug.
In the last issue we printed a letter from member H. Spencer complaining about the treatment he received from Technical Micro Systems, Inc. (TMSI) in connexion with his purchase order for their SuperSet H19/889 character upgrade kit. We were unable to air TMSI's side of Mr. Spencer's complaint then, but we now have their position clarified in this letter they sent the JOURNAL. We sincerely hope this satisfies all those concerned with any complaints about TMSI! -- ed

Dear Len,

"In reply to Herbert Spencer's letter of 2/9/87, the check he sent us in the amount of $29.95 was returned to him with an explanation as follows:

"Enclosed you will find your check $2996, dated 12/2/816 for $29.95. I believe there has been a misunderstanding about the cost of the SuperSet. The going retail price is $49.95. We cannot apply a $20.00 credit towards a purchase when there is a $0.00 balance."

"I am including a current price list and catalogue in hope that it will somewhat clarify the situation."

"I have also sent him the information he has requested on the V30 option, and name and address of where he can get information for printing SuperSet Fonts on an Epson printer, since we don't have it ourselves."

"I believe this presents our side of his subject, which is that we returned his money since we couldn't fill his order, and that we have supplied all the information available to us."

"Lee Hart (per TSG)"

[Thanks, Lee! We hope this will improve relations between your firm and Mr. Spencer, as well as others who seem to be out of sorts with you--as you can see in these letters.]

Dear Mr. Geisler:

"You've got my vote for not typing ACK of ACES! My cheque for $6.96 is enclosed. Soft-sector Heath CP/W format is fine.

Your newsletter is really fine, and written at--and to--the right level as far as I'm concerned."

Regarding TMSI — I think Lee Hart's work is superb (FlickerFree Mod, and Write Hand Man), but I too experienced delays similar to what Harry Spencer wrote about in 1:7. Don't know what kind of operation TMSI is, but it's very obviously SMALL and appears to be "hand-to-mouth." There are a few entrepreneurs in the computer aftermarket that can be so classified. I don't think any are destined for the "Fortune 500," but they do make life interesting. That's why your "honest" appraisals of products available for the SEBHC really means a lot to me.

I've decided not to buy another thing based on an advertisement alone. In that regard, I can rave about the Dressel- bause DOTS PERFECT upgrade kit for my MX-80. They even gave me a credit of $30 for returning my 4-year-old MX-PLUS mod! Now that's a company with a real chance for success. Dan Dresselhaus has my vote for product of the year in DOTS PERFECT! And now I notice that someone is discounting it! It's not fair.

In the meantime, keep up the good work!

Dave Kletter

[Thank you for the kind words, Dave! Lots of folks have been sending orders for our GAME DISC #0 in both hard and soft-sector format. A couple of H/1000 "associate" members bought copies too, and they're delighted the way the game runs for them on their machines. And we're working on a DOS version which should be available in April.

Regarding Dresselhaus Computer Products: They serve a much bigger market than TMSI does. There are millions of MX-80/100 printers out there, all needing a Dots Perfect upgrade. But at most (I'm definitely guessing now!) there are about 350,000 H/2-69s, of which maybe 5% to 10% will ever feel the touch of TMSI's components or software. Rather like comparing bananas to pearls, wouldn't you say?

You're correct in assuming that TMSI is a small, although innovative firm. But that doesn't automatically stamp them as deliberately trying to bilk customers out of their money. As far as I've been able to ascertain, TMSI has--despite numerous problems inflicted upon them by certain unscrupulous parties--stubbbornly working to fulfill all outstanding orders to their clients' ultimate satisfaction. I also waited quite a while for an expensive item, but the wait was not as traumatic for me as it appears to have been for others.]

Dear Lenny -

Mr. Spencer may or may not have a legitimate bone to pick with TMSI--as I write this I don't know what TMSI's reply to his complaint will be. But in my own case, I've run into a number of things which could make me blow my top, but I'm holding back for the moment. I ordered some $1500-plus worth of stuff from TMSI over nine months ago, and have not yet received the complete order. Because I liked the many articles by Lee Hart that you've published in the JOURNAL I'm willing to wait just a little bit longer; his firm may yet pleasantly surprise me. I hope it's before I lose my temper!

Regards,

Jim Frank

[Jim, TMSI says the back order went out and you will be receiving it almost before you read this! I understand there was an unfortunate incident with the instruction manual for your H-1000 board at the printer's shop. The printer somehow lost the original manual's offset printing plates. Consequently, Lee had to single-handedly re-create the entire manual (text and illustrations) from scratch--all while 'doing a zillion other jobs'! Now that he has full-time clerical help,
More BOUQUETS, BRICKBATS, etc.

Lee is getting the few remaining back orders taken care of and shipped out as fast as humanly possible. I received my Northwest Digital "Graphics-Plus" card ordered last October a week or so back, so I personally know TMSI is in there and pitching hard--thanks Lee, it works just swell!

"BOB BOY--SOMETHING MIGHT BE WRONG!" News Department

In our last issue, new editorial staff member Ziggy Nebish somehow managed to thoroughly garble those assembly-language listings (CLS.ASM and LABEL.ASM) in putting them into double-column format. Ziggy's a pretty good B/Z computerist, but it took him a while to become familiar and expert with our two-column page format; he's promised to do better in the future. But that's another story. I hereby apologize to you members who copied the listings, and then couldn't get the programs to assemble and run properly. Here they are, CORRECT this time:

LISTING 1.

CLS.ASM CP/M CLEAR-SCREEN ASSEMBLY-LANGUAGE PGM

References: 280 Cookbook, SEKTANT #5, REMark V5#2
by L.E.Geisler, editor/publisher SEBHC JOURNAL
12-Jun-86

SET UP PROGRAM

bdos equ 0005h
esc equ 027h ;ESCEape Function (= 1bhex)

RUN PROGRAM

org 0100h ;TOP OF TPA
main: lxi d,cls ;Load Heath clear-screen code
mvi c,9 ;CP/M print-string function
call bdos ;Call CP/M
ret ;Return to CP/M
cls: db esc,'E','$' ;clear screen (print ESC E)

LISTING 2.

LABEL.ASM CP/M ASSEMBLY-LANGUAGE DISC LABELLER

References: 280 Cookbook, SEKTANT #5 & REMark V5#2

BOOT EQU 0000H
BDOS EQU 0005H
CONIN EQU 1 ;Console input
CONOUT EQU 2 ;Console output
PLINE EQU 9 ;CP/M print-string function
ESC EQU 18H ;ESCEape function (= 027d)

org 0100H ;Top of TPA
main: lxi D,TAB ;load tabs

CALL SENDLN ;print them
LXI D,VIDEO ;Load reverse video
CALL SENDLN ;Toggle reverse video
LXI D,M ;Load disc label "message"
CALL SENDLN ;Print label on screen
LXI D,VIDEO ;Toggle normal video
CALL BDOS ;Call CP/M
RET ;Return control to CP/M

SENDLN: MVI C,PLINE
CALL BDOS
RET

HEALTH TERMINAL ESCAPE CODES

RVIDEO = REVERSE VIDEO
NVIDEO = NORMAL VIDEO

RVIDEO: DB ESC,7,P,7
NVIDEO: DB ESC,7,7,7

TAB: DB 7,7,7
MSG: DB CP/M-80 ASSEMBLY-LANGUAGE BOOT DISC '7'
DS @14H ;Reserve space for 16 entries
STACK: DS @04H ;Top of Stack
BUFFER: EQU @ ;Storage area delineator

NOTICE

---------------

Someone kindly wrote to REMark about the SEBHC JOURNAL. We sincerely appreciate the publicity, but 'slick' publications are a bit out of phase, "datewise". This has resulted in many new JOURNAL applications coming in with cheques written for the wrong price.

To correct the problem, we're notifying low-priced subscribers by postcard of the correct price, and that they'll get only ten issues rather than 12 at $12.50. We believe that is fairest for all. At $15/year (one-half the $2.50/copy price) we're barely meeting the cost of publishing and mailing the JOURNAL. Want a full year? Please send $2.50 more. We'll update the expiration date on your very next address label.

Just received a mangled JOURNAL from the Post Office stamped "DAMAGED IN HANDLING IN THE POSTAL SERVICE R222" and "Returned For Better Address". Where do those @#&%!$*% get the nerve to demand increasing first-class postage! Henceforth you'll be getting JOURNALS wrapped in plastic baggies. Let's see what the Postal (mis)Service does with THAT!

Can you imagine what life'd be WITHOUT your B/Z 8-bit friends?
8 BIT CP/M SOFTWARE FOR YOUR HEATH/ZENITH

We would like to offer Heath/Zenith users 25% off the listed price on the following software through April 30, 1987:

CP/M-80:  DISK FIX, a file recovery and disk editing utility ($150)
           PROGRAM MAP, a Cross reference utility for MicroSoft BASIC programs ($150)
           INFO-80, an Integrated data base management system ($395)

CP/M-80 & PC-DOS:  MasterSweep, our new file maintenance utility ($49)
                    MasterCom, our new telecommunications utility ($49)

RUN YOUR HEATH/ZENITH SOFTWARE ON YOUR PC!

Have you been forced to purchase a PC? Do you have some favorite Heath/Zenith programs you would like to run on your PC? We would like to introduce you to a software program that allows you to add CP/M capability to your PC! This program is RUN/CPM by Micro Interfaces.

RUN/CPM is an exciting new software program that allows you to execute CP/M 8 bit software on PC's while remaining within the MS-DOS environment. You can run your CP/M programs from "hard disks" and "RAM disks" with support for; sub-directories, pathname assignments, terminal emulation for many CP/M terminals, a disk-transfer utility for transferring CP/M software to MS-DOS format and the ability to issue MS-DOS commands in CP/M mode. You will also be able to run your CP/M programs in color, re-direct your I/O, run MS-DOS background programs such as "Sidekick" while remaining within your CP/M program. RUN/CPM supports over 175 different disk formats including 48 and 96 TPI disk drives. (Sorry, H/Z hard sector format is not supported but soft sector format is).

RUN/CPM ($99.95) Requires replacing your PC processor chip with ONE of the following:

V-20 Chip ($25) Replaces Intel 8088 or 8088-2 processor
V-30 Chip ($25) Replaces Intel 8086 processor
Z80 Board ($100) PC addon card with Z80 and Memory for any PC, AT or XT

With the purchase of RUN/CPM before April 30, 1987, we will give the V20 or the V30 chip with chip puller free or $25 off the purchase price of the Z80 board. To make this offer unbeatable, we provide a 30 day money back guarantee. IF YOU ARE NOT SATISFIED WITH RUN/CPM YOU CAN RETURN IT FOR A COMPLETE REFUND.

This offer is good only until April 30, 1987. Please include the disk format for CP/M software.

The Software Store
706 Chippewa Square
Marquette, MI 49855
(906) 228-7622
RENUMBER.BAS — A Selective BASIC Renumber Program

Copied from CodeWorks magazine Issue 9, Jan-Feb 1987

[Editor's note: This program is too useful to pass up! Even though you think BASIC's beneath notice as a Real Programmer's Language, there are times when it's easier to write BASIC programs which ACTUALLY WORK, than to devote hours of intensive effort to developing programs in assembly or some other closer-to-machine language. Run a BASIC program through a compiler and it will be virtually impossible to identify as BASIC because it is changed to machine language!]

BASIC's RENUM is fast and easy to use, but it has some limitations. Although slower, this program will let you renumber just a few lines, or all of a program and all line references remain intact. The few things which BASIC's RENUM Absolutely Won't Do can be annoying! For one, it renumbers from any program line to the end, but it WILL NOT change line references in the un-renumbered portion. And it Absolutely Won't let you renumber a section within a program. It's syntax is: RENUM, STARTLINE, INCREMENT. If increment isn't specified, BASIC defaults to the standard 10. There's no provision for STOPLINE, and the only program part renumbered is what was specified.

RENUMBER.BAS is our answer to this dilemma. It has a few characteristics which some folks who're quite irrational about BASIC will gleefully use as excuses for avoiding RENUMBER.BAS. It's in MBASIC and slower than MBASIC's RENUM. Also you must first save a program in ASCII before RENUMBER it. But it does what you want, plus taking care of all line references elsewhere in other parts of the program. You can elect to RENUMBER the entire program, or any section within it. If you customarily keep subroutines at 1000, 2000, 3000, etc., and happen to run out of space between 2000 and 3000 because all your lines are bunched near 2000, renumber only that section and give yourself some more space! As it renumbers, this program will look at--and change as required--all line number references into and out of the area being renumbered. It also makes a check first to see if the numbers and increments will fit within the specified area. If not, it will tell you the maximum increment to use to fit it in.

A BRIEF OVERVIEW --

RENUMBER.BAS reads the target program line by line, stripping off line numbers and saving them in one of two parallel arrays, A(). Then it fills the other B() array with new numbers calculated from your answers to INPUT questions about where to start, stop, and increment. The B() array contains zeros for any index number in the A() array not needing changing, but contains new numbers for those lines scheduled for change. The program then re-reads the target program one line at a time, checks all line numbers, finds them in the A() array, then checks the B() array. This includes not only the line number itself, but any line number reference within that line. RENUMBER also prints out 'Undefined LINE in LINK n' and prints a question mark at that spot in the renumbered line; an unusually handy debugging feature not found in BASIC's RENUM.

DETAILED PROGRAM DESCRIPTION --

Please refer to Listing One as you read the following.

The first few REMark lines need no explanation. Line 140 is necessary with MBASIC V4.02 and perhaps in V5.2x. Line 150 sets a maximum number of 600 code lines in the target program; change only if memory is adequate for larger arrays.

Line 160 dimensions A() and B() arrays to 600 lines each.

Line 170 thru 330 are heading and description lines. Line 180 asks for name of file to be renumbered and reassigns it as F$.

Line 370 sets the Starting line Number for the renumbering process.

Line 400 gives a choice between a specific Ending Number or letting RENUMBER run until end of target program.

Line 420 asks for LN (increment) between lines.

Line 440 tells you the target file is being renumbered.

Lines 470 thru 540 reads the target program, finds each line number, converts it from string form to an integer and puts it in the A() array. The loop starting at line 480 reads target program lines from 1 to NL (as set in line 150, above).

In line 490, if reaching end of file (ZOF) before reaching NL, we jump out of the loop because all target program lines must have been read.

Line 500 "line inputs" target file lines through buffer $1 and assigns each line to AS$.

Line 510 examines each line as it is read, searching for the first space from the left end. (BASIC insists a space follow a line number.) Variable S identifies the line position this space occupies.

Line 520 then assigns the actual line number to variable LN by taking the VAL(ue) of AS$'s left end to space S and converts
it from a string to an integer and the remainder of A is discarded; only line numbers are needed. Line 530 assigns line numbers (LN) to the A() array. As the loop in lines 460 thru 540 reaches EOF, A() holds the target program's integer line numbers.

The loop counter runs one count ahead of the lines read. Line 560 makes W (target-program lines) equal the counter, less one. We do not look for EOF, or use IF statements to find the file's length, only loop or read from one to W. And when finished here, line 550 closes file 1.

Lines 570 and 580 set line-renumbering limits. The first item in A() is A(1), the lowest/first target program line number. Back in lines 360 and 370 we said pressing RETURN would start the renumbering process at the first line. In line 570 we check if the start number (SN) is less than A(1). Pressing RETURN leaves SN at zero. If it's less, then SN:=A(1). If SN is equal or greater (>=) than A(1) then line 570 is ignored. If we didn't specify an ending number in lines 390 and 460, line 580 makes the ending number = 65500—almost the highest number BASIC can use (actually 65529). This comes into play when renumbering an entire program and using large between-line increments.

Thus far we have start and end number and between-line increments (SN, EN, and IC, respectively). Now we build the B() array in lines 610 thru 640, with the A-array number index (I). If the A-array number falls between specified start and ending numbers, we put the new number into the B-array at that index location.

In line 610 we make variable SU = SN. Variable SU is used temporarily, then dumped; we don't want to change the start number—we'll need it later. But SN must increment by IC each time thru the loop. Rather than destroy the value in SN we assign SU as equal to SN, using it instead.

The loop from 620 thru 640 reads each A-array value. If it is larger than SN and less than EN the corresponding B-array value is determined by temporary start number SU plus increment—SU being incremented each time around the loop.

At line 630's end, variable CT is an accumulating counter, registering how many lines are to be changed. It's needed later when RENUMBER tries to see if that number of lines with a given increment will fit the NL 600-line space without running into existing line numbers. When finished with this loop the original SN is intact and for every A-array line number which needs to be changed there will be a properly incremented number in the B array at the same index location.

Lines 670 thru 700 are used to check if the number of lines to change and their increments will fit in the allowed space without crashing into the Ending Number. Line 670 says, "If SN plus CT multiplied by Increment is less than EN, then continue (they'll fit)." If the total is equal to or larger than EN, it's a no fit. RENUMBER tells you that, then proceeds to line 800, calculates the maximum increment which would fit and tells you what it is. If what you'd like to do won't work, you're stuck with the B-array full of bad numbers.

In line 860 you're asked to enter a new increment, then in line 700 the B-array is cleared out and we pop back to 610, redefine SU, and clear the CT counter. We have to clear CT because it's an accumulator and a number left in it would be doubled—something we don't need.

Either way, whether lines fit or not, we eventually get to line 710 which prints the index number and the corresponding values in the A and B arrays on the display.

Line 710 is very useful as it shows if what you wanted to happen actually did. Delete this line (or REN it out) when satisfied RENUMBER BAS works as described, but do remember to change the reference to line 710 in line 670 from 710 to 740!

At this point the target program is still on disc and we've read one line of it at a time, stripped off its' line numbers and put them into the A array. We know where to start and end renumbering and what increment to use, and we know that what we want done can be. The B array has been set up and contains numbers which will replace the A-array numbers when actual renumbering takes place.

FIND AND REPLACE --

Now we read the target program again, one line at a time, find any line-number references which need changing, do it, and write them back out to a new file. We could build the new file in memory and then put it back again with the same name, but we'd only have one shot at getting it right! It's better we don't monkey with the original file other than reading it, and build a new renumbered file. For that we need a new file name.

In line 740 we strip off the last 4 character positions of the old file name (the period and BAS). Then we attach a new extension to the old name (.NEW) and call it F1$. We find the old file name's length, then take its' mid string (MID$) starting at position one to the length, less four. We'll call that much of it TP$ (temporary string). Lastly, we create a new filename by replacing TP$ with NEW.

Now that we've our new file name, we open it for output in line 770 to file buffer #2. This is because buffer #1 is also open and reading from the old file as buffer #2 writes to the new. BASIC won't simultaneously read and write files.
THE MAIN LOOP --

Lines 790 thru 1000 are the main (I) processing loop. In
side this loop is a secondary (Q) loop, lines 870 thru 900. We'll run through the main I loop first and the Q loop later.

Line 800 reads one line from the ASCII target program disc file. These lines don't accumulate in memory. In fact, the only program in memory is RENUMBER and its' two arrays built earlier. "LINE INPUT #1,A$" reads the first target line up to its' carriage return (every BASIC code line ends with aリアル). 

Line 810 is a checkout line which may be removed later. It prints a just-input line on screen so you can see how it looks before any changes take place.

Line 820 finds the just-input line's length and adds one to it. Without that, lines ending with a line-number reference--such as '100 GOTO 1925' would lose '5', the last character.

Line 830 finds the space between line number and the first code character as done earlier in line 510.

Line 840 establishes LN as the integer value of the line number as done in line 520.

Line 850 checks the A array to see if the line number--but no reference numbers within it-- interacts with the starting number but less than the ending number. If so, it is changed, else we go on. A number is changed by making another temporary string T$, which is the MID$ of A$, starting at S, and going to the end of the line minus whatever characters S was. Next down the line we build a new A$, starting it with the string value of what was in the B array at I and adding T$ back to it.

Line 860 again finds the length of A$.

Now things get a bit complex! We did this earlier, why do it again? Because we could have replaced a line number two digits long with one 4 or 5 digits long--or the other way around--and the Q loop which follows must know exactly how long the line is. For now let's just say the Q loop will find line number references in A$ and change them. Let's first finish the 1-loop.

Assuming the Q loop has done its' job, line 970 will print the new A$ through buffer #2 to the new file. Notice that the Q loop would have changed the name from A$ to C$.

Line 980 is another checkout line which prints the changed line directly below the original line printed on screen earlier for you to compare. This line also may be deleted after you've satisfied your copy of the program works ok.

Line 990 sets C$ back to a null string and clears a couple flags which may have been set inside the Q loop.

Line 1010 CLOSEs all files after all lines in the target program have been checked and written back to the output file, then the filenames are printed on screen and the program ends.

THE Q LOOP --

To effectively renumber any or all of a program you must examine ever line of it because there may be references into or out of the renumbered area. And there may be references within the renumbered area which refer to other lines within it.

The 1-loop brought in one line and changed its' line number if necessary. Now the Q loop examines that line in detail for any line references within which must be changed.

Line 870 starts the Q loop by reading the line of code from the line's first position to its' end, one character at a time.

Line 880 lets CS equal itself plus the next character via the MID$ of A$, starting at Q for 1 character. Here is an example line of how it works, assuming A$ as this line:

100 IF A=F THEN 850

As the Q loop increments, C$ looks like this:

1
10
100
100
100 I
100 IF A
100 IF A-
100 IF A=5
100 IF A=5
100 IF A=5 TH
100 IF A=5 THEN

As C$ builds, we let S$ equal the right four characters of C$ in line 990. As each character gets added to S$ various checks are done in lines 900 thru 950 to see what the line contains.

[ more ]
Line 900 checks to see if the two right characters of $A$'s right four characters equal 'ON'. This will be used later in the ON...GOTO and ON...GOSUB cases. If ON is found, line 900 sets flag F1 to 1, otherwise the loop proceeds and looks for other distinctive keywords. There are THEN, ELSE, GOTO, and GOSUB. ("GOSUB" instead of "GOSUB" to differentiate between it and GOTO and to keep all keywords four characters long.)

Line 910 finds that $A$ does equal THEN, the Q-index number is at 15 (count the numbers and spaces in the last line of our example) and further, there is a number following THEN so we go to a subroutine at line 1070.

Note that in any BASIC program, keywords THEN and ELSE do not necessarily need to be followed by a line number, ad in "THEN A=4" or "ELSE$=MID$(A8,P,4)", but they may be followed by line numbers. GOTO and GOSUB are always followed by line numbers.

FIND/REPLACE SUBROUTINE --

We wouldn't have gotten here from loop Q if we hadn't encountered one of the keywords earlier. But in our example we found "THEN", so we need to know what follows it. The first thing we do in line 1070 is let $A$ = the integer value of whatever follows the "A" in THEN. To do that we look at the MID$$(A$, starting at Q1 (Q was at the 15th position) for then next 9 characters. Why Q+1? Because looking at Q would give us the integer value of "$A" which would be zero. And why look 9 characters ahead? Because the number (if any) could be as long as five characters, and some folks like to (or accidentally) put extra spaces in their lines.

Let's briefly examine VAL; it's a very interesting function which returns the integer value of numbers which are in string form. If there're no numbers in the string VAL=0. VAL reads only numbers up to but not including the first non-numeric character, and it ignores spaces. The VAL of string "A123" is zero because of the letter "A", and of string "12:30" = 12 because of the colon. In the above example, if we didn't increment Q by 1, the VAL of characters following "THEN" would be 0 because of the "$A", but VAL of Q+1=850.

Line 1080 jumps to line 1150 and we go back where we came if we don't find an integer value starting at Q+1, and C5 will continue building as we look for other keywords. But we found "850" following "THEN"; what to do with it? Either replace 850 with a number from the B-array--if it's in the renumbering range--or put it back if not.

In Line 1090 we let Q=INDEXing value of position of string value $A$ (850) + length that string $A$ would make less 1. So Q will be repositioned in $A$ at the next position following the number just removed.

The minus A removed the extra space that STER$ put ahead of STER$(A), otherwise Q would be too far down the line and miss the first character following the number being replaced. In the example this doesn't happen, but what if the line read "100 IF A=-5 THEN GOSUB B50;GOTO 130"? That colon is very important and we don't want to miss it.

Line 1100 jumps out to a subroutine at 1330 which scans the entire A array for a match of "850". If there's a match we go back—it's a valid reference; if not, we get "Undefined IX in XX" on screen and a "?" mark between two quote marks in C$ and then the question mark in AI where the undefined line reference was. We then return to the earlier subroutine which sent us here.

In line 1110, having taken care of line reference validity, we now check if our number (850) is in the renumbering range. If it is, line 1110 sends us to line 1120. This may seem odd, as BASIC usually "falls through" to the next line, but we definitely want the loop between 1120 and 1140 to search the A array for "850" and this is the most straightforward method to find WHERE "850" is.

In line 1120 index M tells us where to get our new number from the B array, and in line 1130 we add the STER of the B-array number to C$ and return to line 920.

If our "850" wasn't in the renumbering range then line 1110 simply adds the STER of the original number back to C$, then goes to 1150 to return to 920.

Again back in the Q loop of lines 870 thru 960 we check for more keywords and make appropriate changes as necessary.

THE ON...GOTO PROBLEM --

In line 900, if we encounter "ON", we set flag F1 to 1. If in the same line we later encounter keywords "GOTO" or "OSUB" we set flag F2 to 1. The first number following an ON...GOTO or ON...GOSUB will be treated as described above. But numbers subsequent to the first can't be handled so. RENUMBER assumes there is nothing following the list of numbers after an ON...GOTO or ON...GOSUB.

In line 950 if both flags are set to 1 we know there's an ON...GOTO or ON...GOSUB situation in the subject line and we then jump to the subroutine at 1100.

At 1180 we use loop (X) to find the commas in the subject line. Variable P gives the first comma's position. If P=0 it means there're no more numbers and we return via line 1300, else if there's a comma, line 1210 looks for the number after it (we've already taken care of the 1st number). Again, if A=0 we return. The subroutine at line 1330 checks to see if
the number is a valid line number as previously described, then replace the A array number with the B array number, or restore it, if it's not in the area to be renumbered.

There is a slight difference here. In lines 1240 and 1260, rather than making the number's STR$ we strip off the leading space by using MID$, starting at position two. If we didn't, the number we put back would overwrite the trailing comma.

In line 1280 we increment the X index to the most recent comma's position. "NEXT" moves X by one past that position, where "" again looks for the next comma. We build the entire remainder of C$ thusly and return to line 950 where a hard SOTO takes us around the NEXT $ in 960 because we're finished with this line.

LIMITATIONS --

RENUMBER.BAS makes no check to see if a new line created by C$ becomes longer than 255 characters. If you use very long lines in the original target program, new line numbers (if larger than the old ones) may push line-length limits and you'll get a "String too long" error.

There are some additional keywords which may be followed by line-number references. You may want to add some such as these: RESTORE, RESUME and RETURN (some BASICS allow the use of RETURN with a following line number). To add any of these keywords, put them between lines 920 and 930. Use the last four characters of a command as in this example: 922 IF $x="TORE" THEN GOSUB 1970.

Compiling this program will improve it's execution speed. We tried compiling it with Microsoft Quickbasic compiler and got a reduction from about four minutes to less than 45 seconds on a 100-line program. There is no guarantee that RENUMBER.BAS will work on programs where there are no spaces around key words (as in MBASIC V4.82 for HDOS--ed.), but we've tried it on long lines of tightly packed code to see if it worked, and were not disappointed. Now if the program is finished, you may see many other possible ways to accomplish the same job.

[I found it took about six minutes for RENUMBER to run with the check lines left in, and well under 3 minutes with check lines and all REM lines removed when I ran it on a 100-line test program. That isn't bad at all, considering the hours it takes to break a program into separate blocks, renumber them, save in ASCII and then MERGE the program back together. And this program is free of all the horrible mistakes which creep in when using MBASIC's RENUM "futility". ---- ed]

The listing for RENUMBER.BAS starts in the next column.
INPUT "With what increment";IC:IF IC=0 THEN IC=1
PRINT 
MSG$=" Renumbering program ":
PRINT TAB(40-LEN(MSG$));"V2$;MSG$;V"

REM Read file & put its' line numbers into A() array.
OPEN "I",1,F$
FOR I=1 TO ML
IF EOF(I) THEN S550
LINK INPUT #I,A$
S=INSTR(A$," ")
LN=VAL(LEFT$(A$,S))
A(I)=LN
NEXT I
CLOSE

REM Read A() array and put new line numbers in B() array.
S=S+LN:CT=0
IF A(I)<SN AND A(I)<EN THEN B(I)=SU+IC:
SU=SU+IC:CT=CT+1
NEXT I

REM Check to see if what you intended doing will work.
IF (SN+CT+IC)<EN THEN 710 ELSE
PRINT "It won't fit! Reduce the increment";
PRINT " or less to prevent line clashes."
INPUT "Enter new increment ";IC:IF IC=0 THEN IC=1
FOR I=1 TO N:B(I)=B(I):NEXT I:GOTO 610
FOR I=1 TO N:PRINT I,A(I),B(I):NEXT I
RETURN

REM Set up name for the new output file.
IF INSTR(F$,," ") THEN X=LEN(F$):TPS=MID$(F$,1,X-4):
F$="TP$+"NEW" F$="TP$+"NEW"
NEXT I
REM Now read file again and substitute line numbers.
OPEN "O",2,F$
OPEN "I",1,F$
FOR I=1 TO N:REM
LINK INPUT #1,A$
PRINT A$
L=LEN(A$)+1
S=INSTR(A$," ")
LN=VAL(LEFT$(A$,S))
IF LN<SN AND LN<EN THEN T$=MID$(A$,S,LN):
A$=STR$(B(I))+T$
RENUMBER,A$
FOR Q=1 TO L
C$=C$+MID$(A$,Q,1)
S=RIGHT$(C$,4)
NEXT Q
REM Find & replace number after keyword and increment Q.
A=VAL(MID$(A$,Q+1,9))
IF A=0 THEN 1150
Q=INSTR(Q,A$) STR$(A)+LEN(STR$(A))-1
GOSUB 1330
IF A<EN AND A<SN THEN 1120 ELSE C$=C$+STR$(A):GOTO 1150
FOR N=1 TO
IF A=A(M) THEN C$=C$+STR$(B(M))
NEXT M
RETURN
REM Take care of the "On...Goto" and "On...Gosub" cases.
FOR X=Q TO L
P=INSTR(X,A$,,")
IF P<>0 THEN 1330
A=VAL(MID$(A$)P+1,9))
IF A=0 THEN 1300
GOSUB 1330
IF A<SN AND A<EN THEN 1250 ELSE C$=C$+","+MID$(STR$(A),2):GOTO 1260
FOR N=1 TO
IF A=A(M) THEN C$=C$+","+MID$(STR$(B(M)),2)
NEXT M
X=P
NEXT X
RETURN
REM Find unreferenced line numbers
FOR M=1 TO N
IF A(A(M)) THEN M=N:GOTO 1380
NEXT M
PRINT "Undefined line:"A$"in":LN
C$=C$+CHRS(32)+CHRS(34)+CHRS(63)+CHRS(34)
RETURN
END:REM
REM Buy RENUMBER.BAS w/Ace of Aces on SEBHC GAMES DISC #0--$6.98!
You already own half of a great printer...
Are you still looking for the other half?
Look no further!

Introducing: Dots-Perfect™
an upgrade kit for
Epson® FX, JX,
RX, and MX printers

Dots-Perfect™ is here
at Super Discount Prices
Just $63.00 + $2.00 S/H

For orders send check or money order to:
Myra D. White
849 E. Bonita Ave.
La Verne, CA 91750

California residents add $4.10 sales tax

(714) 592-6897 after 6PM PST

All orders shipped within 48 hrs of receipt
You won't want to use MailMerge without the file-juggling muscle of Filebase.

Creating a database with Filebase is a cinch.

- COMPATIBLE with MultiMate, MailMerge, WordStar, InfoStar, dBase, BASIC, etc.

Data Based Advisor
AN EASY-TO-USE DATA MANAGEMENT TOOL
- POWERFUL SORTING & RECORD SELECTION
- VARIABLE LENGTH FIELDS

I was able to produce a neatly formatted checkbook register in one try with no wasted paper.

Dr. Dobb's Journal
For Users of Small Computer Systems

EASY-TO-USE
- 100% MENU & PROMPT DRIVEN!

"Top 50 New Business Software Packages
A Low-Cost File Manager For Work and Home"

$125

EWDP Software, Inc. P.O. Box 40283C Indianapolis IN 46240 (317) 872-8799
Divine Profit

"St. Silicon" Preaches to Computer Converts

by David M. Poppe in the Arizona Business Gazette

PHOENIX, Ariz. -- He calls himself St. Silicon, the patron saint of appropriate technology and founder of the first computer religion, C.H.I.P.--the Church of Heuristic Information Processing.

With a microchip glued to his forehead, he explains that his new religion is "user friendly" and that he is committed in his search for the "Divine Profits".

It's all in a day's work for Jeffery Armstrong, a former international sales manager for Apple Computers in the Middle East and now a high-tech evangelist and stand-up comic.

Here recently to deliver the keynote address at a convention of Micro-Age Computer Store dealers, St. Silicon spread the word as "it was downloaded unto me from the Giver of Data (G.O.D.)."

Asked how it was that he was chosen to found C.H.I.P., St. Silicon explains it was destiny. "My first computer came with a loser's manual," he says. "I was a good blank template for G.O.D. to use."

Now, calling himself the "Fourth-Quarter Profit", and touring the country to preach the DOSpel from the Binary Bible, St. Silicon proudly says his is the world's first "for-profit" religion. He maintains he was divinely inspired to pursue his calling four years ago.

"I was sitting in front of my Macintosh one night when lightning struck the satellite dish, knocking me unconscious," he recalls. "When I awoke, the 'Keyboard Prayer' was on the screen, and I was aware that I had been called into the service of the Giver of Data to spread his Disc-pensation among the carbon-based entities."

While St. Silicon's act is all in fun, Armstrong insists there is a more serious message behind his Garbage-In, Gospel-Out religion.

In addition to his work with Apple, the 39-year-old mock preacher also works for Corvus systems and Nestar in the computer industry. He has also earned degrees in psychology, creative writing and history. Between quips he says his "preaching" helps humanise an increasingly technological world.

"I like to think that I help heal the wound created by excessive technology," Armstrong says.

"The computer is a cultural artifact but it has arisen so quickly the human aspect has been overwhelmed. Most people don't understand computers and are uncomfortable with them. I try to help people be less afraid of the technological unknown."

St. Silicon, usually dressed in a cream-coloured suit, does his stand-up routine at computer dealer conventions and other shows throughout the country. He gives the "Sermon on the Monitor", reads such proverbs from the Binary Bible as "the Mac-righteous shall inherit the earth", and leads DOSciples in such prayers as "Hail Memory" and "The Keyboard Prayer".

St. Silicon also teaches fear of the evil one, Glitch, to his DOSciples.

Then there are the tales of the splendid "Winchester Cathedral" in Silicon Valley. The Valley is the heartland of Armstrong's computer religion movement, and according to the Binary Bible, "the yuppie centre of the universe."

While some have taken offense at the religious angle of Armstrong's satire of the computer world, he says most people understand the parody and enjoy it.

Armstrong says his act is not intended as a slap at religion but as a joke at the expense of those who put all their faith in technology.

As he puts it, "My followers and I shall reach Nerdevana."

---0---

(Editor's note: Editorial assistant Ziggy Nebbish sent us the above article from a discarded copy of the Arizona Business Gazette he'd found in an Aero Mexico plane's lavatory while on an SEBHC JOURNAL editorial assignment in the southwest. We thought it was worth printing, even though we've had a very difficult time deciphering it; the paper was badly discoloured and torn—possibly by some computer phone! We hope you get as big a kick from reading it as we did!)

SUPPORT THE EIGHT-BIT H/Z VENDORS ADVERTISING IN YOUR JOURNAL!
ADDENDUM/CORRECTIONS TO "Ace of Aces" MBASIC LISTING
(Volume 1, Issues 6 and 7)

Here is an upgrade for the Ace-of-Aces game options listing in Issue 7. We printed it full size so it will be easier for you to copy. Note especially the routine in lines 191 - 198 (works only with MBASIC V5.2 and up) which limits keyboard response to just four acceptable characters, 'YnNo'. When this routine prints its message, it will wait forever until you enter one of the acceptable characters, then it either goes to the subroutine at line 4000 or falls through to line 199 and then gets on with the game.

For those of you who have received the original game disc and want to have us patch these lines into your copy AT NO CHARGE OTHER THAN RETURN POSTAGE, please send us your disc and we'll get it back to you right away.

All new GAME DISC #0 copies are being shipped with the upgraded Ace of Aces, plus the corrected assembly-language utilities which were misprinted in Issue 7. We know you will like the way they assemble and run. (.COM versions also included.)

---

Insert these lines:

82 GOTO 191:REM
191 TXT$:="Want to review game commands (Y/N) ?":PRINT TAB(40-LEN(TXT$))"2"
192 TXT$:="Yes, go on":PRINT TAB(40-LEN(TXT$))"2"
193 P=INSTR("YnNo",X$)
194 IF P=0 THEN 192:REM
195 IF P=1 OR P=2 THEN GOSUB 4000:REM
196 IF P=3 OR P=4 THEN PRINT:PRINT:GOTO 199:REM
199 PRINT CLS$:TXT$="OK, now let's have some FUN!!!":PRINT
200 TAB(40-LEN(TXT$))"2"TXT$:FOR I=1 TO 1500:NEXT:PRINT CLS$

Add these lines -- Note new line numbers:

3999 REM
4000 PRINT CLS$:PRINT:PRINT
4010 TXT$="Table of Keypad Commands":GOSUB 5000
4015 PRINT
4020 TXT$="All commands must be followed by keypad ENTER key.":GOSUB 5000
4025 PRINT
4030 PRINT TAB(11)"0 - Fire gun, fly straight"
4040 PRINT TAB(11)"1 - Slow left turn to 8:00"
4050 PRINT TAB(11)"2 - Slow hard left w/drop"
4060 PRINT TAB(11)"3 - Slow dodge R + left turn"
4070 PRINT TAB(11)"4 - Slow dodge left"
4080 PRINT TAB(11)"5 - Slow ahead, hold altitude"
4090 PRINT TAB(11)"6 - Slow right"
4100 PRINT TAB(11)"7 - Slow right to 2:00 w/drop"
4110 PRINT TAB(11)"8 - Slow hard right to 10:00"
4120 PRINT TAB(11)"9 - Slow left dodge + R turn"
4130 PRINT TAB(10)"10 - Left turn to 10:00"
4140 PRINT TAB(10)"11 - Hard left to 8:00"
4150 PRINT TAB(10)"12 - Dodge left to 2:00"
4160 TXT$="26 - Fast right to 1:00":GOSUB 5000
4170 FOR I=1 TO 7000:NEXT I:REM
4180 RETURN
4999 REM
5000 PRINT TAB(40-LEN(TXT$))"2"TXT$:RETURN

NOTE: SEBHC GAME DISC #0 Contains all listings published to date.

"Self-centered" printing routine
Dear Mr. Geisler,

I must confess that I had never heard of the SEBHC JOURNAL until I read Kirk Thompson's letter in the current REMark. As an '89 user, I was getting increasingly disenchanted with the irrelevance of REMark's contributors to my needs. Coincidentally, my HUB subscription was due.

Hence I thought this seemed the right moment to take out a subscription to the SEBHC JOURNAL. Please find my check enclosed.

Sincerely,

John Brooke

-----------------------------------------------

Dear Sir:

Having lived too long in an informational desert with REMark, it is time to seek other sources. Please find my check for a subscription to the SEBHC JOURNAL enclosed.

Allan C. Erno

[Thank you very much gentlemen for your kind words about our Society's JOURNAL, and welcome to the Society ranks! We started this endeavor for reasons similar to those which prompted you to subscribe and join, namely, the heavy mess of big-bluish, NON-Eight-Bit articles in otherwise excellent H/I-oriented publications. We at JOURNAL headquarters are not against change as long as it is evolutionary and "improves the breed or product" (cf the H/I-100s, H-1000, etc.). Having our favorite machines callously abandoned by self-styled "businessmen" cashing in on artificially-induced enthusiasm for an inferior, blue-tainted product isn't seen as true upgrading change or actual progress by REAL eight-bitters! But that's a story we've told before...]

Len,

Thanks again for your help. I'm "on the air" at four megaflops with my H89A and it is "WUNDERFUL"! I think it beats the IBM 3270 at work for speed and comfort.

Dave Hart

[ Glad to have been of assistance, Dave. Us 8-bit users very definately have to stick together. And welcome to the Society of Eight-Bit Heath Computerists! ]

Dear Leonard:

I sure appreciate your paper! It's one of the few periodicals I eagerly read completely on the day it's received.

Do you have any influence with Lee Hart? I sent him a check for $49.75 for his 19-89 SuperSet upgrade kit and have still not received it, nor have I gotten acknowledgement to two queries I sent.

Here are some items I'd like to find via the JOURNAL:

1 - Source of individual colored ribbons for Diablo 638 printers. Prefer red and blue. Local suppliers sell only in dozen

2 - Large quantity of 8-inch ddss discs, new or used, at a reasonable price.

3 - Large English dictionary (100,000 words or better preferably) as ASCII files. Disc format doesn't matter.

4 - Synonym and antonym dictionaries or word lists as ASCII files. Disc format no problem.

5 - A scanner to read typewritten material that will operate with an H/I-89 or '90 computer.

6 - An H/OS program to dup any kind of disc regardless of format or operating system on disc to be copied. Desire such for both 5-1/4-inch and 8-inch discs.

7 - Does anyone know of a Keymap-type program in H/OS for [function] key redefinition that will run within application programs?

Terry Hall, Wheaton, IL 60187 -- 312-665-4594

[Sure! Send for our GAMES DISC #6 and include your cheque for $6.95. We'll ship it at once upon receipt of your order. And you other members should help Mrs. Guild find more games.]

SEBHC JOURNAL

Volume 1, Number 3 Page 15

Still MORE BOUQUETS & BRICKBATS!

Dear SEBHC JOURNAL Editor:

I recently bought a 2M-100 (iba compatible) for home use but my kids use it for "fun & games" more than I do seriously, i.e., writing the monthly newsletter for an organisation which I serve. Someone suggested that I get a modem and "download" games from the various services but I don't see a return equal to the cost of that investment! My kids are becoming disenchanted with me and the '90s. Can you help a fellow SEBHCer?

Mrs. Carol Guild, Techu, MI 49266 -- phone 517-423-4568
ATTENTION SOFTWARE AUTHORS

Our SEBHC members need HDOS & CP/M Games and Utilities -- Right Now!

Send us your game or utility for inclusion on either GAME DISC #0 or GAME DISC #1 (pending)! If your work is satisfactory, we will PAY you half the net profit from the sale of each SEBHC GAME DISC which has your game, program or utility on it! That means you’d get about $2.35 for every one sold. At present we’re selling one disc a week—your potential earnings could be around $10/month if you send us even ONE saleable item. What are you waiting for??!!

Above -- Portrait of YOU, a new SEBHC Software Author hard at work!
Reviews, editorial references, and advertisements in the SEBHC JOURNAL should not be taken as authoritative endorsements of any products or services. Opinions expressed herein are based on the individual's experiences and shall not be considered as official endorsement or certification in any way, nor do they reflect intensive technical analysis as might be provided by a professional testing firm. Although we do not knowingly publish fraudulent materials, we shall not be held liable for any damages arising from purchase or use of any product. Readers having complaints about goods or services purchased from our advertisers are urged to send us written notification of their specific complaints so that we may take any action which we deem appropriate. Caveat emptor!
* The SEBHC JOURNAL is published twelve times a year and is mailed on or about the 22nd of each month. Editorial deadline: 20th of each month.

* All advertising is printed free of charge. Vendors, please submit B&W “camera-ready” ad copy, 7" wide by 9" high (one page/issue) no later than the 15th of a month in which it’s scheduled to appear. Society Members get a free (new) 250-word want ad in each issue.

* Subscriptions are $15/year as of 1-Jan-87 in Canada, Mexico, the U.S.A. and possessions, and begin on month following application receipt. Make cheques or money orders payable to L.E. Geisler. Single back-issue copies now available on special order only—allow 6 weeks for processing.

* Subscribers automatically become members in the Society of Eight-Bit Heath Computerists. Member’s ID number follows their name on mailing label. Any REGULAR member can vote and hold any Society office. There are three member classes: REGULAR (H/Z 8-bit user), ADVERTISING (one vote each vendor), and ASSOCIATE (non-8-bit computerist, library, etc.). ASSOCIATE members do not hold office or vote in Society elections.

* The SEBHC JOURNAL is composed, edited and printed by L.E. Geisler at 895 Starwick Drive, Ann Arbor, MI 48105. Phone 313-662-0750, 9am – 6pm EST M-F. "Record-a-call" nightly, week-ends, holidays; message time about 50 secs.

SEBHC Journal

895 Starwick Drive
Ann Arbor, MI 48105