SCELBAL UPDATE

ISSUE 01 - 8/76

WHAT IS IT?

So what is this little publication titled SCELBAL UPDATE supposed to be? Well, first of all it is just what its title denotes. A means of keeping registered SCELBAL owners up to date on the status of the program in regards to the correcting of "bugs" that might appear, additional operating information that may be of interest to owners, clarification of points raised by users and so forth. More than that, however, this publication is sort of an experiment. It is an experiment to determine just how much our readers would like to participate in the process of refining the fundamental program as it has been presented in the SCELBAL manual, or participate in the creation and sharing with others, of application programs written to run using the SCELBAL interpreter.

The potential for tailoring a package such as SCELBAL to a wide variety of applications, of adding additional features, of improving its operating efficiency, is virtually endless. Are you, the users, interested in seeing this done? Do some of you want to participate in the arena? Would you like to have a vehicle such as this through which you could communicate with other users? Would you like to join with the program authors in improving and adding to the program's capabilities? Would groups of you like to work on specific sections? Would you like to have a medium for the presentation of application programs that use the language. Do you want to see application programs for games, or would you prefer programs that have more practical applications - such as programs for handling business, scientific and engineering problems?

You, the individual readers, are the ingredients in this experiment. It is you who will determine in what direction(s) the experiment goes and what conclusions may be arrived at!

Write us, tell us what you think, send us you suggestions, tell us what you are interested in, remit your program ideas, send us application programs written in SCELBAL!

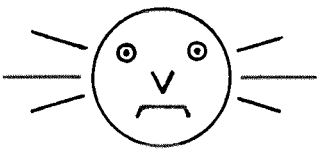
(To avoid any possible squabbles, lets have it understood that submissions do become the property of SCELBI C. C., INC.. However, we shall point out that to sort of provide a little incentive, submissions we find worthy of publication will receive an honorarium payment, which will, we are sure, more than cover the postage for such submissions.)

How far could this thing go? As has been said, that is up to you. We are simply providing the opportunity. We will be providing three or four issues during the next six months or so as a service to our SCELBAL customers. If, at the end of that time it appears there is a sufficient base to support the concept, we are prepared to implement it on a subscription basis. If not, then, at least, we will have learned something from the SCELBAL UPDATE experiment, and, we are sure, so will have you!

You may address your comments on this matter, along with submissions to be considered for publication, to:

SCELBAL UPDATE EDITOR
SCELBI C. C., INC.
1322 Rear - Boston Post Road
Milford, CT 06460

IS OUR FACE RED!



We pride ourselves at SCELBI on accuracy. It is tough - preparing complex programs in the form of books - making sure that source listings and object listings get transcribed from computer print outs to type set without errors. For instance, three separate "proofers" spent countless hours checking to ensure that the critical object code listings in chapters 12 and 13 of the SCELBAL publications were absolutely perfect. After that, the typeset listing was used to verify proper operation of the program and to get an idea of how long it might take readers to implement the program on a computer using a keyboard loader. (Six to twelve hours for most, depending on how well they can handle a keyboard.) Even after all that checking it is a long wait between sending the copy to the printers and getting the first reports in from readers!

At this time, a number of customers have already reported that they have SCELBAL up and running fine - so we are finally satisfied with the "proofing" part of the job. The printed copy does agree with our originals.

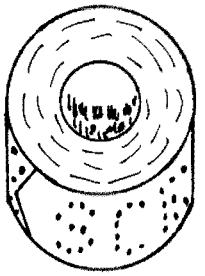
Unfortunately, no matter how good a job our clerical staff does in preparing a program publication, the program authors can blow it all when they goof!

Well, SCELBI has been producing such publications long enough to know that it is downright impossible to create a program the size and nature of SCELBAL and not find a few "bugs" or disagreeable features down the road after publication. That is the reason for providing some blank pages at the back of the book marked "NOTES." And, of course, a few bugs have shown up in SCELBAL at this point. These have been corrected by PATCH1 and PATCH2 which are pasted into the first edition of SCELBAL on the NOTES pages in the rear of the book prior to shipping.

The problem that necessitated PATCH2 did not show up until just a few days before the first lot of books were due to arrive from the printer. This meant, in order to ship promised books on time, that PATCH2 had to be created and rushed to print quite hastily! The program authors, in conference, quickly arrived at a suitable solution to the problem and created PATCH2. Author Arnold suggested that the patch be placed at the end of memory page 32 where there was plenty of room for such a patch. Author Wadsworth, aiming to "save such a "large" unused area for a REAL EMERGENCY??" thought he saw another location that the patch seemed to just perfectly fit into starting at location 224 on page 32 in memory! Since author Wadsworth had been designated as overall program manager for SCELBAL, the clerical staff hastily scurried to have the patch printed up to reside starting at that location IN A HURRY! Thus, PATCH2 arrived from the printer the same day that SCELBAL books arrived and were duly pasted in as books were packed for shipment.

Alas, as a number of our ever alert customers quickly noted,

(cont. pg. 3)



SCELBAL

just $25.00 ea.

PAPER TAPE

specify 8008 or 8080

OBJECT CODE AVAILABLE NOW!

SCELBAL AVAILABLE ON PAPER TAPE!

For several years now the company has been producing programs in the form of books - leaving it up to individual users to load programs into memory using keyboard loaders. In the past, with the majority of programs falling into the under 2K category, most readers were content with the "book only" delivery method. Apparently, going to a 7K program has bent a number of customers fingers out of joint. We have had quite a few request for paper tapes of the object code, and a number for the source listing.

We are going to start with making the object code available. (The source listing may be made available at a later date?)

One of the reasons the company has not been in any great hurry to start providing programs on paper tape was because of the lack of standardization of format. While there are still many formats in use, it is the consensus here at SCELBI that the Hexidecimal Paper Tape Format promulgated by Intel Corporation for use in their INTELLEC MCS* (*TM) is a suitable compromise among the many possibilities and one that is most familiar to industry and university users where the majority of the requests for such tapes appear to be coming from in our analysis.

Several features that the firm's staff considered worthy in this format include its frequent testing for reader errors and capability to recover from an error condition by simply backing up a few inches to the last block read successfully (instead of having to re-read an entire tape); the header style block format that allows different areas in memory to be loaded, and the fact that, when used with a typical ASCII teletype system, the tape itself can generate a hexidecimal listing of the data on the tape for checking and reference purposes.

Thus, it is being announced that the official standard at SCELBI for core images produced on paper tape for the firm's products will be the Hexidecimal format which is detailed below.

HEXIDECIMAL FORMAT for PAPER TAPE

The hexidecimal paper tape format that will be used by SCELBI for core images consists of the following.

A paper tape will contain one or more blocks of information. Each block will be a self-contained unit that includes a header containing information regarding the location of the information in the core area (an address), the amount of data contained in a block (a data byte count), a record type indicator, the actual data in hexidecimal notation, and a checksum. The start of each block of information will be indicated by a special character. All of the information within a block will be arranged in the order illustrated next on a row-by-row basis.

ROW 1 - Start of block mark consisting of the ASCII character code for the colon sign (:).

ROW 2,3 - Block length count consisting of two hexidecimal

characters (MSD then LSD). The block length count refers to the number of actual data bytes in a block. This value may be in the range 00 to FF (0 to 255 decimal). However, a count of zero (00) will indicate an END OF FILE block.

ROW 4 - 7 - Address at where data will begin to be loaded in memory expressed as four hexidecimal ASCII encoded characters. (High address then low address.)

ROW 8,9 - Type of block indicator. For standard core images this indicator will consist of the two ASCII encoded characters 00. Other types of indicators may be used in the future.

ROW 10.....X - Data. Each byte of data to be loaded into memory will be expressed as two ASCII encoded hexidecimal characters (MSD,LSD) requiring two rows on the paper tape.

ROW X+1, X+2 - Checksum. Expressed as the negative of the sum of the value of all rows in the block since the start of block marker (neglecting carries).

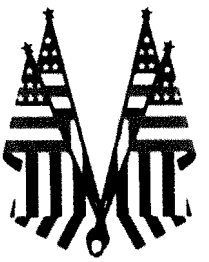
NOTE: Paper tapes punched in hexidecimal format will use the convention of not using the parity bit (eighth bit). This is opposite to the convention established for most SCELBI programs! The decision to follow the convention for the paper tape format was based on fostering compatibility and increased standardization, at least in the area of program loading capability!

PLEASE!!!

Do NOT write and ask us for SCELBAL on magnetic tape! We will not be supplying magnetic tapes until such time as we are satisfied that there is a fairly stable agreement concerning recording methods and formats. At this time we are watching the progress of the "K.C." standard closely. However, we feel it will be at least six months to a year, and possible longer, before standardization has set in to the degree that we will invest in the necessary equipment, personnel, etc., to start providing programs on magnetic tape.

BUT - you may write and ask for information concerning paper tapes of other SCELBI programs. We will soon be making paper tapes available for most of the programs presented in previous SCELBI publications - such as our Editors, Assemblers, Monitors, Games, etc.

NOTE - paper tapes supplied by SCELBI will be virtually useless if you do not have the corresponding publication! They are being made available as an optional supplement to the books - not as a replacement. Users will still have to provide I/O routines etc., as described in the related books and information regarding the locations of such routines, operating instructions, etc., will NOT - repeat - NOT be supplied with the paper tapes!



FEEL RESTRICTED BY BEING LIMITED TO 20 VARIABLES?

You shouldn't.....when it is so easy to essentially quadruple this capacity by using a set of elements in an array as individual variables! For instance, instead of using a group of variable names such as N1, N2,...N9; simply DIMension an array (in this case having nine elements) named N:

DIM N(9)

Then use the elements N(1), N(2),....N(9) as different variables. Using this technique you can add up to 64 more variables in a program for a total of 84. A program utilizing 84 variables will be a pretty "busy" program!

FA? FA? FA? FA?

Oops! We forgot to tell you something. While it is not mentioned in chapter fourteen (see the list on pages 19 and 20 in that chapter), nor is it shown on the handy pocket reference card included with the book (bound at the back with the registration card); the symbol FA is a valid SCELBAL error code! It means that the interpreter has encountered a Function or Array error condition.

Why not pencil in a little note to that effect on your pocket reference card? The error code is especially likely to come up if you do not have the DIMension capability included in your version of the program (and have substituted NOPs in the indicated memory locations) and then attempt to perform an operation that specifies an array element!

THINKING OF ALTERING PORTIONS OF SCELBAL?

Individuals planning to modify small sections or subroutines can probably do well enough using hand assembly methods. However, those who plan to undertake extensive revisions - such as, for example, compacting the program by taking advantage of the 8080's extra instructions - would do well to remember that SCELBI has assembler programs suitable for such tasks that operate in just 4K of memory (and can use memory beyond that amount to provide extensive symbol table storage). The SCELBI 8080 ASSEMBLER program is designed to process the mnemonics as they appear in the SCELBAL manual (original INTEL mnemonics for the 8008) as well as providing for the extended instruction set of the 8080 CPU. See SCELBI advertising literature for additional information.

(from pg. 1)

author Wadsworth's choice of location for PATCH2 overlooked the fact that locations 224 and 225 on page 32 were already occupied by the address bytes of the instruction JMP ERROR that would be executed if a SQuare Root error (negative argument) condition was encountered. Author Wadsworth, after mumbling something about "it was just a test to see if the readers were awake" agreed to relocate the patch to start at location 364 on page 32. A new "PATCH2 - Revised" was printed to replace the original patch number two. The revised version is included in books currently being shipped. Early customers who received the original patch will find a copy of the revised (simply relocated) PATCH2 enclosed with this literature which may be pasted over the original version - to erase all evidence ....as though the whole thing never occured!

SCELBAL UPDATE

ISSUE 02 - 9/76

MODIFIED SCELBAL

This is the beginning of a section that we plan to have on a regular basis in SCELBAL UPDATE. The purpose of this column will be to present modifications to SCELBAL that will provide some improved operation or desirable features to the fundamental program. Users are urged to contribute to this column.

In order to maintain some kind of overall organization of the fundamental program as various improvements are thought of, and suggestions for implementing those improvements made and/or contributed, it would be wise to lay out a few rules for contributors to follow. While these rules may not be considered as hard and fast at this point, they will at least serve as an initial guide. More "rules of the game" may become necessary as others join in the fun.

In the example modification to be described in this issue, the following rules were adhered to.

1. The improvement was made by altering the machine code within an address range delimited by labels.

2. The modification is essentially complete and self-contained within the boundary established in item number 1 above. That is, it was not necessary to "patch" the program by establishing subroutines external to the area modified.

3. The improvement does not rely on another improvement or modification. Adherence to this rule will insure that readers do not end up with a problem of having to refer to previous modifications ad infinitum. Note that this does not mean that a new contributor cannot modify an improvement. It simply means that the presentation should include all modifications and references to the original version of SCELBAL, and not the modifications. Of course, if in doing so one wants to reference an improved subroutine for purposes of discussion or to indicate a point of inspiration, one should certainly do so.

4. This column will relate only to improvements that can be implemented on an 8008 CPU based system. The optimization of SCELBAL for an 8080 is an entirely different matter which will be discussed at a later date.

5. The improvement does not alter the starting address of any label that is referred to by routines outside of the area being modified. That is, it should not be necessary to locate any references in subroutines outside of the improved area in order to implement the modification. Naturally, if the improvement or alteration does not require as many machine instructions as the original version, then NOP instructions may be inserted to the next label point, or a jump instruction may be used to continue operations to the next label point. Of course, if the improvement relates to a subroutine, then a RET instruction would be used to conclude the shortened program.

Following these initial guidelines should help to prevent chaos as contributors with various interests begin to point out ways in which the program may be improved, incorporate additional features, or possibly correct any potentially troublesome situations.

As pointed out in chapter 15 of the book, SCELBAL was deliberately published, not as a highly compacted, intricate, ultra-sophisticated program that would have been most difficult to explain and quite difficult to safely modify, but rather in a format that was more conducive to explanation and alteration. The reader with a minimal amount of machine language programming capability will be able to find all kinds of ways in which various portions of SCELBAL might be modified to suit individual taste. The range of modifications that one can envision are virtually too numerous to enumerate. Some readers might be interested in studying ways in which to speed up the operation of various sections of the program. Other users might be interested in adding "bells and whistles" to the program. Still other readers might be interested in finding ways in which to considerably compact the amount of memory the program utilizes. (Again, reference here is made to the 8008 version. Obviously, SCELBAL can be considerably compacted if the 8080 instruction set is capitalized upon. As pointed out earlier, however, that matter will be handled separately from this column.)

The modification to be discussed in this issue can be classified as a "bells and whistles" feature.

Have you ever created a SCELBAL program and inadvertantly used more than 20 regular variable names? If so, you probably did not discover your error until you attempted to run the program and received a BG error message. After some head scratching, when you finally figured out that the problem was caused by too many variable names, you attempted an easy solution by combining mathematical statement lines to reduce the number of variable names. Alas, however, you discovered that after modifying the program you were stuck in a nasty situation. Every time you tried to run the program that BG error message came back again. Why? Because eliminating a variable name from a program statement does not eliminate that variable name from the variables table. The variables table remains filled. How does one normally get out of that situation? By use of the SCR command. Unfortunately, while this command does indeed clear out the variables names table, it also clears out the user program buffer, making it necessary for the programmer to re-enter the revised program. This may not be so difficult if the user has high speed bulk storage facilities and can utilize the LOAD command. Nor is it tough if the program is relatively small. However, in most cases a program overflowing from excessive variable names will have been a relatively large program and re-entering it by keyboard may be a little frustrating.

A user that has really studied SCELBAL and that has a resident Monitor facility on their computer system might discover that a shortcut to getting out of that type of situation would be to use the Monitor program to initialize the variables table to the effectively empty condition. This can be accomplished by placing a zero byte at the start of the regular variable symbol table (which is at address PG 27 LOC 210), and, re-initializing the value in the variables counter at PG 27 LOC 077 to a value of 001.

That action is one of several that is performed when a SCR command is issued. But, the SCR command also results in the user program buffer being effectively erased. It might be nice if one could have two types of initializing commands. One would be an all-inclusive

initializing command just like the SCR command; the other would be a special command that only initialized the variables symbol table.

The modification presented herein provides that capability by replacing the SCR command with two single letter commands. One single letter command signified by the letter S for "scratch" provides the all-inclusive initializing capability for the interpreter. The second command signified by the letter E for "erase" allows the programmer to effectively erase just the variables symbol table while leaving the user program buffer intact.

This improved capability can be provided by modifying the section of SCELBAL that starts at the label NOLIST and ends with the label NOSCR. The source listing for the original version of this section is discussed in chapter 4 on pages 5 and 6. The area in the assembled listing starts on PG 10 LOC 354 and ends at PG 11 LOC 066.

The source listing of the modification that follows illustrates how the improvement was affected by re-organizing the order in which specific initializing actions were taken; splitting the original SCR command in the command look-up table into two character strings, one containing a S, the other an E; and "tightening up" the program a little bit by ascertaining the possible contents of the D and E and the H and L CPU registers whenever the program returned from the STRCP subroutine.

Assembled object code listings of a modification for both the 8008 and 8080 processors are presented on the following page.

To operate the modified version, simply remember that the SCR command has been replaced by the single letter command S. Additionally, a new command, invoked by entering an E followed by a carriage return when in the executive mode, will cause the array and regular variable symbol tables to be effectively erased without disturbing the contents of the user program buffer.

SOURCE LISTING

```
/
ORG 001 346
/
001                 /(CC) FOR 'E'
305                 /E
001                 /(CC) FOR 'S'
323                 /S
/
ORG 010 354
/
NOLIST, LLI 342     /SET H&L TO ADDRESS OF 'RUN'
LHI 001             /** IN COMMAND LOOK UP TABLE
LEI 000             /SET D&E TO ADDRESS OF START
LDI 026             /**OF LINE INPUT BUFFER
CAL STRCP           /COMPARE STRINGS
JTZ RUN             /GO TO 'RUN' ROUTINE ON MATCH
LDI 026             /**RESET D&E TO START OF
LEI 000             /THE LINE INPUT BUFFER
LLI 346             /SET H&L TO ADDRESS OF 'E'
LHI 001             /** IN COMMAND LOOK UP TABLE
CAL STRCP           /COMPARE STRINGS
JTZ HAVEE           /IF MATCH, HAVE 'ERASE' CMND
LEI 350             /ELSE, SET PNTR TO ADDR OF 'S'
LLI 000             /SET PNTR TO START OF INPUT BF
CAL STRCP           /COMPARE STRINGS
JFZ NOSCR           /IF NO MATCH, CONTINUE PGM
LLI 364             /ON MATCH, POINT TO USER PGM
LMI 033             /†† LINE PNTR & INITIALIZE TO
INL                 /STARTING ADDRESS OF THE BUFF
XRA                 /FIRST THE HA (PG 33) THEN THE
LMA                 /LA (LOCATION 0P0) PORTION
DCL                 /NOW SET H&L TO ADDR OF START
LHM                 /OF USER PROGRAM BUFFER
LLA                 /AND INITIALIZE THE BUFFER
LMA                 /WITH A ZERO BYTE
/
HAVEE, LHI 027      /** SET H&L TO ADDR OF THE
LLI 075             /NUMBER OF ARRAYS COUNTER
XRA                 /CLEAR THE ACCUMULATOR AND
LMA                 /INITIALIZE THE COUNTER
LLI 120             /NOW POINT TO START OF ARRAY
LMA                 /VARIABLES TABLE - INITIALIZE
LLI 210             /NOW POINT TO START OF REGULAR
LMA                 /VARIABLES SYMBOL TABLE - INIT
LLI 077             /POINT TO VARIABLES COUNTER
LMI 001             /INITIALIZE TO COUNT OF ONE
LHI 057             /•• POINT TO START OF ARRAYS
LLA                 /•• STORAGE PAGE
/
SCRLOP, LMA         /•• FORM A LOOP TO
INL                 /•• CLEAR OUT ALL LOCATIONS
JFZ SCRLOP          /•• IN THE ARRAYS STORAGE AREA
JMP EXEC            /BACK TO EXECUTIVE WHEN DONE
/
HLT                 /SAFETY HALT FOR UNUSED BYTE
/
```

8008				8080			
01 346	001			01 346	001		
01 347	305			01 347	305		
01 350	001			01 350	001		
01 351	323			01 351	323		
10 354	066 342			10 354	056 342		
10 356	056 001			10 356	046 001		
10 360	046 000			10 360	036 000		
10 362	036 026			10 362	026 026		
10 364	106 332	002		10 364	315 332	002	
10 367	150 070	013		10 367	312 070	013	
10 372	036 026			10 372	026 026		
10 374	046 000			10 374	036 000		
10 376	066 346			10 376	056 346		
11 000	056 001			11 000	046 001		
11 002	106 332	002		11 002	315 332	002	
11 005	150 035	011		11 005	312 035	011	
11 010	046 350			11 010	036 350		
11 012	066 000			11 012	056 000		
11 014	106 332	002		11 014	315 332	002	
11 017	110 071	011		11 017	302 071	011	
11 022	066 364			11 022	056 364		
11 024	076 033			11 024	066 033		
11 026	060			11 026	054		
11 027	151			11 027	257		
11 030	271			11 030	167		
11 031	061			11 031	055		
11 032	357			11 032	146		
11 033	261			11 033	157		
11 034	271			11 034	167		
11 035	056 027			11 035	046 027		
11 037	066 075			11 037	056 075		
11 041	151			11 041	257		
11 042	271			11 042	167		
11 043	066 120			11 043	056 120		
11 045	271			11 045	167		
11 046	066 210			11 046	056 210		
11 050	271			11 050	167		
11 051	066 077			11 051	056 077		
11 053	076 001			11 053	066 001		
11 055	056 057			11 055	046 057		
11 057	261			11 057	157		
11 060	271			11 060	167		
11 061	060			11 061	054		
11 062	110 060	011		11 062	302 060	011	
11 065	104 266	010		11 065	303 266	010	
11 070	000			11 070	166		

OBJECT

REGISTER YOUR COPY OF SCELBAL NOW!

The first two issues of SCELBAL UPDATE were sent to all purchasers at their purchase addresses. It cost a considerable amount of money to send out copies of SCELBAL UPDATE. Future copies will only be sent to those purchasers who have registered their copies of the publication. The registration card may be found on the last page of your SCELBAL book.

A PLUG FOR CREATIVE COMPUTING

The game presented on the next page is a slightly revised version of a program that appeared in an excellent magazine that is appropriately named CREATIVE COMPUTING. The magazine is published by an enthusiastic and creative organization headed by David H. Ahl. In addition to games such as that shown in this issue, the magazine regularly presents a variety of articles, book and product reviews, educational material, and a good selection of general information which we feel most of our customers would find highly interesting. Recent issues of the magazine contained 88 pages (8½ x 11). Considering the fact that relatively little advertising appears in those 88 pages, the amount of text and editorial material per issue far exceeds any other computer-related publication that has come to our attention to date. Individuals interested in subscribing to the publication may do so at the following rates. 1 year - $8.00, 3 years - $21.00. If you have any doubts, you can get a sample copy for $1.50. (The magazine is issued bimonthly.) Subscription orders should be forwarded directly to:

CREATIVE COMPUTING
P. O. Box 789-M
Morristown, NJ 07960

DEEPSPACE PROGRAM!

```
500 DIM D(4)
510 DIM E(4)
520 DIM C(2)
600 DIM N(5)
605 PRINT
610 PRINT 'WANT AN ACTION CHART? ';
615 INPUT M$
618 PRINT
620 IF M=217 GOTO 660
630 GOTO 770
660 PRINT
670 PRINT ' 1      FIRE PHASERS'
680 PRINT ' 2      FIRE ANTI-MATTER MISSILE'
690 PRINT ' 3      FIRE HYPERSPACE LANCE'
700 PRINT ' 4      FIRE PHOTON TORPEDO'
710 PRINT ' 5      HYPERON NEUTRO FIELD'
720 PRINT ' 6      SELF-DESTRUCT'
730 PRINT ' 7      CHANGE VELOCITY'
740 PRINT ' 8      DISENGAGE'
750 PRINT ' 9      PROCEED'
760 PRINT
770 PRINT 'WHICH SYSTEM (1-3)';
815 INPUT N
820 IF N=1 GOTO 2380
830 IF N=2 GOTO 2430
840 GOTO 2480
850 D0=0
860 D(1)=0
870 N(1)=0
880 N(2)=0
890 N(3)=0
900 N(4)=0
910 D=0
920 PRINT 'WHICH SPACECRAFT (1-3)';
925 INPUT N
930 IF N=1 GOTO 1790
940 IF N=2 GOTO 1830
950 IF N=3 GOTO 1870
960 GOTO 920
970 C=C(0)
980 PRINT 'YOU HAVE';C;' UNITS OF STORAGE.
990 PRINT 'WEAPON';
992 INPUT W
995 PRINT 'AMOUNT';
998 INPUT N
1000 IF W=1 GOTO 1910
1010 IF W=2 GOTO 2010
1020 IF W=3 GOTO 2100
1030 IF W=4 GOTO 2190
1040 IF W=5 GOTO 2280
1050 GOTO 980
1060 IF N*C(1)>C GOTO 2530
1070 C=C-N*C(1)
1080 IF W=1 THEN 1990
1090 IF W=2 THEN 2050
1100 IF W=3 THEN 2170
1110 IF W=4 THEN 2260
1120 GOTO 2360
1130 IF C>1 GOTO 980
1140 REM
1150 S1=S0*RND(0)
1160 R=(3*RND(0) +5)*100
1170 PRINT
1180 PRINT 'RANGE TO TARGET:';R
1190 PRINT 'RELATIVE VELOCITY:';S1
1200 PRINT 'ACTION';
1205 INPUT M
1210 IF M=1 GOTO 1940
1220 IF M=2 GOTO 2030
1230 IF M=3 GOTO 2120
1240 IF M=4 GOTO 2210
1250 IF M=5 GOTO 2310
1260 IF M=6 GOTO 1660
1270 IF M=7 GOTO 1390
1280 IF M=8 GOTO 2760
1290 IF R<500 GOTO 1500
1310 R=R-(S1*8.3)*1.25
1340 IF R>1500 GOTO 2590
1350 IF R>0 THEN 1370
1360 R=-R
1370 PRINT
1380 GOTO 1180
1390 PRINT 'CHANGE TO BE EFFECTED:';
1395 INPUT S2
1400 IF (S1+S2)>S0 THEN 2550
1410 S1=S1+S2
1420 GOTO 1180
1430 F0=P1*(Z/R)^1.5
1450 D0=(2*F0+3*F0*RND(0))/5
1460 D=D+D0
1470 PRINT 'SCANNERS REPORT ENEMY DAMAGE NOW:';D
1480 IF D>99 THEN 2720
1490 GOTO 1520
1500 D0=0
1520 K=E(1)+E(2)*RND(0)
1540 E=E(3)+E(4)*RND(0)+5/P0*RND(0)
1560 F3=E*(K/R)^1.85
1570 D(2)=(3*F3+3*F3*RND(0))/5.5
1580 D(1)=D(1)+D(2)
1590 IF (Z=D0)/(R=500)>2.2 GOTO 1620
1600 D(3)=D0=2/(R^2*P0)
1610 D(1)=D(1)+D(3)
1620 PRINT 'YOUR VESSEL DAMAGE:';D(1)
1630 IF D(1)>99 GOTO 2740
1640 IF D>99 GOTO 2760
1650 GOTO 1310
1660 PRINT 'SELF DESTRUCT FAILSAFE ACTIVATED!!'
1670 PRINT 'INPUT 1 TO RELEASE FAILSAFE.';
1675 INPUT U
1680 IF U=1 THEN 1700
1690 GOTO 1290
1700 PRINT 'SELF-DESTRUCT ACCOMPLISHED. '
1710 IF R>60 THEN 1740
1720 PRINT 'ENEMY VESSEL ALSO DESTROYED.'
1730 GOTO 2760
1740 D(4)=3200/R
1750 D=D+D(4)
1760 IF D>99 THEN 1720
1770 PRINT 'ENEMY VESSEL SURVIVES WITH';D;' DAMAGE.'
1780 GOTO 2760
1790 S0=10
1800 C(0)=16
1810 P0=1
1820 GOTO 970
1830 S0=4
1840 C(0)=24
1850 P0=2
1860 GOTO 970
1870 S0=2
1880 C(0)=30
1890 P0=5
1900 GOTO 970
1910 C(1)=.12
1920 N=100
1930 GOTO 1060
1940 P1=4
1950 IF N(1)=0 THEN 2610
1960 N(1)=N(1)-1
1970 Z=200
1980 GOTO 1430
1990 N(1)=N(1)+N
2000 GOTO 1130
2010 C(1)=4
2020 GOTO 1060
2030 P1=20
2040 IF N(2)=0 GOTO 2640
2050 N(2)=N(2)-1
2060 Z=500
2070 GOTO 1430
2080 N(2)=N(2)+N
2090 GOTO 1130
2100 C(1)=4
2110 GOTO 1060
2120 P1=16
2130 IF N(3)=0 GOTO 2660
2140 N(3)=N(3)-1
2150 Z=550
2160 GOTO 1430
2170 N(3)=N(3)+N
2180 GOTO 1130
2190 C(1)=2
2200 GOTO 1060
2210 P1=10
2220 IF N(4)=0 GOTO 2680
2230 N(4)=N(4)-1
2240 Z=400
2250 GOTO 1430
2260 N(4)=N(4)+N
2270 GOTO 1130
2280 C(1)=.1999998
2290 N=100
2300 GOTO 1060
2310 P1=6
2320 IF N(5)=0 THEN 2700
2330 N(5)=N(5)-1
2340 Z=250
2350 GOTO 1430
2360 N(5)=N(5)+N
2370 GOTO 1130
2380 E(1)=150
2390 E(2)=500
2400 E(3)=3
2410 E(4)=4
2420 GOTO 850
2430 E(1)=200
2440 E(2)=350
2450 E(3)=4
2460 E(4)=3
2470 GOTO 850
2480 E(1)=150
2490 E(2)=400
2500 E(3)=5
2510 E(4)=2
2520 GOTO 850
2530 PRINT 'NOT ENOUGH SPACE. RESELECT.'
2540 GOTO 980
2550 PRINT 'CHANGE BEYOND MAXIMUM POSSIBLE. '
2560 PRINT 'INCREASING TO MAXIMUM.'
2570 S1=S0
2580 GOTO 1310
2590 PRINT 'OUT OF RANGE. DISENGAGED.'
2600 GOTO 2760
2610 PRINT 'PHASER BANKS DRAINED.'
2620 PRINT 'SELECT ANOTHER COURSE OF ACTION.'
2630 GOTO 1200
2640 PRINT 'OUT. '
2650 GOTO 2620
2660 PRINT 'OUT. '
2670 GOTO 2620
2680 PRINT 'OUT. '
2690 GOTO 2620
2700 PRINT 'OUT. '
2710 GOTO 2620
2720 PRINT 'ENEMY VESSEL DESTROYED.'
2730 GOTO 1520
2740 PRINT 'YOUR VESSEL DESTROYED.'
2760 PRINT
2770 PRINT 'WANT ANOTHER BATTLE? ';
2780 INPUT M$
2790 IF M=217 GOTO 605
2800 END
```

THE PROGRAM WAS MODIFIED FOR SCELBAL BY REPLACING SOME OF THE REGULAR VARIABLE NAMES WITH ARRAY ELEMENTS. A PRACTICE SUGGESTED IN SCELBAL UPDATE ISSUE NR. 1.

DEEPSPACE IS ANOTHER VERSION OF A SPACE BATTLE. YOU BECOME THE COMMANDER OF EITHER A SCOUT SHIP, CRUISER, OR BATTLESHIP. YOU THEN PICK THE WEAPONS AND THE PLANETARY SYSTEM YOU DESIRE TO PATROL. THEN IT'S TIME TO DO BATTLE.

THE CLOSER YOU GET TO THE ENEMY, THE BETTER YOUR CHANCE OF DESTROYING HIM. UNFORTUNATELY, HIS CHANCES OF DESTROYING YOU ALSO IMPROVE. IF YOU GET TOO CLOSE, YOU CAN DAMAGE YOURSELF. WHEN A VESSEL'S DAMAGE RATING EXCEEDS 99 IT IS DESTROYED!

THE REMARKS IN THE PROGRAM HAVE BEEN REMOVED AND ARE PRESENTED HERE TO SAVE PROGRAM STORAGE SPACE. (PROGRAM WILL THUS FIT IN A 12K SYSTEM RUNNING SCELBAL WITH DIMENSION CAPABILITY INSTALLED.)

"THIS IS DEEPSPACE, A TACTICAL SIMULATION OF SHIP-TO-SHIP COMBAT IN DEEP SPACE. YOU ARE ASSIGNED TO PATROL A SECTION OF YOUR STAR EMPIRE'S BORDERS AGAINST HOSTILE ALIENS. ALL YOUR ENCOUNTERS WILL BE AGAINST HOSTILE VESSELS. YOU WILL FIRST BE REQUIRED TO SELECT A VESSEL FROM ONE OF THREE TYPES, EACH WITH ITS OWN CHARACTERISTICS.

TYPE	SPEED	CARGO	SHIELDS
1: SCOUT	10X	16	1U
2: CRUISER	4X	24	2U
3: BATTLESHIP	10X	30	5U

SPEED IS GIVEN RELATIVE TO OTHER SHIPS. CARGO SPACE IS IN UNITS OF SPACE ABOARD SHIP WHICH CAN BE FILLED WITH WEAPONS. PROTECTION IS THE RELATIVE STRENGTH OF THE SHIP'S ARMOR AND SHIELD FORCE. ONCE A SHIP HAS BEEN SELECTED YOU WILL BE ALLOWED TO ARM IT WITH WEAPONRY FROM THE FOLLOWING LIST:

TYPE	NAME
1	PHASER BANKS
2	ANTI-MATTER MISSILE
3	HYPERSPACE LANCE
4	PHOTON TORPEDO
5	HYPERON NEUTRALIZATION

WEAPON TYPES REQUIRE THE FOLLOWING AMOUNTS OF STORAGE SPACE AND HAVE THE FOLLOWING RELATIVE STRENGTHS:

TYPE	CARGO SPACE	STRENGTH
1	12	4
2	4	20
3	4	16
4	2	10
5	20	6

WEAPON TYPES 1 AND 5 MAY BE FIRED 100 TIMES. ALL OTHER TYPES MAY BE FIRED ONCE FOR EACH ON BOARD.

RANGE IS GIVEN IN THOUSANDS OF KILOMETERS.

CAUTION! FIRING HIGH YIELD WEAPONS AT CLOSE RANGE CAN BE FATAL TO YOUR SHIP!

"THERE ARE THREE SYSTEMS ONE MAY PATROL, EACH HAVING DIFFERENT CHARACTERISTICS.

SYSTEM NUMBER	NAME
1	ORION
2	DENEB
3	ARCTURUS

THE FIRST TIME YOU PLAY A GAME ANSWER THE FIRST QUESTION WITH A "Y" FOR YES. YOU WILL LEARN SOME VITAL INFORMATION!"

DEEPSPACE PROGRAM

ORIGINAL AUTHOR: UNKNOWN

THE PROGRAM PRESENTED HERE IS AN ADAPTATION FOR SCELBAL OF A PROGRAM THAT WAS MODIFIED BY BILL COTTER OF PITTSFIELD, MASS., AND IS REPRINTED HERE WITH THE PERMISSION OF THE COPYRIGHT OWNER - FOR WHICH WE EXTEND OUR THANKS ON BEHALF OF OUR READERS.


```
RUN

WANT AN ACTION CHART? Y

 1      FIRE PHASERS
 2      FIRE ANTI-MATTER MISSILE
 3      FIRE HYPERSPACE LANCE
 4      FIRE PHOTON TORPEDO
 5      HYPERON NEUTRO FIELD
 6      SELF-DESTRUCT
 7      CHANGE VELOCITY
 8      DISENGAGE
 9      PROCEED

WHICH SYSTEM (1-3)?2
WHICH SPACECRAFT (1-3)?3
YOU HAVE 30.0 UNITS OF STORAGE.
WEAPON?1
AMOUNT?1
YOU HAVE 18.0 UNITS OF STORAGE.
WEAPON?2
AMOUNT?2
YOU HAVE 10.0 UNITS OF STORAGE.
WEAPON?3
AMOUNT?1
YOU HAVE 6.0 UNITS OF STORAGE.
WEAPON?5
AMOUNT?1
NOT ENOUGH SPACE. RESELECT.
YOU HAVE 6.0 UNITS OF STORAGE.
WEAPON?4
AMOUNT?3

RANGE TO TARGET: 537.2858
RELATIVE VELOCITY: 8.5989952
ACTION?7
CHANGE TO BE EFFECTED:?+2
CHANGE BEYOND MAXIMUM POSSIBLE.
INCREASING TO MAXIMUM.

RANGE TO TARGET: 520.6848
RELATIVE VELOCITY: 2.0
ACTION?4
SCANNERS REPORT ENEMY DAMAGE NOW: 5.37
YOUR VESSEL DAMAGE: 3.713214

RANGE TO TARGET: 504.0848
RELATIVE VELOCITY: 2.0
ACTION?9

RANGE TO TARGET: 487.4848
RELATIVE VELOCITY: 2.0
ACTION?1
SCANNERS REPORT ENEMY DAMAGE NOW: 6.609635
YOUR VESSEL DAMAGE: 5.935885

          ■
          ■
          ■
          ■
          ■

RANGE TO TARGET: 285.2841
RELATIVE VELOCITY: 2.0
ACTION?3
OUT.
SELECT ANOTHER COURSE OF ACTION.
ACTION?2
SCANNERS REPORT ENEMY DAMAGE NOW: 96.17733
YOUR VESSEL DAMAGE: 83.53137

RANGE TO TARGET: 168.6841
RELATIVE VELOCITY: 2.0
ACTION?1
SCANNERS REPORT ENEMY DAMAGE NOW: 99.95826
ENEMY VESSEL DESTROYED.
YOUR VESSEL DAMAGE: 94.71087

WANT ANOTHER BATTLE? N
READY
```

LETTERS

Mr. S. J. Toy is one of those hearty souls who utilizes a Baudot encoded teleprinter with his computer system. These machines are generally considerably less expensive than the sought after ASCII encoded devices. We don't know how many other SCELBAL users may be using the same type of machine but we thought Mr. Toy's comments - relating to the use of such a machine - and other matters, would be of interest to all. (Users with Baudot machines might be interested in communicating directly with Mr. Toy on mutual grounds.)

When Mr. Toy originally received his copy of SCELBAL he was apparently a little crestfallen when he discovered the limitations on the use of CPU registers specified in the book. The recomendation that only CPU registers A and B be used for I/O routines met with the following comments. . . . . "Since the accumulator is loaded with the data to be inputted or outputted this really leaves only register B. I normally need H and L for the Baudot-ASCII conversion. After casting about for several days trying to decide what hardware modifications had to be made, I finally decided to look into the possibility of program modification. To my surprise I found that the ECHO routine leaves H and L free, so there is no problem on output. The input situation, however, was not as easy. After considerable study I concluded tentatively that D and E were free. So I went ahead and developed some I/O routines on this basis. The results so far indicate apparent success. (But wait - read on some more! Ed.) I have now tried everything in the chapter on operating SCELBAL up to and including page 14-3 with the correct results, with one exception. In addition, simple problems in addition, subtraction, multiplication, and division yield the correct answers.

The one exception mentioned above was that the TAB function did not work properly. Instead of all spaces between "HELLOS", the first character was a space as expected but the rest were something else. A study of this problem revealed that at least for TAB the contents of the accumulator must also be saved on output. To make a long story short, the simplest solution was to change the contents of 015 010 from 003 to 001. This reloads the accumulator with a "space" each time a space is supposed to be sent." Don't change your system yet - read on! Ed.)

A few days later another letter was received from Mr. Toy and the discussion started above was continued. . . . . "On the matter of the TAB function, my original quick fix turned out to be for the comma controlled routine only, PCOM1. It is also necessary to similarly modify TABLOP for the numerically controlled spacing, and the BACKSP for backspacing. The latter would require a patch so I gave up on this tack, modified my output routine to save and restore A.....Incidently, PCOM1 and TABLOP are identical except for addresses so one of them can be eliminated if memory space is needed."

Mr. Toy then went on to a new topic. . . . . "I have tried all the example programs in the SCELBAL manual except for the last one. They all appear to operate properly except the two programs involving the CHR function on pages 14-24 and 14-29. In the table program the last character of the octal number comes out as a letter. In the line printing program only the first character in the line comes out correctly. Unless my I/O routines are associated with these problems, which seems unlikely, it would appear that registers B, D and E are free on input, and B, H, and L are free on output. In addition, on output, A must be saved and restored for the TAB function."

Mr. Toy must really be working his system out because in a few more days he added the following comments . . . . . "After several hours of hard labor I finally found out why the CHR program on page 14-29 is so complicated that it requires about half a second for each character to be processed before the program looks for the next character! This delay seems to be unusual, so readers may well be advised of this fact in connection with this particular program, especially if they are using an 8008." (True - the delay is rather disconcerting on an 8008 based system. 8080 users, however, will find the delay barely perceptible. Ed.)

"I still have not determined why the octal numbers in the CHR table program on page 14-24 do not come out correctly. However, I am now reasonably satisfied that my I/O routines work properly on all functions, so I will not spend much more time on this. For your information I am enclosing a printout of my results.

Please note that I have substituted a dash for the READY message. This involved changing only two bytes in SCELBAL; 001 352 is reloaded with 003 and 001 353 is reloaded with 255. The result is a single line space for "READY" instead of three. This uses up much less paper, especially when operating in the "calculator" mode."

Not one to give up. Mr. Toy soon followed up with:. . . . . "I finally discovered why the program on the Table of ASCII characters would not work. An "8 X" in statement 130 was missing. A printout of the correction and a RUN enclosed. You may also be interested in the substitution of characters to use the model 15 TTY."

THE EDITOR REPLIES

Communications of the type Mr. Toy has submitted are exactly why we established the support publication SCELBAL UPDATE. It is through such communications that SCELBAL itself can be improved, or tailored to suit the requirements of individual users or groups of users. Mr. Toy's letters are the first of what we hope become a flood of similar such communiques aimed at disseminating information about SCELBAL amongst its users.

Now, to answer or explain a few of the questions raised by Mr. Toy.

Mr. Toy has apparently made some very useful discoveries in regards to the availability of certain CPU registers during I/O operations. His observations should be of considerable interest to users with special I/O devices who find they need more CPU registers available. The stipulation made in the publication regarding limiting the use of CPU registers to just A and B was given on the basis of design guidelines that the program authors established. In other words, the program authors, during the development stages, reserved those two registers for use during I/O operations, so that they would have the freedom of using all other CPU registers if desired. They did not, during the development process, keep track of whether every other possible register was thus actually in use during I/O operations. Mr. Toy's observations are as interesting to the authors as they may be to others and may be taken for what they are worth. (Which is a lot if your running a Baudot machine!)

Mr. Toy's observation regarding the saving of the accumulator's original status during an output operation that utilizes a TAB is correct. The users output routine should exit with the original character in the accumulator still present.

Our thanks to Mr. Toy, (and our apologies to all readers) for discovering the clerical error on line number 130 of the example SCELBAL program on page 24 of chapter 14. The line should read:

```
130 Q3=INT(N - 64*Q1 - 8*Q2)
```

The suggestion regarding the use of a hyphen to shorten the READY seems like a good one for those that want to implement it.

Users who anticipate using a Baudot coded device might be interested in contacting Mr. Toy directly to discuss I/O routines etc.. His address is:

Mr. S. Joseph Toy
Route 3, Box 73
Chico, CA 95926

SCELBAL UPDATE

ISSUE 03 - 11/76

STRING CAPABILITIES FOR SCELBAL

One of the most asked for additions to SCELBAL is capability to manipulate character strings in the manner permitted on most large computer systems when running extended BASIC. Soon, a supplement will be available for SCELBAL that will give it string manipulating features capable of performing the following types of operations:

1. Up to 64 strings and/or string arrays, each string up to 80 characters in length.

2. Substring capabilities as follows:
   A. The right part of a string.
   B. The middle part of a string.
   C. The left part using B.
   D. A string array can be substringed in the same expression.

3. Two additional numeric functions:
   A. LEN - will return the length of a string.
   B. ASC - will return the decimal value of the first character in a string.

4. One additional string function - CHR$ - (will replace CHR).

5. String arrays do not require dimensions.

6. Concatenation of string expressions.

7. Input and output of strings.

8. Comparison of string expressions.

The following discussion will amplify the capabilities of the string handling routines that will be made available in the new supplement.

STRING VARIABLES

A string variable may be any letter followed by a dollar sign ($). For example, A$ would be a legal string variable. A string variable may be subscripted in the normal fashion: B$(3) would yield the third element of the string array B$. The difference between numeric arrays and string arrays is that unsubscripted string variables are treated the same as one with a subscript of one, so A$ and A$(1) reference the same string. String arrays do not require (or allow) a dimension to be specified in a DIMension statement. This feature allows the full string capability to be implemented in a system without the array option installed.

SUBSTRINGS

It is often desirable to access certain characters within a string by specifying the starting and stopping positions in that string. This capability is know as accessing a substring. To access J characters starting the N'th character in a string A$ the format would be: A$(:N;J), where N and J could be any expressions. For example, if A$ contained "ABCDE" then A$(:1;4) would yield "ABCD." A string array could also be substringed: B$(4:2;3) which would yield the second through fourth characters of the fourth element of B$. If the semicolon and expression following it were omitted, the result would be all the characters to the right of (including) the N'th character. Thus, A$(:3) will result in "CDE." Subscripted strings are handled in a similar fashion: B$(5:3) would result in all characters to the right of the second character of the fifth element of B$ being specified.

CHR$ FUNCTION

The CHR$ function is used to generate a single character string by converting the decimal value of its argument to ASCII. For example, CHR$ (193) would result in the string "A." This string function replaces the old CHR function.

STRING LITERALS

The string literal is just like the old text in a PRINT statement: either single or double quotes enclosing the characters that form the string. For example, "THIS STRING" or 'ABCD $ 44.'

STRING CONCATENATIONS

Strings can be concatenated using the + operator. Concatenation is the joining together of two or more strings. For example, "AB"+'CD' forms "ABCD," and A$+B$ (8:4) + 'Q' forms a string of A$ joined with the fourth character through the end of the eighth element of B$ and the literal 'Q.'

ADDITIONAL FUNCTIONS

Two new numeric functions add additional power to the language:

LEN(A$): This function returns the length of a string or string array as a decimal number. For example, if A$ has the value as in the above examples, LEN(A$) returns 5.

ASC(A$): This function returns the decimal value of the first character of the string or string array specified in ASCII. For example, ASC(A$) would return 193, because A$(:1;1) has a value of "A."

These functions can be used anywhere in a numeric expression where a regular function is legal.

STRING EXPRESSION

A string expression is any string variable, string array, string literal, use of CHR$ function, or any concatenation of these. For example: A$+'THIS' or CHR$(N)+'T'+W$(D+E:6;J). String expressions are legal in PRINT statements (where they replace the old text strings) and on the right of an = in a LET.

STRING LET

The string LET statement is similar to the regular LET, and may take two forms:

string = string expression
or
string array = string expression

For example, A$='EXAMPLE' or C$(N)=A$+D$(:3) or 30 LET A$=A$+C$.

STRING OUTPUT

A string may be output in a PRINT statement subject to the normal rules for spacing and tabbing along with numeric data. For example: PRINT 'AB'+'CD' would print ABCD, or PRINT A$;2*2,B$ would print A$, then immediately print 4, then tab to the next column and print B$.

STRING INPUT

Strings or string arrays can be input using the INPUT statement in the normal fashion. For example: INPUTA$,B$(3),N would print a ? and ask for the string value of A$, then when the CR was entered, would print another ? and ask for B$(3), and then would finally input N in the normal fashion. Note that this feature replaces the old automatic conversion of ASCII input using the $.

STRING COMPARISON

String expressions can be compared using the normal comparison operators such as =,<,>,>=,<=, or <>. If the condition is satisfied, a value of 1.0 is returned as a numeric result, and 0 is returned otherwise. The comparison goes character by character until unequal characters are found, or until all of the characters in the shortest string have been tested. In the former case, the test comparison is made between the two unequal characters, and in the latter, the length is used as the deciding factor.

TRANSLATION FROM OTHER BASIC'S

Programs written for other BASIC's can probably be translated to SCELBAL with strings as follows:

RIGHT$(A$,N)	becomes A$(:N)
LEFT$(A$,N)	becomes A$(:1;N)
MID$(A$,N,J)	becomes A$(:N;J)

The reason this format was chosen over the normal "function" format is that the SCELBAL notation is more concise and requires less memory to implement.

ADDITIONAL FUNCTIONS

Three new functions add additional power to the language:

LEN(A$): This function returns the length of a string or string array as a decimal number. For example, if A$ has the value as in the above example, LEN(A$) returns 5.

ASC(A$): This function returns the decimal value of the first character of the string or string array specified in ASCII. For example, ASC(A$) would return 193, because A$(:1;1) has a value of "A."

VAL(A$): This function converts the characters in the string from an ASCII representation of a decimal number to its numeric value. For example, VAL('2') returns 2.

These functions should be used only at the beginning of an expression. The arguments of these functions should be either a plain string, such as A$, or a string array subscripted by a regular variable, i.e., B$(J). So LEN(A$) and ASC(C$(N6)) would be legal, but LEN(C$(6)) and ASC(A$(:2)) would not be legal. (The reason for this restriction is that on an 8008 system using a function like LEN(A$(6)) pushes the PC stack down more than 8 levels. An 8080 system would not have this problem.)

MEMORY USAGE

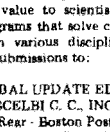
The string package is designed to supplement SCELBAL configured to run in systems with 12 K or more of memory. The string package uses one page for working pointers and registers, one page for a string variables symbol table, and as many pages as the user assigns for storage of the actual strings. The string operating routines require about 1.5 K of memory.

The supplementary string handling addition to SCELBAL is in the checkout and documentation stages. The supplement is scheduled to be placed on the market in a few months at a moderate price. It is anticipated that paper tapes of the object code of the string supplement will also be made available for purchase.

PREMIUMS FOR YOUR PROGRAM

If you have developed your own original program to perform tasks that may be of interest to other SCELBAL users, chances are you are in a position to make some money. Original programs that we accept for publication in SCELBAL UPDATE earn the author an honorarium check and a handsome certificate. We are particularly interested in programs that may be of value to scientists, engineers, and businessmen. Programs that solve commonly encountered formulas in various disciplines for example. Please send your submissions to:

SCELBAL UPDATE EDITOR
SCELBI C. C., INC.
1322 Rear - Boston Post Road
Milford, CT 06460



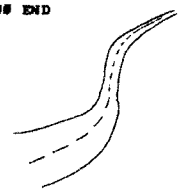
PROGRAM CALCULATES WEEKLY WAGES ALONG WITH FWT AND FICA DEDUCTIONS

SCELBAL users that operate a small business might find the following program quite a time saver. Type in the number of regular and overtime hours worked, number of personal allowances claimed, and the hourly pay rate. The program responds with gross pay, deductions, and net pay. The calculations are based on current government standards.

```
100 PRINT '1976 WEEKLY PAYROLL PROGRAM'
105 PRINT
110 PRINT
115 PRINT
120 PRINT 'REGULAR HOURS WORKED:';
125 INPUT RH
130 PRINT 'OVERTIME HOURS WORKED:';
135 INPUT OH
140 PRINT 'WITH HOLDING ALLOWANCES:';
145 INPUT WH
150 PRINT 'SINGLE (0) OR MARRIED (1)';
155 INPUT SM
160 IF SM = 0 GOTO 170
165 IF SM <> 1 GOTO 150
170 PRINT 'HOURLY WAGE:';
175 INPUT HW
180 PRINT
200 PRINT 'REGULAR PAY =';RH*HW
210 PRINT 'OVERTIME PAY =';OH*1.5*HW
215 GP = RH*HW+OH*1.5*HW
220 PRINT  'GROSS PAY =';GP
230 IF SM <> 0 GOTO 245
235 GOSUB 380
240 GOTO 250
245 GOSUB 480
250 PRINT 'FWT DEDUCTION =';TX
260 SS = 0.0585*GP
270 PRINT 'FICA WITH HOLDING =';SS
280 NP = GP-TX-SS
290 PRINT 'NET PAY =';NP
295 GOTO 110
300 TT = GP - (WH*14.4)
305 IF TT <= 0.0 GOTO 315
310 IF TT => 25 GOTO 320
315 TX = 0
318 RETURN
320 IF TT => 67 GOTO 330
325 TX = (0.16*(TT-25))
328 RETURN
330 IF TT => 115 GOTO 340
335 TX = 6.72 + (0.20*(TT-67))
338 RETURN
340 IF TT => 183 GOTO 350
345 TX = 16.32 + (0.23*(TT-115))
348 RETURN
350 IF TT => 240 GOTO 360
355 TX = 31.96 + (0.21*(TT-183))
358 RETURN
360 IF TT => 279 GOTO 370
365 TX = 43.93 + (0.26*(TT-240))
368 RETURN
370 IF TT => 346 GOTO 380
375 TX = 54.87 + (0.30*(TT-279))
378 RETURN
380 TX = 74.17 + (0.36*(TT-346))
385 RETURN
480 TT = GP - (WH*14.4)
485 IF TT <= 0.0 GOTO 415
410 IF TT => 48 GOTO 420
415 TX = 0
418 RETURN
420 IF TT => 96 GOTO 430
425 TX = (0.17*(TT-48))
428 RETURN
430 IF TT => 173 GOTO 440
435 TX = 8.16 + (0.20*(TT-96))
438 RETURN
440 IF TT => 264 GOTO 450
445 TX = 23.56 + (0.17*(TT-173))
448 RETURN
450 IF TT => 346 GOTO 460
455 TX = 39.03 + (0.25*(TT-264))
458 RETURN
460 IF TT => 433 GOTO 470
465 TX = 59.53 + (0.28*(TT-346))
468 RETURN
470 IF TT => 500 GOTO 480
475 TX = 83.89 + (0.32*(TT-433))
478 RETURN
480 TX = 105.33 + (0.36*(TT-500))
485 RETURN
```

ROADRACE

```
100 DIM C(2)
220 PRINT 'WHICH CAR (1-4)';
230 INPUT C(1)
240 C1=INT(C(1))
250 IF C(1)>4 GOTO 280
260 IF C(1)<1 GOTO 280
270 GOTO 300
280 PRINT 'INVALID CAR TYPE. NEW CAR';
290 GOTO 230
300 PRINT
350 PRINT 'WHICH COURSE (1-5)';
360 INPUT C(2)
370 C(2)=INT(C(2))
380 IF C(2)<1 GOTO 410
390 IF C(2)>5 GOTO 410
400 GOTO 500
410 PRINT 'INVALID COURSE NUMBER. NEW CHOICE';
420 GOTO 360
500 GOSUB 1600
510 B=B/10
530 A1=.5
540 M1=0
550 C(1)=C(1)/2
560 V=0
570 PRINT
580 R1=0
590 T=0
600 D=0
610 G1=0
620 PRINT 'PRESENT VELOCITY ='JV;' MO. OF GALLONS ='JA1
630 PRINT 'NO. OF MILES ='JM1;' TIME PASSED ='JT;' SECONDS.'
640 IF M1>=5 GOTO 1460
650 PRINT 'WHAT IS YOUR NEW RATE OF GAS';
660 INPUT G
670 IF G<-10 GOTO 700
680 IF G>10 GOTO 700
690 GOTO 720
700 PRINT 'NOT VALID. NEW RATE';
710 GOTO 660
720 IF G<9 THEN 780
730 Z=Z+1
740 IF Z>4 THEN 760
750 GOTO 790
760 PRINT 'YOUR ENGINE BLEW. YOU GOT HIT BY A PISTON.'
770 GOTO 1270
780 Z=0
790 V=INT(B*G-M*V+V)
800 T=T+10
810 PRINT
820 PRINT 'ROAD CONDITIONS:';
830 IF V>0 GOTO 850
840 V=0
850 M1=M1+V/460
860 IF G<0 GOTO 890
870 A1=A1-(G*S)/5000
880 IF A1<0 GOTO 1380
890 IF R1=1 GOTO 1050
900 IF Q1=1 GOTO 950
910 Q=INT((C(2)+1)*RND(0))
920 R=INT((3.75-C(2))*RND(0))
930 IF R>0 GOTO 1290
940 IF Q>0 GOTO 1340
950 PRINT ' CLEAR AND STRAIGHT.'
960 PRINT
970 GOTO 620
980 H=INT(15+35*RND(0))
990 H=H+5*C(1)
1000 IF V>H GOTO 1500
1010 PRINT ' THROUGH CURVE.'
1020 PRINT
1030 Q1=0
1040 GOTO 620
1050 E=E-(V-D)*3
1060 IF E<0 GOTO 1100
1070 PRINT ' VEHICLE'JE' FEE T AHEAD.'
1080 PRINT
1090 GOTO 620
1100 IF V-D<5 THEN 1180
1110 PRINT ' VEHICLE PASSED BY';
1120 D=V-D
1130 PRINT DJ' MPH.'
1150 PRINT
1160 R1=0
1170 GOTO 620
1180 PRINT ' VEHICLE BEING PASSED.'
1190 D=INT(25+40*RND(0))
1200 PRINT 'GRAYHOUND BUS IN OTHER LANE DOING'JD' MPH.'
1240 D=V+D
1250 PRINT 'CRASH VELOCITY ='JDJ' MPH.'
1270 PRINT 'WHERE IS THE FUNERAL BEING HELD?'
1280 GOTO 1560
1290 PRINT ' VEHICLE AHEAD 500 FEET.'
1300 PRINT
1310 D=INT(25+35*RND(0))
1320 R1=1
1330 GOTO 620
1340 PRINT ' WARNING! CURVE AHEAD!'
1350 Q1=1
1360 PRINT
1370 GOTO 620
1380 PRINT 'EXCELLENT! BUT WAIT .... YOU RAN OUT OF GAS.'
1410 GOTO 1550
1420 PRINT 'BUT SOME HOW YOU MADE IT!'
1430 PRINT
1440 R1=0
1450 GOTO 620
1460 PRINT
1470 PRINT
1480 PRINT 'YOU MADE IT (LUCK) !!!!'
1490 GOTO 1560
1500 PRINT ' ARE TERRIBLE.'
1510 H=H-5*C(1)
1520 PRINT HJ' WAS THE SPEED THROUGH THE CURVE.'
1530 PRINT VJ' W AS YOUR SPEED. BY THE WAY ....'
1540 GOTO 1270
1550 PRINT 'YOU LEAD FOOTED IDIOT!!'
1560 PRINT 'YOU WANT TO TRY IT AGAIN? ';
1570 INPUT I$
1572 PRINT
1573 PRINT
1575 IF I=217 GOTO 220
1580 END
1600 IF C(1)<>1 GOTO 1625
1605 B=45
1610 M=.53
1615 S=10
1620 RETURN
1625 IF C(1)<>2 GOTO 1650
1630 B=60
1635 M=.5
1640 S=13
1645 RETURN
1650 IF C(1)<>3 GOTO 1675
1655 B=70
1660 M=.41
1665 S=15
1670 RETURN
1675 B=80
1685 M=.39
1690 S=18
1695 RETURN
1700 END
```



THIS PROGRAM WAS ADAPTED FOR 12 K SCELBAL BY MAKING SEVERAL MINOR CHANGES AND ELIMINATING A FEW REMARKS STATEMENTS. THE ESSENCE OF THOSE STATEMENTS IS PRESENTED HERE.

"ROADRACE" PUTS THE PLAYER IN THE DRIVER'S SEAT OF A CAR OF CHOICE SELECTED FROM THE FOLLOWING POSSIBILITIES.

1 - VW
2 - 283 NOVA
3 - Z-28
4 - FERRARI

THE SELECTED CAR IS TO BE DRIVEN ALONG A HIGHWAY CHOSEN BY THE PLAYER WHICH IS RANKED IN DIFFICULTY FROM 1 (EASY) TO 5 (QUITE DIFFICULT)! THE DEGREE OF DIFFICULTY RELATES TO THE NUMBER OF CURVES AND OTHER HAZARDS THAT MAY BE ENCOUNTERED ON THE DRIVE.

THE PLAYER MUST SUCCESSFULLY NEGOTIATE FIVE MILES OF TREACHOROUS ROAD WHILE BEING LIMITED TO 1/2 A GALLON OF GAS. THE PLAYER HAS CONTROL OF AN "ACCELORATOR" TO SPEED UP OR SLOW DOWN THE PROGRESS OF THE CAR. NATURALLY, A FERRARI CAN GO FASTER (AND STICKS TO THE ROAD BETTER) THAN A VW, JUST AS NATURALLY, IT GUZZLES MORE GAS!

ROAD CONDITIONS ARE CONSTANTLY CHANGING AS THE RACE TAKES PLACE. RACE? YES, THE OBJECT IS NOT ONLY TO COMPLETE THE COURSE (WHICH CAN BE CHALLENGING IN ITSELF), BUT TO COMPLETE IT IN THE LEAST AMOUNT OF TIME WITH THE MAXIMUM AMOUNT OF FUEL! THUS, THERE IS ALWAYS ROOM FOR THE SUCCESSFUL DRIVER TO IMPROVE.

THE HAZARDS ALONG THE DRIVE APPEAR RANDOMLY SO NO TWO GAMES WILL BE ALIKE. FRANKLY, THIS GAME APPEARS TO REMAIN FUN SOMEWHAT LONGER THAN A LOT OF THE COMPUTER GAMES ONE SEEMS TO ENCOUNTER THESE DAYS.

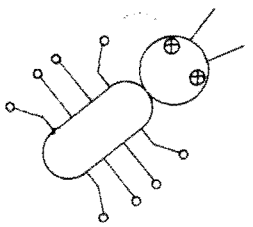
HAVE FUN!

ROADRACE PROGRAM

ORIGINAL AUTHOR: UNKNOWN

THE PROGRAM PRESENTED HERE IS AN ADAPTATION FOR SCELBAL OF A PROGRAM THAT WAS MODIFIED BY BILL COTTER OF PITTSFIELD, MASS., AND IS REPRINTED HERE WITH THE PERMISSION OF THE COPYRIGHT OWNER - FOR WHICH WE EXTEND OUR THANKS ON BEHALF OF OUR READERS.


```
WHICH CAR (1-4)?4

WHICH COURSE (1-5)?1

PRESENT VELOCITY = 0  NO. OF GALLONS = 0.5000000
NO. OF MILES = 0  TIME PASSED = 0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?6

ROAD CONDITIONS: CLEAR AND STRAIGHT.

PRESENT VELOCITY = 48.0  NO. OF GALLONS = 0.4783999
NO. OF MILES = 0.1043478  TIME PASSED = 10.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?6

ROAD CONDITIONS: WARNING! CURVE AHEAD!

PRESENT VELOCITY = 77.0  NO. OF GALLONS = 0.4567999
NO. OF MILES = 0.2717391  TIME PASSED = 20.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?-2

ROAD CONDITIONS: THROUGH CURVE.

PRESENT VELOCITY = 38.0  NO. OF GALLONS = 0.4567999
NO. OF MILES = 0.3369565  TIME PASSED = 30.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?6

ROAD CONDITIONS: VEHICLE AHEAD 500 FEET.

PRESENT VELOCITY = 66.0  NO. OF GALLONS = 0.4351999
NO. OF MILES = 0.4884347  TIME PASSED = 40.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?4

ROAD CONDITIONS: VEHICLE PASSED BY 31.0 MPH.

PRESENT VELOCITY = 72.0  NO. OF GALLONS = 0.4287999
NO. OF MILES = 0.6369564  TIME PASSED = 50.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?3

ROAD CONDITIONS: VEHICLE AHEAD 500 FEET.

PRESENT VELOCITY = 67.0  NO. OF GALLONS = 0.4099999
NO. OF MILES = 0.7826086  TIME PASSED = 60.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?4

ROAD CONDITIONS: VEHICLE PASSED BY 28.0 MPH.

PRESENT VELOCITY = 72.0  NO. OF GALLONS = 0.3955999
NO. OF MILES = 0.9391303  TIME PASSED = 70.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?3

■

■

■

■

■

PRESENT VELOCITY = 79.0  NO. OF GALLONS = 0.1831998
NO. OF MILES = 3.395648  TIME PASSED = 280.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?4

ROAD CONDITIONS: VEHICLE AHEAD 500 FEET.

PRESENT VELOCITY = 68.0  NO. OF GALLONS = 0.1687998
NO. OF MILES = 3.569561  TIME PASSED = 290.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?4

ROAD CONDITIONS: VEHICLE PASSED BY 26.0 MPH.

PRESENT VELOCITY = 68.0  NO. OF GALLONS = 0.1543998
NO. OF MILES = 3.743475  TIME PASSED = 300.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?4

ROAD CONDITIONS: VEHICLE AHEAD 500 FEET.

PRESENT VELOCITY = 68.0  NO. OF GALLONS = 0.1399998
NO. OF MILES = 3.917387  TIME PASSED = 310.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?4

ROAD CONDITIONS: VEHICLE PASSED BY 33.0 MPH.

PRESENT VELOCITY = 80.0  NO. OF GALLONS = 0.1255998
NO. OF MILES = 4.08913  TIME PASSED = 320.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?10

ROAD CONDITIONS: CLEAR AND STRAIGHT.

PRESENT VELOCITY = 120.0  NO. OF GALLONS = 0.595998E-01
NO. OF MILES = 4.369568  TIME PASSED = 330.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?1:\0

ROAD CONDITIONS: VEHICLE AHEAD 500 FEET.

PRESENT VELOCITY = 158.0  NO. OF GALLONS = 0.5359984E-01
NO. OF MILES = 4.713839  TIME PASSED = 340.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?0

ROAD CONDITIONS: VEHICLE PASSED BY 52.0 MPH.

PRESENT VELOCITY = 96.0  NO. OF GALLONS = 0.5359984E-01
NO. OF MILES = 4.921734  TIME PASSED = 350.0 SECONDS.
WHAT IS YOUR NEW RATE OF GAS?0

ROAD CONDITIONS: CLEAR AND STRAIGHT.

PRESENT VELOCITY = 55.0  NO. OF GALLONS = 0.5359984E-01
NO. OF MILES = 5.047828  TIME PASSED = 360.0 SECONDS.


YOU MADE IT (LUCK) !!!!
YOU WANT TO TRY IT AGAIN? N
```



BUG FOUND & EXTERMINATED

A minor bug has been discovered and corrected by the program authors. Since no complaints have been received by SCELBAL users it is assumed that the bug was in the latent stage! The bug would appear under the conditions illustrated here when an error condition occured in a FOR/NEXT loop. Once an error message was generated, the interpreter would continue to display an error message even after the error producing fault had been removed from the high level program. This only occured when an array variable was used in the loop. An example of the problem is illustrated from an actual print-out presented below. Note that even after the range of X is changed from an invalid argument for a square root operation (-1) to a valid argument (0) that the "SQ" error message continues to be generated.

```
10 DIM A(5)
15 FOR X=1 TO 5
20 LET A(X)=SQR(X)
25 PRINT X;A(X)
30 NEXT X
35 END

RUN
 1.0            1.0
 2.0            1.414213
 3.0            1.732051
 4.0            2.0
 5.0            2.236068

READY

15 FOR X=-1 TO 5

RUN
SQ AT LINE 20

READY

15 FOR X=0 TO 5

RUN
SQ AT LINE 20
```

The bug is caused by a failure to reset the ARRAY/VARIABLES flag (PG 27 LOC 201) when an error condition causes an abnormal exit. The problem is easily corrected by adding a small patch to insure that the ARRAY/VARIABLES flag is always reset after an error message is displayed. A suitable patch may be installed beginning at PG 11 LOC 307 after changing the instruction at PG 12 LOC 354 from:

JMP EXEC

to:

JMP PATCH3

PATCH3 simply consist of the following sequence:

```
PATCH3,  LLI 201      Pntr to A/V storage
         LHI 027      ** Pntr to A/V page
         LMI 000      Clear A/V flag
         JMP EXEC     Now go back to Exec
```

The object code for the patch for an 8008 would appear as:

```
11 307  066 201  PATCH3, LLI 201
11 311  066 027     **    LHI 027
11 313  076 000           LMI 000
11 315  104 266 010       JMP EXEC

12 354  104 307 011       JMP PATCH3
```

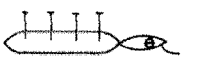
While the object code for an 8080 would appear as:

```
11 307  056 201  PATCH3, LLI 201
11 311  046 027     **    LHI 027
11 313  066 000           LMI 000
11 315  303 266 010       JMP EXEC

12 354  303 307 011       JMP PATCH3
```

Tha actual print-out below illustrates how the bug is eliminated by the above patch. The first time the program is executed after the patch is installed the error condition is displayed because the A/V flag has still not been reset. However, the execution of the patch causes the ARRAY/VARIABLES flag to be properly reset and thereafter the program executes properly.

```
READY

LIST
10 DIM A(5)
15 FOR X=0 TO 5
20 LET A(X)=SQR(X)
25 PRINT X;A(X)
30 NEXT X
35 END

READY

RUN
SQ AT LINE 20

READY

RUN
 0              0
 1.0            1.0
 2.0            1.414213
 3.0            1.732051
 4.0            2.0
 5.0            2.236068

READY
```

Users may desire to paste this patch notice into one of the NOTES pages at the back of their copies of SCELBAL, or to copy this information into their books for safekeeping.



Getting More FOR your NEXT!

Sometimes it is desirable to be able to jump to a new level of a nested FOR/NEXT loop before a loop has been completed. In the original version of SCELBAL a direct attempt to do so will result in an error message. An improvement to SCELBAL is presented here that will allow the interpreter to jump to a new level in a series of nested FOR/NEXT loops without causing an error message. This is accomplished by inserting a few instructions in the original NEXT statement routine. The instructions that are inserted cause the entire contents of the FOR/NEXT software stack to be searched for a variable name indicated in a NEXT statement (instead of simply examining the top-most variable name as was the case in the original version). Now, an error condition (FN) message will not be displayed unless the specified variable is not present anywhere in the FOR/NEXT stack. (Previously it was displayed if the specified variable was not in the top position of the FOR/NEXT stack.) This slight improvement in FOR/NEXT statement execution is provided as a suggestion for improved performance. It is not a correction to the program. If you do not desire the added feature, don't waste time adding it to your version.

The upgrading may be accomplished using patching techniques by simply inserting the instructions bracketed by the asterisks in the accompanying listing between the instructions JTZ NEXT4 and FORNXT, LAI 306 (lines 22 and 23) of the source listing on page 36 in chapter six. Or, the entire block of code from address PG 30 LOC 013 to PG 31 LOC 004 may be altered as presented here. The latter method conforms to the rules presented in the article "MODIFIED SCELBAL" which appeared in issue 02 of this bulletin. The squeezing in of the instructions to conform to those guidelines was accomplished by removing several "LHI XXX" instructions after careful analysis of the original coding and invoking several other memory saving instruction replacements at points denoted by arrows in the modified listing.

A short example provided below illustrates the effect of the improved capability. Note that when statement line 20 is added to the program, the original version of SCELBAL caused an error message to be displayed. The final RUN illustrates how the program executes when the modification is installed.

```
05 PRINT
10 FOR X=1 TO 3
15 FOR Y=1 TO 3
25 PRINT X;
30 PRINT TAB(6);Y
35 NEXT Y
40 NEXT X
45 END

READY

RUN
 1.0    1.0
 1.0    2.0
 1.0    3.0
 2.0    1.0
 2.0    2.0
 2.0    3.0
 3.0    1.0
 3.0    2.0
 3.0    3.0

READY

20 IF Y>2 GOTO 40
```

```
05 PRINT
10 FOR X=1 TO 3
15 FOR Y=1 TO 3
20 IF Y>2 GOTO 40
25 PRINT X;
30 PRINT TAB(6);Y
35 NEXT Y
40 NEXT X
45 END

READY

RUN
 1.0    1.0
 1.0    2.0
FN AT LINE 40

RUN
 1.0    1.0
 1.0    2.0
 2.0    1.0
 2.0    2.0
 3.0    1.0
 3.0    2.0
```

```
ADDR      8008        8080        MNEMONICS
030 013  066 144      056 144      NEXT, LLI 144
030 015  076 000      066 000      LMI 000
030 017  066 202      056 202      LLI 202
030 021  317          106          LBM
030 022  010          004          INB
030 023  061          055          DCL
030 024  371          160          LMB
030 025                           /
030 025  066 201      056 201      NEXT1, LLI 201
030 027  106 240 002  315 240 002  CAL GETCHR
030 032  150 042 030  312 042 030  JTZ NEXT2
030 035  066 144      056 144      LLI 144
030 037  106 314 002  315 314 002  CAL CONCT1
030 042                           /
030 042  066 201      056 201      NEXT2, LLI 201
030 044  106 003 003  315 003 003  CAL LOOP
030 047  110 025 030  302 025 030  JFZ NEXT1
030 052  066 144      056 144      LLI 144
030 054  307          176          LAM
030 055  074 001      376 001      CPI 001
030 057  110 066 030  302 066 030  JFZ NEXT3
030 062  066 146      056 146      LLI 146
030 064  076 000      066 000      LMI 000
030 066                           /
030 066  066 205      056 205      NEXT3, LLI 205
030 070  056 027      046 027      LHI 027
030 072  307          176          LAM
030 073  002          007          RLC
030 074  002          007          RLC
030 075  004 136      306 136      ADI 136
030 077  360          157          LLA
030 100  036 026      026 026      LDI 026
030 102  046 145      036 145      LEI 145
030 104  016 002      006 002      LBI 002
030 106  106 370 002  315 370 002  CAL STREPC
030 111  150 135 030  312 135 030  JTZ NEXT4
030 114                           /*************
030 114  066 205      056 205      LLI 205
030 116  056 027      046 027      LHI 027
030 120  317          106          LBM
030 121  011          005          DCB
030 122  371          160          LMB
030 123  110 066 030  302 066 030  JFZ NEXT3
030 126                           /*************
030 126  006 306      076 306      FORNXT, LAI 306
030 130  026 316      016 316      LCI 316
030 132  104 226 002  303 226 002  JMP ERROR
030 135                           /
030 135  066 360      056 360      NEXT4, LLI 360
030 137  056 026      046 026      LHI 026
030 141  337          126          LDM
030 142  060          054          INL
030 143  347          136          LEM
030 144  060          054          INL
030 145  373          162          LMD
030 146  060          054          INL
030 147  374          163          LME
030 150  066 205      056 205      LLI 205
030 152  050          044          INH
030 153  307          176          LAM
030 154  002          007          RLC
030 155  002          007          RLC
030 156  004 134      306 134      ADI 134
030 160  360          157          LLA
030 161  337          126          LDM
030 162  060          054          INL
030 163  347          136          LEM
030 164  066 360      056 360      LLI 360
030 166  051          045          DCH
030 167  373          162          LMD
030 170  060          054          INL
030 171  374          163          LME
030 172  353          142          LHD
030 173  364          153          LCE
```

```
ADDR      8008        8080        MNEMONICS
030 174  036 026      026 026      LDI 026
030 176  046 000      036 000      LEI 000
030 200  106 046 012  315 046 012  CAL MOVEC
030 203  066 325      056 325      LLI 325
030 205  056 001      046 001      LHI 001
030 207  106 012 013  315 012 013  CAL INSTR
030 212  304          173          LAE
030 213  240          247          NDA
030 214  150 126 030  312 126 030  JTZ FORNXT
030 217  004 002      306 002      ADI 002
030 221  066 276      056 276      LLI 276
030 223  056 026      046 026      LHI 026
030 225  370          167          LMA
030 226  066 330      056 330      LLI 330
030 230  056 001      046 001      LHI 001
030 232  106 012 013  315 012 013  CAL INSTR
030 235  304          173          LAE
030 236  240          247          NDA
030 237  110 302 030  302 302 030  JFZ NEXT5
030 242  066 004      056 004      LLI 004
030 244  056 001      046 001      LHI 001
030 246  106 244 022  315 244 022  CAL FLOAD
030 251  066 304      056 304      LLI 304
030 253  106 255 022  315 255 022  CAL FSTORE
030 256  361          150          LLB
030 257  056 026      046 026      LHI 026
030 261  317          106          LBM
030 262  066 277      056 277      LLI 277
030 264  371          160          LMB
030 265  106 224 003  315 224 003  CAL EVAL
030 270  066 310      056 310      LLI 310
030 272  056 001      046 001      LHI 001
030 274  106 255 022  315 255 022  CAL FSTORE
030 277  104 353 030  303 353 030  JMP NEXT6
030 302                           /
030 302  041          035          NEXT5, DCE
030 303  066 277      056 277      LLI 277
030 305  056 026      046 026      LHI 026
030 307  374          163          LME
030 310  106 224 003  315 224 003  CAL EVAL
030 313  066 310      056 310      LLI 310
030 315  056 001      046 001      LHI 001
030 317  106 255 022  315 255 022  CAL FSTORE
030 322  066 277      056 277      LLI 277
030 324  056 026      046 026      LHI 026
030 326  307          176          LAM
030 327  004 005      306 005      ADI 005
030 331  061          055          DCL
030 332  370          167          LMA
030 333  066 000      056 000      LLI 000
030 335  317          106          LBM
030 336  066 277      056 277      LLI 277
030 340  371          160          LME
030 341  106 224 003  315 224 003  CAL EVAL
030 344  066 304      056 304      LLI 304
030 346  056 001      046 001      LHI 001
030 350  106 255 022  315 255 022  CAL FSTORE
030 353                           /
030 353  066 144      056 144      NEXT6, LLI 144
030 355  056 026      046 026      LHI 026
030 357  371          160          LMB
030 360  066 034      056 034      LLI 034
030 362  050          044          INH
030 363  106 012 013  315 012 013  CAL INSTR
030 366  304          173          LAE
030 367  240          247          NDA
030 370  066 202      056 202      LLI 202
030 372  056 026      046 026      LHI 026
030 374  370          167          LMA
030 375  150 126 030  312 126 030  JTZ FORNXT
031 000  004 003      306 003      ADI 003
031 002  066 203      056 203      LLI 203
031 004  370          167          LMA
```

SCELBAL UPDATE

ISSUE 04 - 1/77

SCELBAL II UNDER DEVELOPMENT

As SCELBAL owners know, SCELBAL was developed primarily for 8008 system owners. There were several reasons for doing so. First, when SCELBI COMPUTER CONSULTING, INC., first went into business, it produced a microcomputer based on the 8008 CPU. A number of those systems are still out in the field and many owners had indicated a desire to have the capabilities of a high level program available. We no longer manufacture microcomputer systems, but we felt an obligation towards those who had helped us pioneer in the field of the personal computer.

Second, in addition to those 8008 microcomputer systems sold by SCELBI, there were several thousand similar systems (8008 based) known to be in existence produced by other early microcomputer system manufacturers along with numerous personal systems based on the MARK-8 article that appeared in RADIO ELECTRONICS magazine some two years ago. Many of these people had written to us indicating that they felt the rapid growth of the acceptance of the 8080 and other more advanced CPUs, and the attention they were getting, would leave the early 8008 users high and dry without ever having a high level language developed for it.

Third, we felt that developing such an interpreter for a micro CPU as primitive as the 8008 is now considered, instead of being a waste of time (as apparently everyone else thought it would), would be a valuable experience. After all, if it could be accomplished for such a primitive CPU, upgrading the fundamental concepts and routines from that point to take advantage of the increased power of instruction sets available on more advanced CPUs would be a pretty straightforward task.

Additionally, we of course knew that an interpreter written for an 8008 could be directly assembled to operate on an 8080 even if it was not "efficient" in making use of that CPU's capabilities. This meant though, that many users who were planning on eventually upgrading their personal systems from an 8008 to an 8080, with the existence of SCELBAL, could do so without having to modify a single one of their SCELBAL higher level programs!

Finally, it was felt that presenting SCELBAL in detail, with complete source listings, flow charts, etc., for the primitive 8008 CPU, in the manner in which it was done (not using any of page zero, not trying fancy packing tricks, etc.) would result in an information source which users could have fun with! One can pick almost any section one might be interested in and find ways to improve it by using better coding techniques, etc. 8080 owners, as pointed out in chapter fifteen, could go to work with vigor on compacting the program if they so desired. (The key here is that those upgrading from an 8008 to an 8080 do NOT have to modify the interpreter to increase its efficiency if they are not interested in doing so!)

More than all those factors combined, however, SCELBAL was developed with the intention that it become an ever-evolving program. As new machine types became available, SCELBAL could be adapted. As users became more sophisticated in their demands for program performance, SCELBAL could be upgraded. Since the entire fundamental organization and logic of the interpreter had been presented, users would not be forced to wait for such advances to come from SCELBI if they had the desire and capabilities to proceed on their own!

Naturally, many users of SCELBAL do not wish to become involved with the intimate details of the interpreter's operation. They just want to be able to use the end result. Fine. SCELBI intends to continue to improve the program as well as to provide the language for other types of microcomputers when it appears that there is a market sufficient enough to justify the expense. It is hoped that by listening to the thoughts of many other users, and by providing an opportunity for others to communicate their needs, the overall quality and capability of SCELBAL can be improved. Indeed, there is no end in sight to the potential. The limiting factor, as in most endeavors, is time and money.

Even as the first copy of SCELBAL was published, work was underway to produce a revised version that would capitalize on the increased power of the 8080 instruction set (over that of the 8008). Work is proceeding smoothly. Feedback from SCELBAL customers who are 8080 system owners indicate they are highly interested in such a revised package.

Essentially, the revised version titled SCELBAL II will simply be a compressed version of the original program. It will remain organized in essentially the same manner, using the same subroutine names etc., so that the original publication will initially remain as the prime reference. Preliminary indications are that the 8080 customized version, with DIMension capability, will reside in about 5K of memory (without using page zero). A few minor operating improvements (such as increasing the number of variable names allowed) are planned. The possibilities for the inclusion of other features remains open at this point pending feedback from users. (By this it is meant operating improvements. The addition of extended functions such as sines, cosines, exponents, string handling capabilities and so forth constitute not merely improvements, but actually the creation of additional features. More has and will be said about such matters in other articles.)

How long before SCELBAL II will be released? Probably another five or six months. We want to provide time for plenty of feedback from users to try and catch any gremlins or add needed improvements. Registered SCELBAL owners will be notified when SCELBAL II is available. Chances are, you will hear more about its development in these pages as it progresses.

In the meantime, if your interested (even anxious?) to work on such a project yourself, the following information may help you get off to a smooth start. Reversing the storage format for three critical double-byte values used in SCELBAL will enable one to capitalize on using a number of the 8080 double-byte manipulating instructions. These storage locations are all on page 26 (octal). They are the locations used to hold the User Program Line Pointer (360 & 361), the Auxiliary Program Buffer Pointer (362 & 363) and the End Of Buffer Pointer (364

and 365). Values placed in these locations in the original SCELBAL version are in the order of PAGE ADDRESS followed by LOW ADDRESS. Reversing the order to LOW ADDRESS followed by PAGE ADDRESS makes it possible to use 8080 instructions such as "SHLD" when manipulating data for those locations etc.

These locations are referred to at numerous points throughout SCELBAL. The following lists all the points known to us at the time of this writing and indicates the new contents of those locations if one wants to set things up so that the LOW ADDRESS value is followed by the PAGE ADDRESS in those storage locations. It is recommended that these changes ONLY BE INCORPORATED IF THE USER INTENDS TO TINKER WITH CUSTOMIZING THE PROGRAM FOR AN 8080 SYSTEM. There is no other reason for making the changes if such is not the case! Consequently, the revisions are shown only for the 8080 version with appropriate 8080 codes.

CHANGES AFFECTING
USER PGM LINE POINTER
(PAGE 26 LOCS 360/361)

ADDR	CONTS
11 132	000
11 135	033
11 173	000
11 176	033
11 257	146
11 260	151
11 275	146
11 276	152
11 365	146
11 366	151
12 011	136
12 013	126
12 031	136
12 033	126
12 077	136
12 101	126
12 115	163
12 117	162
12 130	136
12 132	126
13 107	000
13 112	033
13 122	136
13 124	126
13 140	163
13 142	162
13 164	146
13 165	151

CHANGES AFFECTING
USER PGM LINE POINTER
(PAGE 26 LOCS 360/361)

ADDR	CONTS
15 255	000
15 260	033
15 330	146
15 331	151
15 362	136
15 364	126
16 000	163
16 002	162
16 252	136
16 254	126
16 341	163
16 343	162
17 211	136
17 213	126
30 134	136
30 136	126
30 164	163
30 166	162
31 153	162
31 155	163

CHANGES AFFECTING
AUX PGM BUFFER POINTER
(PAGE 26 LOCS 362/363)

ADDR	CONTS
30 140	163
30 142	162
31 147	126
31 151	136

CHANGES AFFECTING
END OF BUFFER POINTER
(PAGE 26 LOCS 364/365)

ADDR	CONTS
11 017	000
11 022	033
12 170	365
12 174	055
12 201	054
12 206	365
12 212	055
12 265	136
12 267	126
12 273	162
12 275	163
16 004	365
16 012	055

LETTERS

I don't know how many people might be interested in the following modification for SCELBAL but it is very useful to me and saves much time compared with doing the same thing without a computer.

From time to time I find it desirable to rearrange a table of data so that the lines are arrayed in numerical order from top to bottom. One way to do this is to use the SCELBAL program entry routines, entering the other columns as statement text. This works fine except when two or more lines have the same number. One way to overcome this is to rearrange the routines in NOTEND so that statements with the same number are entered without deleting the earlier statement. The changes still allow a statement to be deleted, by entering only the statement number. The rearranged list is obtained by entering a LIST command. To fool the syntax error-checking routines, an "equal" sign or a left hand parenthesis is entered following the statement number. The modified program can still be used for its original purpose, but it will be necessary to enter a statement number by itself to remove a line. The purist can maintain two versions of this portion of SCELBAL.

One advantage of this method is the large buffer space available. Another advantage is that the data is easily stored by using the SAVE command.

Mr. S.J. Toy
Chico, CA

(A listing of the modification for the 8008 version of SCELBAL is provided below. A sample of the modified program in operation was submitted but is not shown for space considerations. It appeared to operate as intended. Looks like a clever way in which to utilize the program's built-in editor as a sorting routine! - Ed.)

```
11 354  006 203          LLI 203          See if line no. only
11 356  056 026      **  LHI 026
11 360  307              LAM
11 361  240              NDA
11 362  110 005 012      JFZ NOSAME       Line no. only if zero
11 365  066 360          LLI 360          Remove line
11 367  056 026      **  LHI 026
11 371  327              LCM
11 372  060              INL
11 373  367              LLM
11 374  352              LHC
11 375  317              LBM
11 376  010              INB
11 377  106 144 012      CAL REMOVE
12 002  104 266 010      JMP EXEC
```

HEY! WE FORGOT TO TELL YOU.....

The ROADRACE program presented in ISSUE 03 of SCELBAL UPDATE was provided courtesy of CREATIVE COMPUTING! The magazine CREATIVE COMPUTING is published by an enthusiastic and creative organization headed by David H. Ahl. In addition to games such as that presented in ISSUE 03, the magazine regularly presents a variety of articles, book and product reviews, educational material, and a good selection of general information which we feel most of our customers would find highly interesting. Recent issues of the magazine contained 88 pages or more in an 8 1/2 by 11 format. Considering the fact that there is relatively little advertising space allotted in those 88 plus pages, the amount of text and editorial material per issue far exceeds most other computer-related publications that we have seen of late. Individuals interested in subscribing to CREATIVE COMPUTING may do so at the following rates. 1 year - $8.00, 3 years - $21.00. If you have any doubts, you may obtain a sample copy of a recent issue for $1.50. (The magazine is published bimonthly.) Subscription orders may be forwarded directly to the publisher:

CREATIVE COMPUTING
P.O. Box 789-M
Morristown, NJ 07960

OOPS!

I believe I have found 2 errors in SCELBAL which have not been mentioned in your UPDATES.

1) 11 030 is 001 should be 000
2) 26 364 is 000 should be 033

In the first case, use of SCR command causes the first regular variable location to become unavailable. You are thereafter limited to 19 regular variables.

In the second case, INSERT picks up the 000 and uses it as a high address with results which vary but are generally disastrous. Use of SCR replaces this 000 with 033 and that makes everything fine.

String variables sound great. I get the feeling that my poor little 8008's 16K limit is going to be reached soon.

A suggestion: We need a cassette data read data write capability. I've tried to use the arrays values block as a means to do this, but I was not happy with my results. SCELBAL should be able to analyse a checking account on tape as well as format the data into records organized into blocks for recording.

Thanks for SCELBAL. It is a lot of fun.

James C. Tucker
Exeter, NH

(Thank you James! Looks like you have found the bug that was bugging several people in regards to the disappearing variable storage location. Seems if you just loaded the program into memory and started operating you could have 20 variables. Later, after using a SCR command you only had 19! Nice piece of detective work.

We hadn't received any complaints regarding the second item you noticed. Probably because most people took the advice given in chapter fourteen to use the SCR command when starting to use SCELBAL. But it could certainly cause a problem as you pointed out and is likely to occur if one, for instance, uses the LOAD command and proceeds to revise a user program without having used an SCR command.

We strongly recommend that readers take James suggestions and change the two bytes indicated to avoid similar problems in their systems. As for you James, your detective work has earned you an hororarium check that should buy quite a few stamps in case you need to report any similar discoveries - which we hope you will not! - Ed)

STRINGS SUPPLEMENT

NOW AVAILABLE

The Strings Supplement to SCELBAL is now available. The 68 page booklet (8 1/2 X 11) may be obtained for $10.00 from the publisher at the address shown below. The booklet provides the source code and assembled object listings for both 8008 and 8080 systems for routines that will enable SCELBAL users to add String Function capabilities to their systems. Users intending to add the Strings capabilities should have a minimum of 12K memory (read and write) available in their system.

Details of the Strings Supplement capabilities were provided in Issue 03 of SCELBAL UPDATE.

The $10.00 price for the STRINGS SUPPLEMENT includes postpaid delivery by U.S. Mail service. Address orders to:

ORDER DEPARTMENT
SCELBI C.C., INC.
PO BOX 133 - PP STN
MILFORD, CT 06460

COMING SOON!

EXTENDED MATHEMATICAL FUNCTIONS FOR SCELBAL

Now in the final documentation stages are five extended mathematical functions soon to be made available for SCELBAL users. The new functions, which will be made available as a supplemental publication, will provide users with the following additional capabilities when installed: SIN, COS, EXP(e), LOG(e), and ATN. The SIN and LOG functions are calculated using Chebyshev optimized Taylor series. The EXP and ATN are calculated using continued fractions. The COS function is calculated using the SIN function. The argument of any function is reduced to an interval where the Taylor series or continued fractions is reasonably accurate. The argument range for the functions will be as follows:

SIN $-4194303 < X < 4194303$
COS $-4194303 < X < 4194303$
EXP $-89 < X < 89$
LOG $X > 0$
ATN $-1E37 < X < 1E37$

The soon to be available booklet will contain source and object listings as in other publications related to SCELBAL. Prospective String Function users should note that assembled object listings for the mathematical functions will reside in some of the same memory locations (pages 50 through 54 octal) as various string routines. This overlapping was based on the premise that from memory space considerations (particularly for 8008 based systems) users would not find it practical to have both string functions and mathematical functions installed at the same time. (String function users theoretically are less likely to be concerned with extended mathematical functions it seems.) Users who might desire to have both types of capabilities installed simultansously would need to relocate one set of routines and would probably want to have 16K or more of read and write memory available in the system.

It is anticipated that the extended mathematical function routines will be available in the form of a supplementary booklet near the latter part of February, 1977. Price of the supplement has been pegged initially at $5.00 including postpaid delivery by U.S. Mail.

PREMIUMS FOR YOUR
PROGRAMS
APPLICATION NOTES
ARTICLES
COMMENTS

If you have developed your own original programs to perform tasks that may be of interest to other SCELBAL users, chances are you are in a position to pick up a bit of cash! User submitted programs accepted for publication by SCELBI earn an honorarium check and a nice certificate attesting to the author's performance! We are particularly interested in programs that may be of value to scientists, engineers, and small businessmen. However, games, and general purpose routines are frequently accepted.

But, you don't have to be a SCELBAL programmer to earn some coins. We are also interested in seeing articles of general interest to SCELBAL users, as well as application notes, and even comments or suggestions!

You may submit your efforts to the address given below. Material accepted for publication earns the author an honorarium check based on originality, usefullness to readers, length, completeness and quality of presentation etc.. Submissions accepted for publication become the property of SCELBI C.C., Inc.. The act of submitting for publication is certification that the material is original and that the author agrees to the terms of this announcement. While every attempt will be made to return rejected material accompanied by a SASE (self-addressed, stamped envelope) SCELBI C.C., Inc. assumes no responsibility for submitted material.

Material to be considered for publication should be forwarded to:

SCELBAL UPDATE EDITOR
SCELBI C.C., INC.
PO BOX 133 - PP STN
MILFORD, CT. 06460

SCELBAL UPDATE

ISSUE 05 - 6/77

UNLIMITED! (WELL - ALMOST) VARIABLE NAMES!

One of the improvements most often suggested for SCELBAL is to increase the number of variable names allowed. The original version allowed a total of 20 regular variable names. It was possible to increase the effective number of variables in a system having DIM capability installed, but even when performing "tricks" such as that, the number of variable names was limited to a maximum of 84. A good many users felt it would be nice to substantually increase the number of variable names allowed in a program - without having to snitch from elements in an array.

O.K.! Here it is - a modification to SCELBAL that will theoretically allow you to have as many variables as can be defined by valid two character symbolic names, provided you have enough memory in your system to store all the variables desired!

Essentially, the modification changes SCELBAL so that it stores variable names and their values starting at the top (highest allowable address value) of the User's Program Buffer and works downward toward the source code in the buffer which is stored in ascending address values as new lines are entered. The variable names table previously assigned to Page 27 starting at Location 210 is no longer used if the user elects to install this modification.

Listings of the modification for both 8008 and 8080 machines are included. The routines shown may be simply "overlaid" over the original routines.

Several notes of caution are in order. First, the modification as shown in the accompanying listings is for the essentially unmodified version of SCELBAL as presented in the basic publication. If you have made modifications to your version - be careful. Same goes if you have implemented any of the supplements.

In particular, if you have been playing around with compacting SCELBAL for an 8080 machine and have changed the order of the bytes stored in the End of User Program Buffer Pointer (Page 26, Locations 364, 365) as mentioned in SCELBAL UPDATE Issue 04, you will have to change things around a little bit in the accompanying listing in the vicinity of the LOOKU3 subroutine at Page 05 Location 157 etc.

If you have installed Strings or Mathematical Supplements, or if your User Program Buffer storage area does not end at Page 54 Location 377 in your system, you will need to alter the values in the accompanying listing marked with a "$$" notation in the comments section (such as Page 05 Location 54 and Page 11 Location 44) so that the end of the User Program Buffer storage area is set up properly by the new unlimited variables modification routines.

It is assumed that those who have otherwise modified SCELBAL or relocated the program, will know how to proceed to adapt the modification.

Finally, a note of caution. The modification checks to see that variables do not run into a user's source listing. However, no check is made to see that the user buffer does not run into the variables table. It is thus theoretically possible to "bomb" the variables table if one was, for instance, inserting new lines into a source listing and alternating with the RUN mode to test the operation of the program being developed. If it looks like storage will be tight in a program; load the source *entirely* before executing a RUN command! Since variable names are added to the variables table as a program is executed, the modified program will indicate if buffer space is exhausted.

Have fun with the new capability!

LISTING FOR AN 8008

```
000 000                       /
000 000                       ORG 005 033
005 033                       /
005 033  106 045 005  LOOK, CAL NEWVT   /CALL NEW VAR STORAGE RTN
005 036  240                  NDA       /CHECK STATUS ON RETURN
005 037  150 155 010  JTZ LOOKU4        /IF FOUND MATCH IN TBL - PROCESS
005 042  104 135 010  JMP LOOK3A        /IF HAVE EOT - ADD ENTRY TO VT
005 045                       /
005 045  066 120      NEWVT, LLI 120    /POINTER TO SYMBOL
005 047  056 026      LHI 026           /**BUFFER STORAGE AREA
005 051  046 377      LEI 377           /POINTER TO START OF
005 053  036 054      LDI 054           /$$ NEW VARS STORAGE AREA
005 055  307          LAM               /FETCH (CC) OF STRING IN BFR
005 056  074 001      CPI 001           /SEE IF IT IS EQUAL TO ONE
005 060  110 067 005  JFZ LOOKUA        /JUMP AHEAD IF NOT EQUAL TO ONE
005 063  066 122      LLI 122           /ELSE SET PNTR AND CLEAR 2ND
005 065  076 000      LMI 000           /BYTE OF NAME TO ZERO
005 067  353          LOOKUA, LHI       /SET POINTER TO
005 070  364          LLE               /FIRST LOCATION
005 071  307          LAM               /IN VARIABLES TABLE
005 072  240          NDA               /SEE IF EQUAL TO ZERO
005 073  150 150 005  JTZ LOOKU3        /IF SO, NOTHING IN TABLE
005 076                       /
005 076  066 121      LOOKU1, LLI 121   /SET POINTER TO 1ST CHARACTER
005 100  056 026      LHI 026           /**OF NAME IN THE SYMBOL BFR
005 102  106 356 022  CAL SWITCH        /SAVE IN D&E AND FETCH
005 105  307          LAM               /POINTER TO VT, THEN FETCH
005 106  061          DCL               /FIRST ENTRY TO THE ACC
005 107  317          LBM               /AND 2ND ENTRY TO REG B
005 110  106 164 003  CAL DEC           /DECREMENT VT PNTR ONCE MORE
005 113  106 356 022  CAL SWITCH        /SAVE VT POINTER AND GET SB
005 116  277          CPM               /POINTER. SEE IF HAVE SAME
005 117  110 132 005  JFZ LOOKU2        /NAME. TO NEXT ENTRY IF
005 122  060          INL               /NOT. BUT, IF FIRST LETTER
005 123  301          LAB               /MATCHES - THEN TRY
005 124  277          CPM               /SECOND. IF FIND NAME
005 125  110 132 005  JFZ LOOKU2        /MATCHES CAN STORE VALUE
005 130  250          XRA               /SO CLEAR ACC TO INDICATE
005 131  007          RET               /MATCH, THEN RETURN TO CALLER
005 132                       /
005 132  016 004      LOOKU2, LBI 004   /PUT 4 INTO REGISTER B
005 134  353          LHD               /FETCH VARIABLES TABLE
005 135  364          LLE               /POINTER INTO REGS H&L
005 136  106 113 003  CAL SUBHL         /SUBTRACT 4 FROM PNTR VALUE
005 141  307          LAM               /FETCH FM ADDR POINTED TO
005 142  335          LDH               /SAVE VARIABLES TABLE
005 143  346          LEL               /POINTER IN D&E
005 144  240          NDA               /TEST LAST BYTE FROM VT
005 145  110 076 005  JFZ LOOKU1        /IF NOT EOT, CONT SEARCH
005 150                       /
005 150  016 006      LOOKU3, LBI 006   /IF FOUND EOT
005 152  106 113 003  CAL SUBHL         /SUBTRACT 6 FROM PNTR AND
```

```
005 155  335          LDH                 /SAVE VARIABLES TABLE
005 156  346          LEL                 /POINTER IN D&E
005 157  056 026      LHI 026             /**SET POINTER TO END   ****
005 161  066 364      LLI 364             /OF USER PROGRAM BUFFER ****
005 163  307          LAM                 /FETCH EOB PAGE VALUE
005 164  273          CPD                 /COMPARE WITH VT PNTR VALUE
005 165  160 176 005  JTS OKDOK2          /IF POS HERE, NO CONFLICT
005 170  060          INL                 /IF NOT, FETCH LOW ADDR
005 171  307          LAM                 /OF END OF USER PGM BF PNTR
005 172  274          CPE                 /AND TEST FOR ROOM ON PAGE
005 173  100 222 002  JFC BIGERR          /IF NOT, HAVE AN ERROR!
005 176  106 356 022  OKDOK2, CAL SWITCH/IF OK, RESTORE VT PNTR
005 201  076 000      LMI 000             /TO H&L AND MAKE EOT MARKER
005 203  106 174 003  CAL INDEXB          /ADD 6 BACK TO VT PNTR
005 206  106 356 022  CAL SWITCH          /SAVE VT PNTR IN D&E
005 211  066 121      LLI 121             /SET PNTR TO 1ST CHAR IN SB
005 213  307          LAM                 /FETCH 1ST CHARACTER TO ACC
005 214  060          INL                 /ADVANCE BUFFER POINTER
005 215  317          LBM                 /FETCH 2ND CHAR TO REG B
005 216  353          LHD                 /GET VARIABLES TABLE
005 217  364          LLE                 /POINTER IN H&L
005 220  370          LMA                 /STORE SYMBOL NAME
005 221  061          DCL                 /IN THE VARIABLES TABLE
005 222  371          LMB                 /- BOTH CHARACTERS -
005 223  006 377      LAI 377             /SET ACC TO ALL ONES TO FLAG
005 225  007          RET                 /JOB DONE, RETURN TO CALLER
005 226               /
000 000               ORG 010 100
010 100               /
010 100  106 045 005  STOSY1, CAL NEWVT /CALL NEW VAR STORAGE RTN
010 103  240          NDA                 /CHECK STATUS ON RETURN
010 104  150 117 010  JTZ STOSY4          /IF FOUND MATCH - PROCESS
010 107  016 004      LBI 004             /IF HAVE EOT THEN SET UP
010 111  106 113 003  CAL SUBHL           /TO ADD ENTRY
010 114  104 127 010  JMP STOSY5          /TO THE VARIABLES TABLE
010 117               /
010 117  106 356 022  STOSY4, CAL SWITCH/RESTORE VT POINTER TO H&L
010 122  016 003      LBI 003             /LOAD 3 INTO REG B
010 124  106 113 003  CAL SUBHL           /SUBTRACT 3 FROM VT PNTR
010 127               /
010 127  106 255 022  STOSY5, CAL FSTORE/FPACC INTO VT LOCATIONS
010 132  104 255 002  JMP CLESYM          /CLEAR SYMBOL BF & EXIT
010 135               /
010 135  250          LOOK3A, XRA         /CLEAR THE ACCUMULATOR
010 136  106 164 003  CAL DEC             /AND PLACE
010 141  370          LMA                 /ZERO
010 142  061          DCL                 /INTO
010 143  370          LMA                 /THE
010 144  106 164 003  CAL DEC             /VARIABLES
010 147  370          LMA                 /TABLE
010 150  061          DCL                 /FOR THE
010 151  370          LMA                 /INITIAL VALUE
010 152  104 165 010  JMP LOOKU5          /GO FINISH UP
010 155               /
010 155  106 356 022  LOOKU4, CAL SWITCH/POINTER TO VT INTO H&L
010 160  016 003      LBI 003             /COUNT OF 3 INTO REG B
010 162  106 113 003  CAL SUBHL           /SUBTRACT 3 FROM VT PNTR
010 165               /
010 165  106 317 022  LOOKU5, CAL SAVEHL/SAVE VT POINTER
010 170  066 227      LLI 227             /SET UP PNTR TO ARITHMETIC
010 172  056 001      LHI 001             /**STACK POINTER
010 174  307          LAM                 /FETCH POINTER VALUE
010 175  004 004      ADI 004             /ADD 4 FOR NEW ENTRY
010 177  370          LMA                 /RESTORE STACK POINTER
010 200  360          LLA                 /AND SET UP NEW AS VALUE
010 201  106 255 022  CAL FSTORE          /PUT THE FPACC ON THE AS
010 204  106 337 022  CAL RESTHL          /RESTORE VT POINTER
010 207  106 244 022  CAL FLOAD           /PUT THE VAR INTO FPACC
010 212  104 231 005  JMP PARSE           /TO THE PARSE ROUTINE
010 215               /
010 215               ORG 011 041
011 041               /
011 041  066 377      LLI 377             /POINTER TO START OF
011 043  056 054      LHI 054             /$$ NEW VAR'S STORAGE AREA
011 045  300          LAA                 /REPLACE WITH NOP INSTRUC
011 046               /
```

LISTING FOR AN 8080

```
003 165               ORG 005 033
005 033               /
005 033  315 045 005  LOOK, CAL NEWVT   /CALL NEW VAR STORAGE RTN
005 036  247          NDA                 /CHECK STATUS ON RETURN
005 037  312 155 010  JTZ LOOKU4          /IF FOUND MATCH IN TBL - PROCESS
005 042  303 135 010  JMP LOOK3A          /IF HAVE EOT - ADD ENTRY TO VT
005 045               /
005 045  056 120      NEWVT, LLI 120      /POINTER TO SYMBOL
005 047  046 026      LHI 026             /**BUFFER STORAGE AREA
005 051  036 377      LEI 377             /POINTER TO START OF
005 053  026 054      LDI 054             /$$ NEW VARS STORAGE AREA
005 055  176          LAM                 /FETCH (CC) OF STRING IN BFR
005 056  376 001      CPI 001             /SEE IF IT IS EQUAL TO ONE
005 060  302 067 005  JFZ LOOKUA          /JUMP AHEAD IF NOT EQUAL TO ONE
005 063  056 122      LLI 122             /ELSE SET PNTR AND CLEAR 2ND
005 065  066 000      LMI 000             /BYTE OF NAME TO ZERO
005 067  142          LOOKUA, LHD         /SET POINTER TO
005 072  153          LLE                 /FIRST LOCATION
005 071  176          LAM                 /IN VARIABLES TABLE
005 072  247          NDA                 /SEE IF EQUAL TO ZERO
005 073  312 150 005  JTZ LOOKU3          /IF SO, NOTHING IN TABLE
005 076               /
005 076  056 121      LOOKU1, LLI 121     /SET POINTER TO 1ST CHARACTER
005 100  046 026      LHI 026             /**OF NAME IN THE SYMBOL BFR
```

```
005 102  315 356 022  CAL SWITCH         /SAVE IN D&E AND FETCH
005 105  176          LAM                 /POINTER TO VT, THEN FETCH
005 106  055          DCL                 /FIRST ENTRY TO THE ACC
005 107  106          LBM                 /AND 2ND ENTRY TO REG B
005 110  315 164 003  CAL DEC             /DECREMENT VT PNTR ONCE MO
005 113  315 356 022  CAL SWITCH         /SAVE VT POINTER AND GET S
005 116  276          CPM                 /POINTER. SEE IF HAVE SAME
005 117  302 132 005  JFZ LOOKU2          /NAME. TO NEXT ENTRY IF
005 122  054          INL                 /NOT. BUT, IF FIRST LETTER
005 123  170          LAB                 /MATCHES - THEN TRY
005 124  276          CPM                 /SECOND. IF FIND NAME
005 125  302 132 005  JFZ LOOKU2          /MATCHES CAN STORE VALUE
005 130  257          XRA                 /SO CLEAR ACC TO INDICATE
005 131  311          RET                 /MATCH, THEN RETURN TO CAL
005 132               /
005 132  006 004      LOOKU2, LBI 004     /PUT 4 INTO REGISTER B
005 134  142          LHD                 /FETCH VARIABLES TABLE
005 135  153          LLE                 /POINTER INTO REGS H&L
005 136  315 113 003  CAL SUBHL           /SUBTRACT 4 FROM PNTR VALU
005 141  176          LAM                 /FETCH FM ADDR POINTED TO
005 142  124          LDH                 /SAVE VARIABLES TABLE
005 143  135          LEL                 /POINTER IN D&E
005 144  247          NDA                 /TEST LAST BYTE FROM VT
005 145  302 076 005  JFZ LOOKU1          /IF NOT EOT, CONT SEARCH
005 150               /
005 150  006 006      LOOKU3, LBI 006     /IF FOUND EOT
005 152  315 113 003  CAL SUBHL           /SUBTRACT 6 FROM PNTR AND
005 155  124          LDH                 /SAVE VARIABLES TABLE
005 156  135          LEL                 /POINTER IN D&E
005 157  046 026      LHI 026             /**SET POINTER TO END   **
005 161  056 364      LLI 364             /OF USER PROGRAM BUFFER **
005 163  176          LAM                 /FETCH EOB PAGE VALUE
005 164  272          CPD                 /COMPARE WITH VT PNTR VALU
005 165  372 176 005  JTS OKDOK2          /IF POS HERE, NO CONFLICT
005 170  054          INL                 /IF NOT, FETCH LOW ADDR
005 171  176          LAM                 /OF END OF USER PGM BF PNT
005 172  273          CPE                 /AND TEST FOR ROOM ON PAGE
005 173  322 222 002  JFC BIGERR          /IF NOT, HAVE AN ERROR!
005 176  315 356 022  OKDOK2, CAL SWITCH/IF OK, RESTORE VT PNTR
005 201  066 000      LMI 000             /TO H&L AND MAKE EOT MARKE
005 203  315 174 003  CAL INDEXB          /ADD 6 BACK TO VT PNTR
005 206  315 356 022  CAL SWITCH          /SAVE VT PNTR IN D&E
005 211  056 121      LLI 121             /SET PNTR TO 1ST CHAR IN S
005 213  176          LAM                 /FETCH 1ST CHARACTER TO AC
005 214  054          INL                 /ADVANCE BUFFER POINTER
005 215  106          LBM                 /FETCH 2ND CHAR TO REG B
005 216  142          LHD                 /GET VARIABLES TABLE
005 217  153          LLE                 /POINTER IN H&L
005 220  167          LMA                 /STORE SYMBOL NAME
005 221  055          DCL                 /IN THE VARIABLES TABLE
005 222  160          LMB                 /- BOTH CHARACTERS -
005 223  076 377      LAI 377             /SET ACC TO ALL ONES TO FL
005 225  311          RET                 /JOB DONE, RETURN TO CALLE
```

```
002 223               ORG 010 100
010 100               /
010 100  315 045 005  STOSY1, CAL NEWVT /CALL NEW VAR STORAGE RTN
010 103  247          NDA                 /CHECK STATUS ON RETURN
010 104  312 117 010  JTZ STOSY4          /IF FOUND MATCH - PROCESS
010 107  006 004      LBI 004             /IF HAVE EOT THEN SET UP
010 111  315 113 003  CAL SUBHL           /TO ADD ENTRY
010 114  303 127 010  JMP STOSY5          /TO THE VARIABLES TABLE
010 117               /
010 117  315 356 022  STOSY4, CAL SWITCH/RESTORE VT POINTER TO H&L
010 122  006 003      LBI 003             /LOAD 3 INTO REG B
010 124  315 113 003  CAL SUBHL           /SUBTRACT 3 FROM VT PNTR
010 127               /
010 127  315 255 022  STOSY5, CAL FSTORE/FPACC INTO VT LOCATIONS
010 132  303 255 002  JMP CLESYM          /CLEAR SYMBOL BF & EXIT
010 135               /
010 135  257          LOOK3A, XRA         /CLEAR THE ACCUMULATOR
010 136  315 164 003  CAL DEC             /AND PLACE
010 141  167          LMA                 /ZERO
010 142  055          DCL                 /INTO
010 143  167          LMA                 /THE
010 144  315 164 003  CAL DEC             /VARIABLES
010 147  167          LMA                 /TABLE
010 150  055          DCL                 /FOR THE
010 151  167          LMA                 /INITIAL VALUE
010 152  303 165 010  JMP LOOKU5          /GO FINISH UP
010 155               /
010 155  315 356 022  LOOKU4, CAL SWITCH/POINTER TO VT INTO H&L
010 160  006 003      LBI 003             /COUNT OF 3 INTO REG B
010 162  315 113 003  CAL SUBHL           /SUBTRACT 3 FROM VT PNTR
010 165               /
010 165  315 317 022  LOOKU5, CAL SAVEHL/SAVE VT POINTER
010 170  056 227      LLI 227             /SET UP PNTR TO ARITHMETIC
010 172  046 001      LHI 001             /**STACK POINTER
010 174  176          LAM                 /FETCH POINTER VALUE
010 175  306 004      ADI 004             /ADD 4 FOR NEW ENTRY
010 177  167          LMA                 /RESTORE STACK POINTER
010 200  157          LLA                 /AND SET UP NEW AS VALUE
010 201  315 255 022  CAL FSTORE          /PUT THE FPACC ON THE AS
010 204  315 337 022  CAL RESTHL          /RESTORE VT POINTER
010 207  315 244 022  CAL FLOAD           /PUT THE VAR INTO FPACC
010 212  303 231 005  JMP PARSE           /TO THE PARSE ROUTINE
010 215               /
010 215               ORG 011 041
011 041               /
011 041  056 377      LLI 377             /POINTER TO START OF
011 043  046 054      LHI 054             /$$ NEW VAR'S STORAGE AREA
011 045  177          LAA                 /REPLACE WITH NOP INSTRUC
011 046               /
```

EXTENDED MATHEMATICAL FUNCTIONS AVAILABLE

Five extended mathematical functions are now available for SCELBAL. The new functions, made available as a supplemental publication, provide users with the following capabilities when installed: SIN, COS, EXP(e), LOG(e), and ATN.

The SIN and LOG functions are calculated using Chebyshev optimized Taylor series. The EXP and ATN are calculated using continued fractions. The COS function is calculated using the SIN function. The argument of any function is reduced to an interval where the Taylor series or continued fractions is reasonably accurate. The argument range for the functions are as follows:

SIN -4194303<X<4194303
COS -4194303<X<4194303
EXP -89<X<89
LOG X>0
ATN -1E37<X<1E37

The supplemental booklet contains source and object listings as in other publications related to SCELBAL. The assembled object listings provided reside in locations on pages 50 through 54. They may be reassembled to reside elsewhere by the user if desired. String Function users should note that those same pages are used by sections of the String Functions.

The price of the Mathematical Supplement to SCELBAL is $5.00 in the U.S. including U.S. mail delivery. Foreign purchasers should include $2.00 for airmail delivery of the supplement.

A FEW CORRECTIONS

C. A. Bannister of Richmond, VA, was the first to report some object code errors in the listing for modified SCELBAL shown on page 3 of SCELBAL UPDATE Issue 02. The object code errors only occurred in the 8008 listing.

It seems that the object codes for XRA, LMA and LLA directives got fouled up in the listing. The code for XRA should be 250, for LMA it is 370 and for LLA it is 360.

Alert Bannister also noted a typographical error on the first line of Mr. Toy's routine shown on page 2 of Issue 04: The code for LLI should be 066 not 006 as printed.

Thanks for the use of your sharp eyes - and our apologies to our readers for letting those errors get by. — Ed.

STRINGS PATCH

Mr. H. J. Lewis of Canada has spotted a glitch in the Strings Supplement. The following patch, (named in his honor!) should be installed at Page 50 Location 327:

JFZ HJLFIX

It will replace the JFZ SSTRCL instruction. The patch, which may be placed on Page 54 at Location 301, is just two instructions:

```
HJLFIX,  CAL SWITCH
         JMP SSTRCL
```

This patch will correct an anomaly in the string comparison routines that can effect string comparison operations.

Many thanks to Mr. Lewis for his persistence in analyzing and solving this problem and bringing it to our attention! —Ed.

MATHEMATICAL FUNCTIONS THE OTHER WAY!

One of your fellow SCELBAL users, Robert Leonard, 3003 Driscoll Drive, San Diego, CA. 92117, sent in a nice set of subroutines to calculate the sine, cosine, tangent, arc tangent, log and exponent. The LOG and EXP functions he provided are natural base. The trig functions expect the angles to be given in radians. The variable names assigned and line numbers of the various routines he provides are summarized as follows:

SIN(X) = SN	GOSUB 10
COS(X) = CS	GOSUB 20
TAN(X) = TN	GOSUB 30
ATN(X) = AT	GOSUB 40
LOG(X) = LG	GOSUB 80
EXP(X) = EX	GOSUB 100

The subroutines making up the high level package are shown alongside this column.

Robert also mentioned that he likes to use a patch to eliminate the decimal point and zero after whole numbers. Says he likes the format for its neatness in games, etc. If you want to take a look at it, the patch he uses is presented here:

```
025 147      JMP PATCH

PATCH,       LLI 166
             LAM
             NDI 370
             RTZ
             LAI 256
             CAL ECHO
             JMP NODECP
```

Thanks for the very nice high level math package Robert. Hope you enjoy the check we have sent you for your efforts! - Ed.

LISTING OF HIGH LEVEL MATHEMATICAL FUNCTIONS

```
10 Z=X
11 SN=X
12 N=2
13 Z=-Z*(X↑2)/(N*(N+1))
14 SN=SN+Z
15 N=N+2
16 IF ABS(Z)>.0001 THEN 13
17 RETURN
20 Z=1
21 CS=1
22 N=1
23 Z=-Z*(X↑2)/(N*(N+1))
24 CS=CS+Z
25 N=N+2
26 IF ABS(Z)>.0001 THEN 23
27 RETURN
30 GOSUB 10
31 GOSUB 20
32 TN=SN/CS
33 RETURN
40 IF X<.7 THEN 60
41 IF X>1.4 THEN 70
42 Y=X/SQR(1+(X↑2))
43 Z=Y
44 AT=Y
45 N=1
46 Z=Z*(Y↑2)*(N↑2)/((N+1)*(N+2))
47 AT=AT+Z
48 N=N+2
49 IF ABS(Z)>.000001 THEN 46
50 RETURN
60 Z=X
61 AT=X
62 N=3
63 Z=SGN(Z)*(-(X↑N)/N)
64 AT=AT+Z
65 N=N+2
66 IF ABS(Z)>.000001 THEN 63
67 RETURN
70 Z=1.570796
71 AT=Z
72 N=1
73 Z=SGN(Z)*(-1/(N*(X↑N)))
74 AT=AT+Z
75 N=N+2
76 IF ABS(Z)>.000001 THEN 73
77 RETURN
80 Y=0
81 IF X<1 THEN 85
82 X=X/2
83 Y=Y+1
84 GOTO 81
85 IF X>.5 THEN 89
86 X=2*X
87 Y=Y-1
88 GOTO 85
89 X=(X-.707107)/(X+.707107)
90 LG=2*((X)+((X↑3)/3)+((X↑5)/5)+((X↑7)/
7))-.346573
91 LG=LG+(Y*.693147)
92 RETURN
100 Z=1
101 EX=1
102 N=1
103 Z=Z*X/N
104 EX=EX+Z
105 N=N+1
106 IF ABS(Z)>.000001 THEN 103
107 RETURN
```

What is the VALUE of VAL?

String functions are designed to allow the user to manipulate "strings" of alphanumeric characters *instead of mathematical quantities.*

However, there may be times when it is desirable to manipulate information in essentially two forms - as a string of characters, and as a *numerical value.*

Suppose, for instance, one wanted to have the computer make a list of groceries showing the price for each item, and then also *mathematically* sum the prices to obtain a total?

TOMATOES	24
LETTUCE	79
CARROTS	38
ORANGES	98

One could use string capabilities to list the items and their prices. But the character strings themselves are useless for calculating mathematical information *unless* one has the special capability to convert between one mode and the other. *That is what the VAL function in the SCELBAL String Supplement provides!*

The VAL function *converts characters in a string from an ASCII representation of a decimal number to its numeric value. In other words, the prices in the example can be converted from character string format to actual numeric values that can be mathematically manipulated by SCELBAL!*

Assume the lines in the above example are each composed of two strings 'A$' (item) and 'B$' (price). The 'price' strings in the example would be elements in string arrays B$(1) through B$(4). One could obtain a numerical value for the total of all the prices in the list with a routine such as:

```
FOR X = 1 TO 4
LET T = VAL(B$(X)) + T
NEXT X
PRINT T
```

This is because the VAL function would convert the numerical character strings *to mathematical VALUES!*

If reader interest warrants, we will discuss capabilities of the String Supplement for SCELBAL some more in the next issue of this publication

SCELBAL UPDATE

ISSUE 06 · 3/78

SCELBAL—II READY FOR RELEASE

For sometime there has been a question as to whether or not SCELBAL-II would ever be released in source format. In appreciation of our early customers, a compromise has been reached. As detailed in a separate flyer that will accompany this edition of SCELBAL UPDATE, the revised edition developed specifically for 8080/Z-80 systems will be made available to registered SCELBAL owners for a modest fee as an *uncommented* assembled source listing. Since SCELBAL-II essentially follows the general structure of the original version, SCELBAL owners with 8080 or Z-80 systems should find the improved version attractive and understandable. Those not having the original SCELBAL documentation would likely find it somewhat discouraging to attempt to decipher the uncommented listing of SCELBAL-II. In any event, SCELBAL-II will only be made available to purchasers of the original SCELBAL documentation.

THIS TO BE LAST ISSUE OF SCELBAL UPDATE

As we indicated when we began publication of this journal, the objectives of this supplementary publication were multiple-purpose. First, it would provide a vehicle for informing SCELBAL customers of program corrections that were liable to be required in a program the size and scope of an interpreter. Second, it would be an experimental publication to determine if users wanted to work through the publication to amplify the package in any way. We said we would provide this publication, free for a limited period of time, and possibly on a subscription basis thereafter, if users showed this is what they wanted.

Well, the free period is over, and support for such a publication on a subscription basis has not been demonstrated. Only a handful of readers have submitted material for publication even though an honorarium is presented for published material. Only a fraction of a percent or readers have expressed any interest in having this publication continue on a subscription basis.

The journal has lived up to its task of informing SCELBAL users of program bugs discovered by users over a more than sufficient time span. SCELBAL, with minor alterations pointed out in this journal, is a proven interpretive language.

Best wishes to all its users!

BOWLING HANDICAPPER IN ONLY 512 BYTES!

Harold F. Bower has been running SCELBAL in an eight K 8008 system for some time so he had a limited 512 bytes of user program storage room. That didn't stop him though. He sent in the following program that has been helping him calculate information used by bowling leagues

```
10  INPUT A                          Input total games to date
20  PRINT "INPUT SCORES";
30  INPUT B,C,D                      Input scratch scores
40  PRINT "SCR TOT";
50  INPUT F                          Input previous scratch total
60  PRINT "HDCP TOT";
70  INPUT G
80  PRINT "TOT";
90  INPUT H                          Input previous total pins -
                                     keeping this list eases problems
                                     with changing players in singles
                                     leagues
100 PRINT "HDCP";                    Input player's previous
110 INPUT I                          handicap
115 PRINT
120 PRINT B+C+D;TAB(12);3*I;TAB(24);3*I+B+C+D
130 PRINT "-----";TAB(12);"-----";TAB(24);"-----"
140 PRINT F+B+C+D;TAB(12);G+3*I;TAB(24);H+B+C+D+3*I
                                     The above three lines give
                                     formatted output of scratch
                                     total, handicap total, and
                                     cumulative total suitable for
                                     a 32 column TV display
150 PRINT (F+B+C+D)/A;TAB(12);.66667*(190 - (F+B+C+D)/A)
                                     The above line prints the new
                                     average and handicap
160 GOTO 20                          If next player has bowled the
                                     same number of games change
                                     this to go to line 10
170 END
```

Harold says that while the above program requires quite a few more manual entries than would be required if master files were maintained in string variable format, and could be saved then later loaded and modified with the new results being saved for the next time, the program does save a considerable amount of work and can be run on a minimal system.

Howard is stationed in Germany at HQ 5th SIG CMD, DCSOPS—TD, APO New York, NY 09056. He has recently upgraded his system to a 12K Z-80 so he should really be cranking out handicaps by this time!

MORE FOR BAUDOT MACHINE USERS

Mr. S. J. Toy, a frequent contributor to this publication, still runs a SCELBI 8008 system with a baudot teletype machine for basic I/O. He recently sent in some more information on his modifications of SCELBAL to facilitate its use with a baudot device.

"A while back I described some modifications I made to the INPUT portion of SCELBAL. [See Update Issue 02 — Ed.] Since that time I have discovered that it would not work with the CHR function, mainly because the latter follows a different route through SCELBAL. To overcome this I have made several changes that now make INPUT even more useful.

To allow more than one item of data to be input on the same line, the CR key obviously cannot be used to terminate the entry. Instead, I use another key, which in my case is the Blank key on my model 15 TTY. The STRINF routine is rearranged so that CRLF is skipped when the blank key is used. My previous changes on page 017 that substitute a semicolon for the comma have been removed, and all routines there are restored to their original form. While this allows more than one input per line on the TTY, it also requires that the end of the line be terminated by a following PRINT statement. This seems to be a good tradeoff. The CR key can be used at the end of the line but it is probably better to use a PRINT statement, which makes the carriage return automatic. My modifications to INPUT now consist only of the following:

```
003 046  ***                          Code for Blank key which replaces code for Control/C.

003 050  105 003                      Address in re-arranged STRINF routine to skip CRLF op.

003 102  106 141 003  STRINF,  CAL CRLF
003 105  312                   LBC
003 106  106 113 003           CAL SUBHL
003 111  372                   LMC
```

If one wishes to retain Control/C the test for Line Feed can be sacrificed instead, since LF is not normally used during input of data.

To input data into the same line as data being printed out from memory under TAB control, it is necessary to increment the COLUMN COUNTER each time a digit is input. This is accomplished by inserting a column counter incrementing routine into CINPUT, which is provided by the user for his own particular input device.....By adding a test for the Blank key and the Delete key, which are both non-printing, the column counter incrementing routine can be skipped. If this is not done, the position of the column will be displaced by one character, although this can be compensated for by changing the TAB value. Skipping the column counter incrementer, however, is better, as it simplifies programming. The complete routine to be inserted into CINPUT.....that I use.....is as follows:

```
074 ——      CPI  ——      Test for Blank key.
150 —— ——   JTZ  —— ——   Skip col cnt increment if Blank.
074 ——      CPI  ——      Test for Delete key.
150 —— ——   JTZ  —— ——   Skip col cnt increment if Delete.
066 043     LLI  043     Point to Column Counter.
056 001     LHI  001       "  "    "      "
317         LBM          Load column cntr into B.
010         INB          Increment column counter.
371         LMB          Restore column cntr to memory.
```

The code for the Blank key or the Delete key is in the accumulator when the routine is entered. If either JTZ is true, the jump is to the byte immediately following the end of the routine, which effectively bypasses the column counter incrementer. Incidently, the Delete key, in my case is the BELL key of the model 15 TTY.....

One needs to be careful that registers B, H, and L are free when the routine is used. Locating the routine here covers both numerical and CHR inputs. This addition is useful only if the preceeding modification to INPUT is made.

Another improvement I have made to SCELBAL is to add a function to limit the number of digits printed out. This has been a problem in printing tables of data where either allowance must be made for printing out the full 7 digits or accept an occasional overlap between columns. The INTEGER function does not seem to work for numbers with more than 4 digits [a result of binary rounding operations that start to show their affect when numbers exceed 4 digits — Ed.], and in any case can be used only with whole numbers. Even a number-rounding routine does not always work because the last stage of division frequently results in the value extending back out to 7 digits.

My new function changes the value at location 025 035 which specifies the number of digits to be printed. It replaces the SGN function, which I have never used, and occupies the same space with one byte left over. The Function Names Table is also changed to DIG. The subscript of DIG is the number of digits to be printed. A user program statement would take the form of:

100 PRINT DIG(3)

This will limit all values to three significant digits, until a subsequent statement changes the limit. Besides the 3 digits, allowance must be made, of course, for a possible minus sign and a decimal point. A listing for the Digits Function follows:"

```
007 360  106 000 020  DIGX,  CAL FPFIX     Cvrt FP to fixed.
007 363  066 124             LLI 124       Point to LSW.
007 365  307                 LAM           Load to Acc.
007 366  066 035             LLI 035       Point to digits
007 370  056 025             LHI 025       Number storage.
007 372  370                 LMA           Load new nmbr.
007 373  104 010 010         JMP 010 010   Jump to suppress
                                           printout of nmbr
                                           and to return.

026 305  304                 304           ASCII "D"
026 306  311                 311           ASCII "I"
026 307  307                 307           ASCII "G"
```

[Thanks for all the new information. We have had a number of people ask about a modification to restrict the number of digits outputted. Your's looks like a real straightforward technique to use! — Ed.]

TEXT CONTROL & TRACE CAPABILITY SUBMITTED BY SCELBAL USER

Robert Pearce of 504 McCoys Fork Rd, Walton, KY 41094, says he is not a technical writer but he took the time to send in some pretty clear explanations of how he added some "extra" capabilities to SCELBAL. We think his additions will be of interest to many SCELBAL users.

The first improvement he discusses is a modification to the TEXTC routine that he has named TEXTCM. The modification provides the user with the capability of halting a listing of a program at any time by depressing any character on the input keyboard (except C/R or CTRL/C). Doing so places the program in an "input loop" effectively halting operations while the user inspects the system's display. To continue the display the user may type

a C/R (carriage return). Or, to end the listing and return to the EXECutive routine the user can enter CTRL/C.

Naturally, this capability will be super for those using a CRT display who need capability for displaying a section of the user program buffer at a time. And, it is valuable for any user in that it allows the termination of a long listing when a point of interest has been reached.

The second improvement he presents provides program trace capability. It requires the insertion of a patch at the routine labeled SYNTX3. When trace is activated SCELBAL will display the line number of each line executed in a user's program. Trace capability is controlled using a switch activated via a UDF function.

Robert notes that coupling the trace capability with the TEXTCM modification provides a powerful debugging combination.

He also mentions that his version of SCELBAL has been implemented in a MIKE-2 system.

A commented source listing of the modifications required to implement his improvements is shown below.

```
TEXTCM,  LCM          Fetch (cc) from the first location in
         LAM          The buffer (H&L pnting there)
         NDA          Into Reg C & A. Test the (cc) value.
         RTZ          No display if (cc) is zero.
TEXTCL,  CAL ADV      Advance pointer to next location.
         LAM          Get character from buffer.
         CAL ECHO     Display character.
         IN *         Get input from keyboard.
         CPI 000      Test for 0.
         JTZ END      If yes, continue with TEXTC rtn.
INLOOP,  CAL INPUT    (User subrtn without echo) stop here.
         CPI c/r      And wait for a C/R or a CTL/C.
         JTZ END      If get C/R, continue with display.
         CPI ctl/c    If get CTL/C exit to
         JTZ EXEC     Start over.
         JMP INLOOP   Else cycle.
END,     DCC          Decrement (cc).
         JFZ TEXTCL   If (cc) is not zero continue display.
         RET          Exit to calling routine.
```

[AT PAGE 02 LOCATION 061 CHANGE:]

```
SYNTX3,  CAL TRACE    Insert TRACE patch call.
```

[AT A SUITABLE PATCH AREA ADD:]

```
TRACE,   LLI 201      Replace SYNTX3 instructions.
         LBM
SWITCH,  RET/NOP      RET = NO Trace, NOP = Trace
         .            (Editors note: be careful here, the
         .            label SWITCH has been used else-
         .            where in SCELBAL!)
         LLI 340      Point to line number buffer.
         CAL TEXTC    Display line number.
         LAI 001      Set up number of blanks.
         CAL TABC     Display blank.
         LLI 201      Replace SYNTX3 instructions.
         LBM
         RET          Return to SYNTX3.
```

[AT PAGE 07 LOCATION 074 SET UP:]

```
         JTZ UDF(*)   Jump to UDF function.
```

[AT A SUITABLE PATCH AREA ADD:]

```
UDF(*),  LLI 126      Point to MSB of FPACC.
         LHI 001
         LAM          Get MSB.
         CPI 100      Compare for a FPFIX "1."
         LLI ***      Address of SWITCH point
         LHI ***      For TRACE switch.
         JTZ TRAC     If comparison = 0 move a NOP
         LMI 007      To the switch, else move a RET
         RET          to the switch. Then exit.
TRAC,    LMI 300      Set up a NOP for the switch.
         RET          Exit.
```

ONE MORE TIME

In SCELBAL UPDATE Issue 04 of 1/77 on page 03 Mr. James Tucker of 3 Grove Street, Exeter, NH 03833 discussed a problem with storage of the first variable in the variables symbol table. He recently wrote to notify us of a related problem and a proposed correction:

"The program as it now functions skips the first storage cell when the first variable encountered is a "FOR—NEXT" variable. The search for this variable counts through the variables symbol table and gets part way through the page *(on which the variables are stored — Ed.)* again before finally finding the variable it seeks in the FOR—NEXT stack."

Mr. Tucker submitted two patches shown here "that look for an empty variables symbol table. If empty, a jump avoids advancing the pointer."

```
Present program:
010 132    106 356 022          CAL SWITCH

Change to:
010 132    104 052 075    **    JMP PATCH  (or suitable loc)

And put in the following patch:
075 052    106 356 022          CAL SWITCH
075 055    307                  LAM
075 056    074 000              CPI 000
075 060    110 135 010          JFZ 010 135  (return)
075 063    106 356 022          CAL SWITCH
075 066    104 201 010          JMP STOSY3A


Present program:
005 065    106 356 022          CAL SWITCH

Change to:
005 065    104 017 075          JMP PATCH  (or suitable loc)

And put in the following patch:
075 017    106 356 022          CAL SWITCH
075 022    307                  LAM
075 023    074 000              CPI 000
075 025    110 070 005          JFZ 005 070  (return)
075 030    106 356 022          CAL SWITCH
075 033    104 134 005          JMP LOOKU2A
```