



# INTERACTION

A  
NEWSLETTER  
FOR  
INTERACT  
OWNERS

JAN. - MARCH, 1982

Vol III, no. 1

## DOCUMENTATION ISSUE

This issue of INTERACTION addresses the complaints of many subscribers about the poor or total lack of Interact documentation. While Walt Hendrickson's and Harry Holloway's articles may seem to be overlapping duplications, they are different interpretations of the same material. Both state that while the source code and mnemonics are accurate their comments are just that, not guaranteed truths. If your own research shows errors or differences contact the authors to see how the differences can be resolved and then write me with the results. I am not knowledgeable enough to weigh differences. I can barely read and understand machine code.

Also in this issue is an introductory article on FORTH for the Interact. Russell Schnapp has done an exceptional job implementing F.I.G.-FORTH but he cannot emphasize enough that his documentation is not a FORTH tutorial. You must acquire a text on FORTH first, if you expect to use FORTH. The book he recommends (Using FORTH) has been republished as STARTING FORTH by Leo Brodie. It is now available from Prentice-Hall, Inc. Englewood Cliffs, NJ 07632 Attn: Robert Jordan, Dept. J-974 for \$19.95 clothbound or \$15.95 paperbound, shipped postpaid if payment accompanies order. The FORTH Interest Group also sells this book and other publications on FORTH. I will try to support FORTH in the newsletter by printing routines (FORTH words) and information as I and others learn. I'm still trying to learn to use the editor and write a printer driver.

## BETTER LATE THAN NEVER!

At last the first 1982 issue is out. I'm sorry my personal life has interfered with the timely delivery of the newsletter and I again like to thank everyone for their patience. There still will be six issues this year though until I catch up I can't give you a deadline date for advertising in a specific issue. As for articles, one problem now is what to print first. Over the past months numerous items have arrived and await publication. I still welcome any articles or programs and will try to print as many as possible in an organized sequence.

## THE INNARDS OF BASIC

### PART 1 - THE USE OF LOW RAM.

Harry Holloway, Box 2263, Ann Arbor, MI 48106.

This article is the first in a series about Level II BASIC. The aim is to provide the machine language programmer with a guide that may be used for further exploration and for modification of Level II. I don't have all of the answers so features that I'm not certain of will be marked with a parenthetic question mark. If you find this doubled, i.e. (??), it means that I really don't have a clue. The material will apply to the 8K Graphics BASIC as well as Level II, but not to RS232 BASIC. Note that all numbers will be given in hexadecimal unless otherwise indicated by a trailing 'd'.

While little has been written about Microsoft's 8080 BASIC for the Interact, there is a substantial literature for the TRS80's Z80 version and the Apple's 6502 version. In choosing names for routines and pointers I have mostly followed Apple nomenclature because, here, the consistency between articles leads me to suspect that many authors have had access to a Microsoft source listing.

Level II uses low memory (49A0-5FRE) for four purposes: 1. As a housekeeping area for the BASIC interpreter. 2. For stack operations. 3. For the user's program. 4. For storage of variables that are generated by the user's program. The following tabulation of these uses is in ascending order of addresses.

- 49A0-49FF. Unused. This is nominally part of the memory that is used for the display and most video banners do indeed load to 4000 through 49FE or 49FF. However, the last 96d locations do not show on the screen and they are unaffected by BASIC's scroll routine. This is a convenient hole for a machine language subroutine.
- 4A00-4. Color table. Set by COLOR which then calls SELCOL (0636). The 5th byte controls the intensity of color 2 and is always set to 00 for full intensity.
- 4A05. (?). Flag used to write long leader before the first 256d-byte section of a CSAVE\* tape file.
- 4A06-8. Used by CSAVE\*, CLOAD\* to keep track of the data.
- 4A09-4B08. A 256d-byte buffer used for transfer of data to and from tape by CSAVE\* and CLOAD\*. A temptingly empty space that may be used for machine-language subroutines if data tapes will not be used, but code here will be clobbered by CSAVE\* and CLOAD\*. (An early version of Graphics BASIC had the code for the extended PLOT command here.)
- 4B09-D. Apparently unused.
- 4B0E. The current value of WINDOW. (Default 4D = 77d.)
- 4B0F. A flag that permits PEEK and POKE when set to 19. (Our old friend POKE19215,25)
- 4B10-4BFF. Mostly unused. Stack operations use space from 4BFF down. Graphics BASIC has code for the extended PLOT command at 4B12-4B52.
- 4C00-5. Start code. LXI SP,4C00 followed by JMP 6000.
- 4C06. The screen line on which the next character will be printed.
- 4C07. The screen column on which the next character will be printed.
- 4C08-C. A buffer for the five-byte header record that will be written to tape before a block of data.

The Innards of Basic, cont.

- 4C0D-4C89. The bytes in this section are transferred from an image of the storage area at 64D9-6555 when BASIC initialises. This was done for a ROM version to set up a storage area with embedded data. Nobody bothered to change Interact BASIC, which still thinks that it is in ROM.
- 4C0D-F. (??). The instruction JMP 6298. This is a jump into a stack initialisation routine. Purpose unknown
- 4C10-2. The instruction JMP <address>. BASIC comes here for the USR function and finds the address that has been poked into 19473-4d. Initially the address is set up to give a syntax error message.
- 4C13-5. The instructions OUT 00 and RET. A relic of an earlier version. (The Interact does not have ports that can be addressed with IN and OUT instructions.)
- 4C16-4C23. A set of skeleton instructions into which numbers are embedded for fast division.
- 4C24-4C4A. (?). Used for random number generation.
- 4C4B-D. Another fossil. IN 00 and RET.
- 4C4E. One of a number of partly functional relics of printer routines. This is one more than the number of nulls to print after crlf.
- 4C4F. Set to zero if the last printer operation was lf.
- 4C50. The position of the printer head.
- 4C51. A flag to indicate choice of screen or printer output.
- 4C52. Line length. Originally set up for 48 (72d) characters then changed during initialisation to 11 (17d) characters for the Interact screen.
- 4C53. The last print position on a line for a comma field. After this a comma field gets printed on the next line. Originally set to 38 (56d), but changed during initialisation to 10 (16d).
- 4C54. Flag set to 01 while erasing characters on screen. Set to 00 otherwise.
- 4C55. Flag set to nonzero to suppress output. Switched between zero and nonzero values by control-0.
- 4C56-7. STKTOP. A pointer to the top of the stack. Originally set to 4C00. Later set to the bottom of string storage at 5F8C. Moved up or down by change in size of string storage with CLEAR and argument.
- 4C5A-b. TXTTAB. A pointer to the lowest memory location that is occupied by the user's program. Stays constant at 4D22. BASIC program tapes reload 4D22 into this pointer.
- 4C5C-4C86. The tape output list (TOL) that is used by CSAVE. The first byte of each 7-byte entry is a code that specifies the function of the entry. The codes are:
- 00 - Specifies the writing of a long leader. The remaining six bytes are ignored.
  - FF - Specifies the writing of data from memory to tape. The format is FF, 2-byte load address, 2-byte byte count, 2-byte source address. In all entries that are used by BASIC the source address is the same as the load address.
  - FE - Specifies that a single byte will be used to fill a section of memory. ( Mostly used to clear the screen before loading a video banner.) The format is FF, 2-byte load address, 2-byte byte count, fill byte, unused byte. This feature is supported by the BASIC tape routines, but it is not used by the BASIC interpreter.
  - FD - Specifies the end of the tape file.
- All of these entries except the first generate a 5-byte tape header record that is written to tape via the buffer at 4C08. The general format for a header record is

## The Innards of Basic, cont.

- 2-byte load address, 2-byte byte count, code byte  
(If the code byte is FD the remaining four bytes are ignored.)  
For more details of Interact tape formats see the article in Interaction vol. II, no. 5. Most of the TOL entries may be diverted to other uses by POKEing different values into them. For example, to save a machine language subroutine along with the BASIC program or to make a backup copy of BASIC itself ( see Interaction vol. II, no. 5).
- 4C5C-4C62. TOL entry for a long leader. May be omitted if the user manually advances the cassette beyond the plastic physical leader.
- 4C63-9. TOL entry that loads TXTTAB (4C5A-B). The value loaded is 4D22 which changes nothing.
- 4C6A-4C70. TOL entry that loads VARTAB (4CFD-E). The value loaded depends on the length of the user's program
- 4C71-8. TOL entry for the 5-character filename that is written from and loaded to a buffer at 4D10. This block is always written to tape, even if no filename is specified.
- 4C79-F. TOL entry for a single byte that specifies the number of characters (0-5) that are to be considered in the filename. This loads at 4D1D.
- 4C80-6. TOL for the user's BASIC program that loads at 4D22.
- 4C87. TOL for the end record. (Only the first byte is significant.)
- 4C88-4CCF. Buffer for a command or an input program line.
- 4CD0. (??). Flag for printer routines.
- 4CD1. (??).
- 4CD2. TTYPOS. Keeps track of the printer position.
- 4CD3. (?). Used in array search.
- 4CD4. VALTYP. A flag for variable type set to 00 for numeric and to 01 for string.
- 4CD5. DORES. Flag, when set to zero allows replacement of reserved words with tokens. (crunching)
- 4CD6-7. MEMSIZ. Points to the highest available memory location. Set to 5FBE.
- 4CD8-B. Storage for string operations.
- 4CDC-4CE9. (??). Some kind of buffer for string operations extends downwards from 4CE6.
- 4CEA-B. FRETOP. The top of the unused string storage. Moves downwards as strings are added from the top.
- 4CEC-D. Storage for character pointer.
- 4CEE-F. Pointer used during FOR/NEXT loop.
- 4CF0-1. DATLIN. Stores the number of the program line from which data are being read.
- 4CF2. SUBFLG. Flag that permits subscripted variables when set to zero.
- 4CF3. Flag used during execution of direct statement from the buffer.
- 4CF4. Flag for origin of data. Zero if from read, nonzero if from input statement.
- 4CF4-6. (??). Pointer to program material.
- 4CF7-8. Storage for text pointer.
- 4CF9-A. OLDLIN. Stores number of last program line executed. Saved for restart with CONT after control-C.
- 4CFB-C. OLDTXT. Points to statement to be executed next.
- 4FFD-E. VARTAB. Points to beginning of variable storage (just after the double null at the end of the user's program). Set by NEW to 4D24 then moves up as program is added.
- 4CFF-4D00. ARYTAB. Points to the beginning of array storage. Set by NEW to 4D24. Moves beyond VARTAB as array variables are added.
- 4D01-2. STREND. Points to the end of storage for the names and locations

The Innards of Basic, cont.

of string variables. Set by NEW to 4D24 and moves upward as new variables are added.

- 4D03-4. DATPTR. Points to the memory location from which data are being read. Set by RUN and RESTORE to 4D21 (just before the start of the user's program).
- 4D05-6. PRMNAM. The name of the active parameter.
- 4D07-A. PRMVAL. Four-byte value of parameter for user-defined function.
- 4D07. PRMVAL low byte of mantissa.
- 4D08. PRMVAL middle byte of mantissa.
- 4D09. PRMVAL high byte of mantissa.
- 4D0A. PRMVAL sign and exponent.
- 4D0B-F. The floating point accumulator (FAC). A storage area for a numerical argument or result. (More about floating point numbers in a later article.)
- 4D0B. FAC low byte of mantissa.
- 4D0C. FAC middle byte of mantissa.
- 4D0D. FAC high byte of mantissa.
- 4D0E. FAC sign and exponent.
- 4D0F. FAC sign when unpacked.
- 4D10-4. Buffer for transfer of filename to or from tape.
- 4D15-9. Buffer for filename to be compared with that on tape.
- 4D1A-C. Apparently unused.
- 4D1D. Storage for the number of characters in a filename.
- 4D1E-4D20. Storage used by floating-point multiply.
- 4D21. TSTACK. The location just before the user's program. Needs to contain the byte 00.
- 4D22. The start of storage for the user's BASIC program. Each program line --- has the the format  
 2-byte link address, 2-byte line number, program material, 00  
 The zero byte is used as an end marker for the line and the link address is used to point to storage for the next program line. The end of the program is marked by a double null where a link address would be expected. The program material is a mixture of ASCII characters that have bit 7 clear (i.e. 00-7F) and tokens that represent reserved words. The tokens have bit 7 set (i.e. 80-FF). More about tokens in a later article.

The space above the user's program is used to store variables including arrays and strings. This subject has already been covered in an excellent article by Edward Berne (Interaction, vol I, no 6) to which the reader is referred.

Next time, reserved words, dispatch addresses, and how to hijack LET and DUMMY for your own BASIC overlays.



The SABRE Port

An RS232 level serial printer port for the Interact. Price includes all hardware, operating and installation instructions, port driver program listing, and a BASIC Overlay program which overlays both Level II and 8K Graphics Basic. All for just \$24.99 plus \$2.00 for shipping and handling from:

SABRE, 1415 Crock Hollow Dr., Seabrook TX. 77586 (713)870-8315

## INTERACT LEVEL II BASIC MAP

1

List Compiled by Walt Hendrickson, 2313 W 181st ST., Torrance, CA. 90504. Date: 11/10/81.

The following listing shows the Starting Address and a short description for Most of the subroutines contained in LEVEL II INTERACT BASIC. There is a wealth of useful subroutines in Level II Basic which can be accessed from both Basic and Machine language programs. If you have further questions regarding this list, please write me at the above address.

NOTES: ACC = Floating point Accumulator.

ADDRESS(HEX)	FUNCTIONAL DESCRIPTION								
4C00	F145b sets the stack pointer, then JUMPS to 6000H.								
6000	2457c Initializes the number of lines counter, clears the screen, sets the default colors, then JUMPS to start at 6227H.								
6025	24513 Keyin subroutine. Reads a keyboard input, checks for A-Z, and sets some internal flags. Returns with ASCII key code in Register A.								
605A	2451c Print the character in register C on the screen. Also handle the 'bell' code(07) and back-space functions. Uses only A,C.								
60EC	24312 Routine to do a carriage-return and line feed. Uses A,C.								
6112	24357 Clear last line on the screen.								
6160	24923 Default colors for basic. (04,03,00,07)								
6165	24933 Tape routines to write programs out to tape (CSAVE). Entry= BC points to tape list: <table border="1" style="margin-left: 40px;"> <thead> <tr> <th>byte</th> <th>function</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>tape code (00=long leader,FD= end code,FE= fill code, FF= data code.)</td> </tr> <tr> <td>2,3</td> <td>load address (LO,HI)</td> </tr> <tr> <td>4,5</td> <td>length (LO,HI)</td> </tr> </tbody> </table>	byte	function	1	tape code (00=long leader,FD= end code,FE= fill code, FF= data code.)	2,3	load address (LO,HI)	4,5	length (LO,HI)
byte	function								
1	tape code (00=long leader,FD= end code,FE= fill code, FF= data code.)								
2,3	load address (LO,HI)								
4,5	length (LO,HI)								
6219	25113 Writes normal tape leader tone on tape.								
6227	25121 Basic interpreter start. Checks location 6239 to determine a cold or warm start (00 =cold). If cold, then moves the block of data from 64D9H-6556H to 4C01H-4C8AH. This block contains the warm start vector, USR vector(4C10H), and IN and OUT instructions, which are not implemented on the interact.								
62A1	25249 'Bytes Free' message								
62B0	25264 Title message 'INTERACT MICROSOFT BASIC VER 4.7 COPYRIGHT 1978 BY MICROSOFT'								
6305	26349 Command JUMP Table: ('SGN' thru 'POINT')								

7B15,7BD9,7B2B,4C10,724B,77E7,77F1,7279,7D9F,7E7E,79BA  
7DED,7EF3,7EF9,7F5A,7F6F,76FB,74FD,7315,7597,750C,751D,  
773E,7879,0000,752D,755D,7547,77CB

Interact Level II Basic Map, cont.

633F 25407 Basic Token table:

END, FOR, NEXT, DATA, INPUT, DIM, READ, LET, GOTO, RUN, IF,  
 RESTORE, GOSUB, RETURN, REM, STOP, ON, PLOT, SOUND, TONE, DEF,  
 POKE, PRINT, CONT, LIST, COLOR, CLEAR, CLOAD, CSAVE, CLS, REWIND  
 WINDOW, OUTPUT, NEW, TAB(, TO, FN, SPC(, THEN, NOT, STEP, +, -, \*, /  
 †, AND, OR, >, =, <, SGN, INT, ABS, USR, FRE, FIRE, JOY, POS, SQR, RND  
 †, LOG, EXP, COS, SIN, TAN, ATN, PEEK, LEN, STR\$, VAL, ASC, CHR\$  
 POT, INSTR\$, DUMMY, LEFT\$, RIGHT\$, MID\$, POINT.

645A 25690 Command Jump table continued: ('END' thru 'NEW')

69EE, 68F5, 6E4F, 6B97, 6D56, 70A3, 6D85, 6BAE, 6B54, 6B37, 6C26,  
 69C2, 6B43, 6B72, 6B99, 69EC, 6C08, 7741, 779C, 77B1, 7281, 7725,  
 6C4A, 6A1D, 689B, 775E, 6AF1, 762E, 75D0, 773E, 7817, 789D, 7820,  
 66D4, 6D13, xxxx, 72AE, 6D16, xxxx, 7B6C, 6925

649E 25753 Jump table for arithmetic operators (+, -, \*, /, †, AND, OR)

7C87, 78BB, 79F9, 7A5A, 7DAB, 6FEC, 6FFB

64B3 25779 Error Code Table.

NF, SN, RG, OD, FC, OV, OM, UL, RS, DD, /O, ID, TM, OS, LS, ST, CN, UF,  
 MO.

64D9 25817 Start of 7D hex byte table which is moved on start-up to  
 4C0D-4C8A. (See 6227 above)

6554 25940 'Error' message.

655A 25941 'In' message.

655F 25951 'ok' message.

6564 25950 'Break' message.

656A 25952 Used by (FOR)...(NEXT) loops to manipulate the stack pointer  
 so that (NEXT) knows where the last (FOR) statement ends.

658D 25997 Gets variable name from a line of basic- loads it into the  
 variable area.

6593 26003 Moves bottom of program further down in memory to allow add-  
 ing a line of basic.

RC= Destination address

DE= stopping address, stop when HL=DE

HL= End of program address, ie source address

659E 26014 Check to see if there's enough free memory for next operat-  
 ion.

65B6 26033 OM ERROR routine.

65BB 26043 Puts last-used (DATA) line number into 4C58H so that Basic  
 will print "?SN ERROR IN xx", where xx = data line number.

65C1 26049 SN ERROR

65C4 26052 /O ERROR

65C7 26055 NF ERROR

65CA 26058 DD ERROR

65CD 26061 UF ERROR

65D0 26064 OV ERROR

A5D3 26067 TM ERROR

## Interact Level II Basic Map, cont.

65D5 26069 Prints error message. To use this routine, load register E with code 0 thru 24H, then JUMP to 65D5H.

Error	Message	E value
NF	Next Without for	00
SN	Syntax error	02
RG	Return without gosub	04
OD	Out of data	06
FC	Illegal function call	08
OV	Calculation overflow	0A
OM	Out of memory	0C
UL	Undefined line number	0E
BS	Bad subscript	10
DD	Redimensioned array	12
/0	Division by zero	14
ID	Illegal direct mode	16
TM	Type mismatch	18
OS	Out of string space	1A
LS	String too long	1C
ST	String too complex	1E
CN	Can't continue	20
UF	Undefined user function	22
MO	Missing Operand	24

- 660C 26124 Print 'OK' - back to DIRECT COMMAND mode.
- 6619 26137 Same as 660C except doesn't print 'OK'
- 6697 26263 Reset program pointers - rejustify link pointers.
- 669A 26266 Rejustify link pointers. This can be used to RECOVER a Basic program after (NEW) or (CLOAD) if the original program has not been over-written.
- 66B4 26292 Searches for a particular line number in a Basic program.  
 To search : DE = required line number (binary LO,HI).  
 If Found : BC = Starting address of line  
 : HL = Starting address of next line  
 : Carry flag = 1
- 66D4 26324 (NEW) routine.
- 66E0 26388 Part of (NEW) routine. Entering here resets the STRING space and variable area pointers, but does NOT destroy the program
- 671A 26394 Keyboard input. Prints "?" then JUMPS to 67ED.
- 6727 26407 Scans the input buffer and converts lower case to upper case (unless enclosed in quotes)- also converts Basic reserved words into TOKENS.
- 67D6 26532 Resets HL to 4C87 and puts three zeros at end of program.
- 67ED 26605 Inputs a string of characters from the keyboard until a RETURN (CR) is pressed. Uses a buffer from 4C88 to 4CCF. On exit, HL=4C87 and B=number of characters entered.
- 681A 26650 Tests for buffer overflow.
- 682F 26671 Compares HL with DE and sets the flags accordingly.  
 Z flag set when DE = HL  
 C flag set when DE < HL
- 6835 26697 Tests for expected characters such as commas, semicolons,



Interact Level II Basic Map, cont.

6840	26000	Prints the character in A register.
688A	26702	This could be used to implement a (GET) function. Inputs one key to the A register.
689B	26779	(LIST) routine.
68F5	26869	(FOR) routine.
6925	26917	(STEP) routine.
696E	26990	Re-entry point to the Basic interpreter. HL should be pointing to the zero byte at the end of a line.
6992	27026	BASIC interpreter starts here. HL should point to the byte before the start of program.
69B2		Skips over blanks in the basic text. The carry flag is set if the next character is a number.
69C2		(RESTORE) routine.
69DD		Routine to pause listings. Control/S=pause, Cont./C=exit.
69EC		(STOP) routine.
69EE		(END) routine.
69F3		Prints 'BREAK' if control-C pressed in direct mode. Prints 'BREAK IN...#' if in RUN mode.
6A1D		(CONT) routine.
6A30		CLOAD* Jumps here.
6A35		CSAVE* Jumps here.
6A74		Write 4 bytes to tape. Used by CSAVE*.
6A9E		Tests for alpha-character pointed to by HL, and resets carry flag if found.
6AA6		Same as 6A9E below, but first evaluates a basic expression pointed to by HL, then puts result into ACC.
6AB2		DE = INTEGER value of floating point value in ACC.
6AC7		FC ERROR routine
6ACC		DE = Hex of numeric string pointed to by HL.
6AF1		(CLEAR n,N) where n=string space, N=top of basic Ram.
6B37		(RUN) routine.
6B43		(GOSUB) routine.
6B54		(GOTO) routine.
6B6D		UL ERROR routine
6B72		(RETURN) routine.
6B7B		RG ERROR routine.
6B97		(DATA) routine.
6B99		(REM) routine.
6BAE		(LET) routine.
6C01		Moves a variable (4 bytes) from ACC to (HL). On exit, HL points to the delimiter and DE points to the first of the 4 bytes in the variable area.
6C08		(ON) routine.
6C26		(IF) routine.
6C4A		(PRINT) routine.
6CA8		Do a CR-LF unless cursor is in the first column.
6CB5		Do a CR-LF.
6CBF		Delay by sending the number of Null characters in 4C4E.
6CF6		Here for (TAB( ) or (SPC( ) first.
6D13		(TAB( ) routine.
6D16		(SPC( ) routine.
6D32		"REDO FROM START" message.
6D56		(INPUT) routine.
6D85		(READ) routine.
6E1A		"?EXTRA IGNORED" message.
6E2B		Used by (READ).

## Interact Level II Basic Map, cont.

6E4F	(NEXT) routine.
6E9A	Calls 6EB3 to evaluate Basic expressions & checks for a TM ERROR.
6E9D	Checks for a Numeric operation.
6E9E	Checks for a String operation.
6EAF	Tests for left bracket and evaluates the expression. Jumps to SN ERROR if no left bracket.
6EB3	Evaluates Basic expression pointed to by HL.
6F2A	Used by several routines to test for +,.,-,",(NOT),(FN) and evaluates them if found. Also sets points to JUMP table at 6305 to get addresses for (SGN) thru (MID\$).
6F31	NO ERROR routine.
6F62	Evaluate expression then check for right bracket ")", else SN ERROR.
6F7B	Loads ACC with variable. HL should point to variable name.
6F8C	All commands (SGN) to (PEEK) and (LEN) to (MID\$) pass thru here.
6FDA	(USR) routine.
6FFB	(OR) ACC = (SP) OR ACC.
6FFC	(AND) ACC = (SP) AND ACC.
70A3	(DIM) routine.
70AB	This routine searches for a variable (name pointed to by HL) and either 1. Returns its address in DE or 2. Creates the variable if not found.
715A	Make ACC point to "OK".
71AF	DD ERROR routine.
71C0	BS ERROR routine.
724B	(FRE) routine.
7279	(POS) routine.
727C	ACC = Floating point of value in A.
7281	(DEF) routine.
72AE	(FN) routine.
72F6	Checks to see if in RUN mode or INDIRECT mode.
72FF	ID ERROR routine.
7315	(STR\$) routine.
733A	Loads (4CE6) with string length in reg. A and (4CE8,4CE9) with start address of string from DE.
7349	Counts number of characters in a string. On entry: HL = points to start of ASCII string. On exit: BC = string length, HL = address of string delimit
7385	ST ERROR routine.
738B	Prints a string using 6840. HL must point to start of string.
73A4	First checks if there's still free string space. OS ERROR if none. After call, DE points to start of sub-string with res. A = sub-string length.
7406	String space garbage collection routine.
74BB	Sets up registers before going to 74C1.
74C1	Used by string commands to remove sub-strings. On entry: L = sub-string length. BC = start address of sub-string. DE = start address of next string.
74FD	(LEN) routine.

## Interact Level II Basic Map, cont.

751D (CHR\$) routine.  
 752D (LEFT\$) routine.  
 755D (RIGHT\$) routine.  
 7567 (MID\$) routine.  
 7597 (VAL) routine.  
 75B2 Tests for a right bracket ")" after RIGHT\$, LEFT\$, or MID\$.  
 On return, B = sub-string length.  
 75D0 (CSAVE) routine.  
 762E (CLOAD) routine.  
 7682 Repair the link pointers in a basic program.  
 76FB (PEEK) routine.  
 7725 (POKE) routine.  
 773E (CLS) routine.  
 775E (COLOR) routine.  
 779C (SOUND) routine.  
 77B1 (TONES) routine.  
 77CB (POINT) routine.  
 77DF (POT) routine.  
 77E7 (FIRE) routine.  
 77F1 (JOY) routine.  
 7817 (KEWIND) routine.  
 7820 (OUTPUT) routine.  
 7879 (INSTR\$) routine.  
 789D (WINDOW) routine.  
 78AE ACC = ACC + 0.5  
 78B1 ACC = (HL) + ACC : addition  
 78B7 ACC = (HL) - ACC : subtraction  
 78BD ACC = BCDE - ACC  
 78C0 ACC = BCDE + ACC  
 7926 Sets ACC = zero.  
 7962 OV ERROR routine.  
 7971 BCDE = - BCDE : negation  
 79BA (LOG) routine.  
 79F2 ACC = ACC \* LOG(2)  
 79F9 ACC = (SP) \* ACC  
 79FB ACC = BCDE \* ACC  
 7A5A ACC = (SP) \* ACC  
 7A5C ACC = BCDE / ACC  
 7AEF ACC = ACC \* 10  
 7B06 Checks if ACC = 0 and sets Z flag if it is.  
 7B15 (SGN) routine.  
 7B1D ACC = Floating point of BCDE. This routine can be used to  
 return values to the USR(x) function.  
     B = Exponent  
     A = MSB incl sign bit  
     D = NSB  
     E = LSB.

7B2B (ABS) routine.  
 7B2F Part of ABS routine, but doesn't test the ACC for zero.  
 7B37 (SP) = ACC. Loads the floating point value from the ACC to  
 the stack.  
 7B44 ACC = (HL). Loads floating point value (4 bytes) from HL to  
 the ACC.  
 7B47 ACC = BCDE  
 7B52 BCDE = ACC

Interact Level II Basic Map, cont.

7B55 BCDE = (HL)  
 7B5E (HL) = ACC  
 7B61 (HL) = (DE). Move 4 bytes from location pointed to by DE to HL location.  
 7B63 (HL) = (DE), B. Move # bytes in B register.  
 7B6C ACC = - ACC ; NOT  
 7B81 Compares ACC with BCDE.  
     If ACC < BCDE then A = FFH (-1).  
     ACC = BCDE then A = 00  
     ACC > BCDE then A = 01

7BD9 (INT) routine.  
 7C0D ACC = Floating point of numeric string pointed to by HL.  
 7CAB Numeric string = ACC.  
 7D75 Compares ACC with 999,999.  
 7D84 Floating point 0.5 (00/00/00/80).  
 7D9F (SQR) routine.  
 7DA8 ACC = (SF) ↑ ACC (power)  
 7DED (EXP) routine.  
 7E2D Table of floating point constants used by EXP - SQR.  
 7E42 FF/FF/7F/7F = 0.5  
     00/00/80/81 = -1  
     00/00/00/81 = +1

7E7E (RND) routine.  
 7EE7 Tables of data used by RND.  
 7EF3 (COS) = SIN(x + PI/2)  
 7EF9 (SIN) = x - x↑3/3! + x↑5/5! - x↑7/7! + x↑9/9!  
 7F3D Table of floating point constants (4 bytes each) used by SIN - COS.  
 7F45 05 = Number of terms required to calc. SIN - COS.  
 7F46 BA/D7/1E/86 = x↑9/9!  
     64/26/99/87 = x↑7/7!  
     58/34/23/87 = x↑5/5!  
     E0/5D/A5/86 = x↑3/3!  
     DA/0F/49/83 = 2PI  
     this is the table used to compute SIN(X).

7F5A (TAN) routine = (SIN) / (COS)  
 7F6F (ATN) routine.  
 7F96 Table of floating point constants used by ATN.  
 7FBB Delay routine. Delays BC counts.

Notes:

1. Floating point numbers occupy 4 bytes each.
2. ACC is the floating point accumulator 4D0B to 4D0E.  
     4D0B = least significant byte  
     4D0C = next byte  
     4D0D = Most significant byte  
     4D0E = Exponent
3. For arithmetic operations requiring 2 operands, one is usually the ACC, and the other is either (HL) or BCDE, where BCDE contains:  
     B = Exponent  
     C = MSB  
     D = NSB  
     E = LSB

4. HL is used as the text pointer for most of the comparison and conversion

Interact Level II Basic Map, cont.

List Compiled by Walt Henorickson, 2313 W 181st ST., Torrance, CA. 90504. Date: 11/10/81.

NOTES: ACC = Floating point Accumulator.

ADDRESS(HEX)	SIZE	FUNCTIONAL DESCRIPTION
4C0D	3	JUMP instruction to Basic Warm Start.
4C10	3	JUMP instruction to the USR( ) function routine. Default address is the FC ERROR routine. The user must put his routine's address here before using the USR( ) function call.
4C13	3	OUT (nn),A routine for the OUT I,J function which is NOT implemented on the INTERACT.
4C16	14	Fast 4 byte subtract routine used by floating point divide for speed.
4C24	35	Pseudo-random number data used as tables and counters by the RND function.
4C47	4	Last pseudo-random number generated by RND, in floating point, for RND(0).
4C4B	3	IN instruction, NOT implemented in the INTERACT.
4C4E	1	Number of ASCII nulls to print after a carriage return. Defaults to 1.
4C52	1	Terminal line length. Defaults to 72.
4C56	2	Pointer to top of Basic stack.
4C58	2	Current line number. OFFFEH at initialization. OFFFFH in direct mode.
4C5A	2	Pointer to start of Basic program text, ie. 4D22.
4CFD	2	Pointer to start of variable space.
4D0B	4	Floating point accumulator.



---- INTERACT PROGRAMS FOR HAM RADIO ----

MACHINE LANGUAGE PROGRAMS FOR MORSE TRANSMISSION AND RECEPTION. INCL. PLANS FOR HOOKUP TO RCVR AND XMR (ELECTRONICS KNOWLEDGE REQUIRED) BOTH PROGRAMS, CASSETTE+INSTRUCTIONS: \$28.50 (LISTING: \$5.00 EXTRA). J.A. MILLER. P.O. BOX 455, MELBOURNE, FLA. 32901

---- ASSEMBLEX-EDITEX PATCH TAPES ----

FOR ASSEMBLEX V4.5, EDITEX V2.5- OVERLAY FIXES BUGS, IMPROVES OPERABILITY. TAPE, LISTING, INSTRUCTIONS FOR BOTH: \$12.50.

## FORTH FOR YOUR INTERACT

by Russ Schnapp 8062 Gold Coast San Diego, CA 92126

There is now a new language available for your 16K or 32K INTERACT: FORTH. FORTH is the highly powerful language described in the August, 1980 issue of BYTE magazine. It can be used for many of the same things as BASIC, but gives you more precise control over your machine. In addition, it is a block-structured language, it is much faster than BASIC, and is much easier to extend and adapt. Included with FORTH for the INTERACT is a graphics package, a simple line-based editor, and even an assembler.

Before getting into any more detail on the specific features of this implementation, let's give a little flavor of FORTH. FORTH is a stack-based language. While most languages are executed using some sort of stack, the stack is very visible in FORTH as a language feature. As an example, the BASIC statement:

```
100 LET A=B*C+D
```

looks like this in FORTH for the INTERACT:

```
B ^ C ^ * D ^ + A !
```

In explanation, naming a variable places the address of its contents onto the top of the stack. The ^ (value) operator pops an address off the top of the stack, and pushes the contents of the 16 bit word at that address back onto the stack. Thus, the FORTH statement, " D ^ " leaves the value of the variable D on top of the stack. The \* (multiply) and + (add) operators pop two words off the stack and push their product or sum, respectively, onto the stack. The ! (store) operator pops an address and then a value off the stack, and stores the value at the word pointed to by the address. The above notation is called Reverse Polish Notation, of Hewlett Packard calculator fame (it is called Polish notation only because no one can spell or say Lucasiewicz, the name of the Polish mathematician who invented it). In FORTH Interest Group standard FORTH, the store operator is @, but the INTERACT keyboard does not have this key.

You may have noticed that there are no sequence numbers on the examples of FORTH statements. While editing a FORTH program, there are line numbers, but they are merely placeholders. This is because, unlike in BASIC, FORTH needs no labels: There are NO GOTO's! This points out another big difference from BASIC. FORTH uses block-style control structures which eliminate the need for labels and GOTO's. For example:

```
100 IF A > B THEN LET C=A : GOTO 300
200 LET C=B
300 ...
```

In the above, "300" is a label. An equivalent FORTH sequence is:

```
A ^ B ^ > IF A ^ ELSE B ^ ENDIF C !
```

There is also a FOR/NEXT equivalent, as well as a few other very useful iteration constructs.

## FORTH for Your Interact, cont.

One of the major advantages of FORTH over BASIC is the somewhat better developed subroutine construct. In BASIC, only "functions" can have parameters, and these functions are limited to being one-liners. BASIC "subroutines" do not support parameters. In FORTH, the equivalent of the subroutine or function is the "word". For example, consider the FORTH word:

```
: MAXIMUM ( 2nd--value of A, 1st-- val B --> 1st--Max of A, B )
  2DUP          ( copy A and B )
  > IF DROP     ( if A>B pop B off the stack )
  ELSE SWAP DROP ( otherwise pop A off the stack )
  ENDIF
```

In the above, comments are delimited by parentheses, ":" begins a FORTH word definition, and ";" terminates it. The above defines a word called MAXIMUM which pops two values off the stack, and pushes the greater of them back onto the stack. If the MAX function wasn't already defined in BASIC, it couldn't be written. There are many other advantages to having parameters. They allow you to modularize a program, and help you with "structured programming" techniques. While we're on that subject, as previously mentioned, FORTH does not have a GOTO equivalent. The IF/ELSE/ENDIF and other block control constructs allow you to do without GOTO's very nicely. In fact, I have been exclusively working with and implementing GOTO-less languages professionally for the last four years, and have never missed the GOTO once. To a great degree, the ability to share code via FORTH words is a much less buggy way than via GOTO's and GOSUB's.

FORTH is very much an interactive language (if you'll pardon the pun). A "program" consists of one or more word definitions. To run a program, you type a FORTH statement which invokes the words you want to run. As an example (in the following, the . (printnum) operator prints out and pops the top stack element value; underlines text is output; (cr) represents a carriage return):

```
1 2 + (cr) OK
. (cr) 3 OK
1 2 3 . . . (cr) 3 2 1 OK
3 2 MAXIMUM . (cr) 3 OK
7 22 5 MAXIMUM MAXIMUM . (cr) 22 OK
```

You can experimentally execute any word you like, giving it any parameters you choose, merely by naming its word. This greatly aids in debugging medium to large programs.

FORTH for Your Interact, cont.

For more information on the FORTH language, see the August 1980 BYTE, or get in touch with the FORTH Interest Group (P.O. Box 1105, San Carlos, CA 94070). Mountain View Press (P.O. Box 4656, Mountain View, CA 94040) also has some good publications on FORTH.

As I mentioned, FORTH for the INTERACT includes a graphics package. This package allows fast plots of individual pixels, rectangles, and vectors (lines). It also has provisions for changing colors and character sets, positioning the cursor, and setting the graphics window. How fast? It was quick and easy (two evening's work) to design, code, key-in, and debug a PONG-like game. I've also adapted a Breakout game from a listing in BYTE, also in around one evening.

There is also an assembler included. This is NOT a generalized symbolic assembler. It is primarily designed to allow you to build a word from assembler code, rather than from other FORTH words. This allows you to selectively speed up critical portions of a program.

FORTH is NOT all roses, of course. It is somewhat more difficult to read than BASIC (although easier than assembly or machine language). The editor is a bit difficult to use, and files are painful to update. FORTH is a compiled/interpreted language, so after changing part of a program, one must usually recompile all of it. For these reasons, I am not particularly fond of the FORTH language (if it is, indeed, a language). It is more powerful, better structured, and faster than BASIC, though. It is, at the same time, easier to read, and almost as fast as machine/assembler language.

FORTH for the INTERACT is available as advertised elsewhere in this issue, along with documentation of the differences from and additions to the F.I.G. standard. I strongly suggest you learn a bit more about FORTH from other sources (such as "Using Forth," available from F.I.G.) before ordering your copy.



QUEST!.. an adventure style game that lets you use the computer as your slave to find trinkets and treasure. YOU could become filthy rich, rich, or just filthy, it depends upon your ingenuity. A real bargain (a steal actually) at \$5.95



## PRODUCT REVIEW

## EDITEX-ASSEMBLEX PRODUCT REVIEW

J.A. Miller P.O. Box 455

Melbourne, Fla. 32901

This is a review of MicroVideo's Assembler and Editor for the Interact. This review applies only to the versions now in my possession (Editex Version 2.5, and Assemblex version 4.5, purchased 1 Dec. 1981). These products represent a positive step toward opening the Interact to the serious programmer. Unfortunately, the indicated versions of both programs contain "bugs", which will be discussed in this review.

It should be noted that Assembly language programming is intended for advanced programmers, or programmers who are highly motivated to learn its rigors and discipline. Those who have "POKED" short "USR" routines for use with Basic programs can appreciate the tedium involved in remembering hexadecimal Instruction codes and calculating addresses. An Assembler reduces the work involved by allowing use of mnemonic instruction codes and symbolic names for addresses or data. An Editor is used to create and manipulate the user's "source" program. The Assembler is then used to translate the source program into a machine-executable "object" program.

Assemblers and Editors in general are not new to the computer world. A number of available books provide detailed tutorials on 8080 Assembler use. To name a few: "8080A-8085 Assembly Language Programming", by L. A. Leventhal, "Z80-8080 Assembly Language Programming", by Kathe Spracklen, and Intel's 8080 Assembly Language programming book.

Editex is not limited to creation and modification of Assembly Language source text. It is actually a general purpose text editor. It can be used to build, edit and save (on cassette) any user-written text file. I have even found an unadvertised capability within Editex to handle both upper and lower case text. This letter, in fact, was composed and printed using Editex. Editex cannot, however, be used to edit Basic source programs, which use token values rather than raw text.

Editex allows for both line and, to some extent, character oriented editing. Each text line is assigned a unique number for user reference. The user may display or change specific lines, insert text between lines, erase lines. Special keyboard control characters allow for nominal editing within a line.

Both Editex and Assemblex operate below 5F00H (but not together!). Each program overwrites the other when loaded. Both process text in the text buffer at 6000H. After an editing session, Assemblex can be loaded and the text already left at 6000H by the Editor can be assembled without reloading it from tape. Unfortunately, the converse is not true. A fault of Editex is that it loses its pointers to the text area when it is reloaded. Thus, the user is advised to always save his text on tape.

Assemblex recognizes and processes all standard 8080 mnemonics. Assemblex also recognizes several "pseudo operators" which allow the programmer to establish base addresses (ORG), reserve storage areas (DS), define data bytes (DB) as constants, to assign values to symbols (SET), as well as other useful functions. The ORG pseudo operator is particularly useful in that the programmer need only define the starting address of a segment of his program. From this base address, the assembler will automatically calculate subsequent addresses.

## Editex-Assemblex product review, cont.

Assemblex performs two scans of the source text in the text buffer. It is called a "two pass" assembler for this reason. The first pass identifies symbolic labels and assigns addresses to the symbols. A symbol table is built in RAM from 5900H up. Any errors detected in pass one are identified with numeric line number and error code displayed on the screen. If pass 1 is successful, the user initiates pass 2 with a keyboard command. Pass 2 will create the "object" file in whatever free RAM there is available above the text buffer (i.e., the RAM left over after all text has been loaded). Herein lies a basic limitation of Assemblex. There is a definite limit to the size of program segments which can be individually assembled. In general, the more text there is in the source code, the smaller the remaining area will be for creation of the object code. The instructions included with Assemblex suggest a procedure for assembling a very large program in segments and linking the individual object files together in RAM with the MicroVideo Monitor (or other suitable monitor). In all fairness, we should realize that this program size constraint is more a limitation of the small 16K Interact than a fault of Assemblex.

After pass 2 finishes, the object program can be written to tape. The data will be written with all necessary tape headers to ensure that the object program will load into the intended RAM locations. The usual Interact "L" command is used to load and execute the object program.

A third (optional) pass is available to provide a display and hardcopy of the assembled program. A list of symbol names and their assigned Hex. values is also provided. Hardcopy is produced via the MicroVideo RS-232 serial port.

A good feature of Assemblex is its ability to create multiple tape records corresponding to multiple ORG Pseudo-Ops in the source program. For example, the user can ORG to 4C00 to assemble the two instruction start code, then ORG to 6000 to assemble the main program. When the object tape is later loaded, only those instructions or data specifically assembled at the various addresses will be written into RAM. This is handy for producing tapes to patch or overlay existing programs. The Assemblex tape list is built from 5E00H and up. Its maximum allowable size is unknown at this writing.

MicroVideo provides adequate instructions (25 pages) on Editex and Assemblex. Strangely enough, when the tapes were received in the mail, there was no documentation at all. A phone call indicated that the instructions were "not available" and would be mailed shortly. They finally arrived two weeks later and we were able to begin using the package.

Now for the problems. Both Editex and Assemblex have what I believe to be serious bugs. First, consider Editex: One of its keyboard commands (the "D" command) allows the user to display a specific line of text by number. For example, a command like D13 will display line 13 on the screen. The line will be displayed preceded by its line number in three digits (013). There is no problem when line numbers are less than 2 digits long. However, when attempting to display line 101, for example, the Editor displays the correct line, but it will incorrectly display the line number as 010, omitting the last digit. The user, if aware of this, can work around it, even though it is a real inconvenience. After disassembling Editex to printer, I was able to locate the faulty code and produce the necessary patches to fix the error.

Editex-Assemblex product review, cont.

Now the major Assemblex error: It fails to properly write program segments longer than OFFH bytes to tape. The error is in its handling of byte counts in its internal tape output list. A disassembly of the Assembler, after much tracing through the multitude of JMPs and patches already there, revealed an error in 16-bit addition done to increment the byte counter.

Another patch was then prepared to fix this flaw. Ironically, the Editor and Assembler were used to create and assemble the overlay tapes which contained their own patches! This at least demonstrates that both are reasonably workable as delivered.

The above represent the major errors in the product. There are other areas which could be "spiffed up" to improve operability. A foremost improvement would be to allow the user to swap between Editex and Assemblex without disturbing existing text. This would speed the Editing and Assembling processes greatly. As another improvement, the screen scroll rate could be increased. The capability to output to printer independently of the screen would speed up the printing process.

I will attempt to develop patch tapes to implement these and other improvements, and should be making these available for a nominal fee in the future.

For those who want to make the major fixes themselves, the assembly listing for the Editex and Assemblex patches is included below. The patches can be keyed in by hand from a high memory monitor (such as HILO) on top of the corresponding locations after loading the Editor or Assembler as the case may be.

In conclusion, Editex and Assemblex, in their current state, are a mixed bag. The excitement of finally having a "real" Assembler for the Interact has been squelched, in my mind at least, because of the existing flaws. It is clear to me that this product was not adequately tested by MicroVideo (at least for programs longer than 99 source lines or FFh object bytes) prior to marketing. These facts should be kept in mind by any potential purchaser. It is hoped that MicroVideo will correct the problems before selling any more of these programs. For the price paid, one would also expect them to provide FREE updates to those who have bought the defective versions. The company's credibility would seem to depend on this.

```

;      EDITEX PATCHES
;
;      FOLLOWING PATCH FIXES ERRORS IN LINE
;      NUMBER MANAGEMENT WITH "D" CMD
      ORG 5235H
      CALL 4DA0H          5235 CD A0 4D
      MVI B,00FFH        5238 06 FF
SCAN: INX D              523A 13
      INR B              523B 04
      LDAX D            523C 1A
      CPI 20H          523D FE 20
      JZ TRM          523F CA 47 52
      CPI 2CH          5242 FE 2C
      JNZ SCAN          5244 C2 3A 52
TRM:  MVI C,04H        5247 0E 04
      DCX D            5249 1B
      LXI H,51D9H      524A 21 09 51
    
```

Editex-Assemblex product review, cont.

COPY: LDAX D	524D 1A
MOV M,A	524E 77
DCX H	524F 2B
DCX D	5250 1B
DCR C	5251 0D
JZ CONT	5252 CA F4 4C
DCR B	5255 05
JNZ COPY	5256 C2 4D 52
JMP CONT	5259 C3 F4 4C
ORG 4CF4H	
CONT: LXI H,51D9H	4CF4 21 D9 51
DCR M	4CF7 35
JMP 4E22H	4CF8 C3 22 4E
ORG 4F3FH	
JZ 4D00H	4F3F CA 00 4D
ORG 529AH	
JZ 4D00H	529A CA 00 4D
ORG 5218H	
JZ 4D00H	5218 CA 00 4D
ORG 508DH	
DW 51EAH	508D EA 51
ORG 51F9H	
JMP 4F04H	51F8 C3 04 4F
ORG 4CCA H	
JMP 5192H	4CCA C3 92 51

END

```

; ASSEMBLEX PATCHES
;
;
; FOLLOWING PATCH FIXES ERROR IN TAPE LIST
; BYTE COUNTS GREATER THAN FFH
  ORG 4C67H
  LHLD 5026H
  MOV A,M ;16 BIT ADD FOR BYTE COUNTS
  ADI 01H ;IN TAPE LIST
  MOV M,A
  INX H
  MVI A,00H
  JMP 543FH
  ORG 543CH
  JMP 5223H
  ADC M
  MOV M,A
  POP PSW
  POP H
  JMP 5223H
;
;
  ORG 5C34H
  JMP 4C7BH
END
  
```

4C67	2A 26 56
4C6A	7E
4C6B	C6 01
4C6D	77
4C6E	23
4C6F	3E 00
4C71	C3 3F 54
543C	C3 23 52
543F	BE
5440	77
5441	F1
5442	E1
5443	C3 23 52
5C34	C3 7B 4C

MICROVIDEO REPLIES

MICRO  VIDEO™

P.O. Box 7357  
204 E. Washington St.  
Ann Arbor, MI 48107  
(313) 996-0626

February 10, 1982

Mr. James A. Miller  
Editex-Assemblex Product Review  
P.O. Box 7357  
Melbourne, Fla. 32901

Dear Jim:

Thank you for sending your review on Editex-Assemblex. Your lengthy overview covers a lot of features of the package rather well and indicated good familiarity with this new offering.

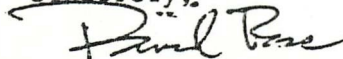
We believe that the Interact owner who developed and then sold these programs to Micro Video did a very commendable job. Many owners took the position that an Assembler "couldn't be done" or simply shyed away from the enormity of the task. He wrote the assembler from scratch on his Interact at home without the benefit of expensive development machines, fancy hardware, or even a printer! That's creativity at its finest.

Unfortunately there are some bugs in this new software that we're now correcting and will replace defective tapes as we have always done in the past.

Contrary to the lukewarm summary of your review, we find most knowledgeable owners to be delighted with the offering and its excellent documentation. Our hope is that offerings like Assemblex-Editex will encourage other owners to develop new games and applications that will appear in future Micro Video catalogues. With Assemblex-Editex the developer now has a middle ground between the tedium of machine language/absolute addressing and the relatively slow execution times of programs written in Microsoft BASIC.

We think that Assemblex-Editex represents a significant new development in the life of the Interact computer which was given up for dead over two years ago. New software, documentation, RS232 and 32K RAM expansions continue to put new life into the machine. We appreciate that you cared enough to take the time to do such a thorough review. We always like hearing from Interact owners as it's your enthusiasm for the machine that has helped all owners to enjoy their Interacts more than they had probably initially anticipated.

Sincerely,



David L. Ross



FORTH FOR THE INTERACT !!

This is an adaptation of F.I.G. standard FORTH for 16K and 32K INTERACT computers. Comes with a graphics package, an assembler and an editor. FORTH is faster and more powerful than BASIC.

Included is documentation of the differences from, and additions to the F.I.G. standard. A FORTH manual is NOT included (these can be obtained from FORTH Interest Group or Mountain View Press).

For your copy of FORTH for the INTERACT, send \$12 to:  
R. Schnapp, 8062 Gold Coast Drive, San Diego, CA 92126

Also available is a super-fast LIFE program. This program uses the left joystick to draw the initial pattern. Price: \$6.

---

```

*****
*
* SKETCH PAD - BASIC program with extensive machine language subroutines for
*   creating, modifying, and saving screen displays. Draws open and filled
*   circles, rectangles, lines, and letters with super-fast joystick posi-
*   tioning. Saves screen on tape with or without stop code (to create pro-
*   gram banners). Hours of fun for all ages.....$8.00
* 8080 DISASSEMBLER in BASIC and QUEST in EDU-BASIC still.....$5.00 each*
*   David J. Schwab      10 Jay Lee Court      Ann Arbor, MI 48104
*
*****

```

---

George A. Leggett's MACHINE LANGUAGE SUBROUTINES

Tape and Documentation for Twenty-One Routines in all: ten of which appeared in the book 8080 FOR EVERYONE, and eleven of which have never before been released by me.

The routines sit between 5C00 and 5F48 and include the following: Multiple Box Data, Multiple Character Data, Diagonal Southeast and Southwest Plotting, Screen Scrolling (Up, down, Left or Right), Typing Keyboard (allows you to type and display print and then store in desired memory), Multiple File Write Tape (allows you to make professional quality tapes with headers, color changes while loading and clearing screen), Sound Generator, (works like BASIC), Sound Off, Time Based Random Generator, Time Based Random Generator with Parameters, Left and Right Isolate for Accumulator A, Keyboard Entry with Tone, Partial Memory Clearing (also used as Partial Screen Clearing), Register B to Two ASCII Digits, Upper Case Key Entry, and Write To Tape.

Send \$10.00 for MACHINE LANGUAGE SUBROUTINES  
Or, send \$22.00 for MACHINE LANGUAGE PROGRAMMING PACKET which includes:  
The book, 8080 FOR EVERYONE, the X-Y PLOTTING MONITOR, and the  
MACHINE LANGUAGE SUBROUTINES.

8080 FOR EVERYONE \$15.00, X-Y PLOTTING MONITOR \$5.00  
George A. Leggett  
52895 Burker Hill Blvd.  
New Baltimore, Mich. 48047



\*\*\*\*\*

Announcing a major utility for the Interact,

**TAPE MASTER**

This is the first effective program that has been specifically designed to make backup copies of program tapes, including those that incorporate anticopy measures. It must be emphasized that release of this program should not be construed as an encouragement of copyright violations. However, the purchaser of software does have specific rights that relate to archival copies. *The purchaser of TAPE MASTER assumes sole responsibility for ensuring that his or her use of the program is in compliance with all applicable laws.*

Functionally, the program has three parts:

**1. TAPE-TO-TAPE COPY ROUTINES.**

There are two of these, one for the older 8K tapes and the other for the new 16K tapes. The procedures are simple and are directed by prompts on the screen. Despite this simplicity, the 16K routine will handle all of the anticopy methods that are known to have been used so far, as well as several that might be used in the future.

**2. TAPE I/O ROUTINES.**

These permit the partial loading of program tapes for study and/or modification. The routines include Load Headers, Load Part (of a specified data block - more powerful than the LP command in the original HILO monitor), Load Bottom (for the lower half of a 16K program), Load Top (for the upper half of a 16K program), and Write from a tape output list. These routines have been made flexible enough that they may be used for investigation and probably for breaking of future anticopy methods.

**3. MACHINE-LANGUAGE MONITOR UTILITIES.**

For the user who wishes to examine and/or change program content (e.g. to remove program bombs, such as tests for the second ROM). The routines include Display memory (as bytes or as ASCII text), Substitute in memory, Fill memory, Move, Disassemble, Assemble (a mini-assembler like that in HILO), Find memory references (three routines for, respectively, addresses in three-byte instructions, addresses in pairs of two-byte instructions, and addresses stored as data), Hexadecimal arithmetic, Hexadecimal/decimal interconversions and a jump to a user-installed routine.

TAPE MASTER is accompanied by comprehensive documentation, including the first available description of Interact tape anticopy methods. The price is \$22.50 postpaid (MI residents add \$0.90 tax). For the serious machine-language programmer who wishes to study the code (there is more than 5K of it), copies of the annotated source listing are available to program purchasers for \$25.00 postpaid (MI residents add \$1.00 tax).

HARRY HOLLOWAY, PO BOX 2263, ANN ARBOR, MI 48106

\*\*\*\*\*

SLAGH SYSTEM SERVICES  
SPRING 1981 PRODUCT LIST

U80M-A/T SERIAL INTERFACE FOR THE INTERACT MODEL ONE. PROVIDES BOTH RS232 I/O AND 20MA CURRENT LOOP OUTPUT. ALSO PROVIDES PARALLEL LATCHED DATA AT BOARD EDGE CONNECTOR. PROTOTYPE AREA ON BOARD. COMES COMPLETELY ASSEMBLED AND TESTED, READY TO INSTALL. COMES WITH DB25P CABLE ATTACHED. FULL DOCUMENTATION. A/T \$70.

U80M-KIT SAME AS ABOVE. INCLUDES FULL STEP-BY-STEP INSTRUCTIONS FOR ASSEMBLY AND INSTALLATION. (LESS CABLE) KIT \$55.

DB25P D TYPE CONNECTOR WITH ATTACHED 9 WIRE CABLE. CABLE \$ 9.

P80I2 OVERLAY PROGRAM FOR LEVEL 2 BASIC WHICH ALLOWS DIRECTED OUTPUT TO U80M PORT, TV SCREEN, OR BOTH. PERMITS INPUT OF ASCII ENCODED INFORMATION THROUGH PORT, FROM EXTERNAL MONITOR KEYBOARD. WRITTEN BY H. HOLLOWAY. CASSETTE \$14.

T80I1 SIMPLE TERMINAL EMULATOR PROGRAM. 300 BAUD. CASSETTE \$ 9.

ICS INTERACT COMMUNICATION SYSTEM, WRITTEN BY G. MEYER. FEATURES MENU SELECTABLE BAUD RATES, WORD SPECIFICATIONS, AND SETTING OF SPECIAL FUNCTION KEYS, AND MORE. RECEIVE AND TRANSMIT INTEL HEX FILES, AS WELL AS BASIC PROGRAMS. FULL SCREEN MEMORY EDITOR, ALLOWS YOU TO INSPECT AND EDIT HEX DATA IN RAM (INSPECTS ANY ADDRESS). FULL DOCUMENTATION INCLUDED. CASSETTE \$14.

INTERACT SERVICE MANUAL UPDATES

NEW DOCUMENTATION PACKAGE BEING COMPILED BY SLAGH SYSTEM SERVICES. INCLUDES COMPLETE REDRAW AND RELABLE OF SCHEMATICS, FUNCTIONAL BLOCK DESCRIPTIONS OF INTERACT ELECTRONICS, MNEMONICS LIST AND WIRE JUMP LIST. DRAWINGS OR PHOTOS OF TEST POINT WAVE FORMS, PARTS LOCATOR CHART, AND SCHEMATIC CORRECTIONS. A VALUABLE TOOL FOR REPAIRING AND UNDERSTANDING YOUR INTERACT, NOW THAT THEY'RE NO LONGER BEING MANUFACTURED AT ALL. INCLUDES REFERENCES TO INTERACT SERVICE MANUAL, WITH ALTERNATE SUGGESTIONS TO "REPLACE MAIN ELECTRONICS BOARD." (NOT VERY PRACTICAL WHEN YOU DON'T HAVE ONE) AVAILABLE BY MAY 1, 1982. UPDATES \$20.

NEW ADDRESS FOR SLAGH SYSTEM SERVICES

SLAGH SYSTEM SERVICES  
BOX 53  
DEARBORN, MI. 48121

NEW TELEPHONE NUMBER: (313) 581-8593  
FEEL FREE TO CALL FOR ASSISTANCE OR QUESTIONS.  
GOOD LUCK CATCHING ME, I KEEP A VERY BUSY SCHEDULE  
NO ! NO ! CALLS AFTER 1 AM, EASTERN TIME.

QUESTIONS ABOUT THE U80

THERE ARE MANY REPETITIVE QUESTIONS THAT ARISE ABOUT BASIC FEATURES OF THE U80M PORT. I HAVE WRITTEN A SHORT ANSWER TO SOME OF THESE QUESTIONS AND WOULD BE GLAD TO SEND A COPY TO ANYONE, FREE OF CHARGE. IT INCLUDES INFORMATION ON USE OF THE PARALLEL CAPABILITY OF THE U80, AND SOME OF THE REASONS WHY WE HAVEN'T OFFERED FURTHER EXPANSION FOR THE INTERACT. SEND AN S.A.S.E. TO;

SLAGH SYSTEM SERVICES  
BOX 53  
DEARBORN, MI. 48121





# INTERACTION

A NEWSLETTER FOR INTERACT OWNERS

---

April - May, 1982

VOL. III, no. 2

---

## Leggett's MACHINE SHOP TALK

by George Leggett 20562 Woodward Mt. Clemens, MI 48043

Throughout the year, I hope to shed a little light on the "unknown world of computers". If you have been using BASIC, and feel that it would be hard to learn Machine Language, you are not giving yourself credit. Not so long ago, BASIC was new to you. And, when you think of it, what is BASIC made up of anyway? Nothing more than Machine Language, the most powerful tool a computer can have. I feel that our Interact is one of the best 8080 systems on the market and with a Monitor we have the power of a very intelligent computer system.

When you are using Machine Language, you are communicating with the computer at its own level (a very low level). Much time and memory is saved since your instructions do not have to be interpreted and converted back and forth between two languages. At the same time, there are many similarities between the two languages. In BASIC, we have countless variables to store and work with information. In Machine Language, we have Registers, which do not take our useful memory and there are only a few to remember, A, B, C, DE, HL. Many functions are similar, but with different terms. Instead of line numbers, we have Addresses of memory locations given in Hexadecimal numbers in all of the Monitors commonly used with the Interact. For GOTO we have JUMP. For GOSUB we have CALL and RETURN, but we can be much more specific about the conditions. Our job is made easier by calling up existing routines in Rom 1, (the Read Only Memory which operates in every Interact). We can use these routines for colors, sounds, plotting and printing messages on the screen.

Here is a small program given in BASIC and Machine Language to draw a line on the screen. You be the judge of which language you would like to use.

```
10 CLS: FOR X + 1 TO 112: PLOT X, 35, 3: NEXT  
20 POKE 19215, 25: POKE 19473, 0: POKE 19474, 94: USR(0)
```

## MACHINE SHOP TALK, cont.

Address	HEX	Mnemonic	Description
5E00	CD	CALL	Call routine in ROM to clear screen.
5E01	73	*	Enter least significant address first,
5E02	05	*	thus address 0573 enters as 73 then 05
5E03	01	LXI B	Load data from the next 2 bytes into BC
5E04	02	*	Color to plot
5E05	70	*	Length of loop (70 Hex = 112 Decimal)
5E06	11	LXI D	Load DE with the next two Bytes of data
5E07	30	*	Y Axis of Plot
5E08	00	*	X Axis of plot
5E09	CD	CALL	Call Plot Routine in ROM at 0600
5E0A	00	*	
5E0B	06	*	
5E0C	14	INR D	Increase Register D by 1
5E0D	05	DCR B	Decrease Register B by 1
5E0E	C2	JNZ	Jump if not 0
5E0F	09	*	Jump back to 5E09 to continue loop
5E10	5E	*	
5E11	09	RET	Return

First we have cleared the screen. Then, we load Register C with a color and Register D with how long our "FOR loop" is. D and E are the starting address of where we will print on the screen. The CALL to 0600 is the routine in the ROM to make a pixel at that given address. We increase Register D, thereby moving our pixel one address up on the X line, we then decrease Register B, which is B which is our "loop" We JUMP if B does not equal 0 back to the plot routine until it has plotted 112 times. When B = 0, we RETURN. Where? Back to BASIC. This is the end of the subroutine.

The routine took 18 Bytes in Machine Language and about 30 in BASIC. Opening up the variable X alone uses 6 Bytes, while the "X" is stored in a Register, not RAM, in our Machine routine.

I hope this has helped to show you the speed, power and ease of learning Machine Language. Please write me with your questions and I will do my best to answer them.

**8080 FOR EVERYONE OWNERS!**

**CORRECTION:** On Page Number 57 of the book, 8080 FOR EVERYONE By George A. Leggett, at Address 5F3C the Data Byte is E8 (E Eight) and NOT EB as shown. Also, in that Write To Tape Routine, you need not call 5E6A, since it is merely a feature in the ROM Monitor which I use to be sure that the second parameter is not lower than the first. Instead, Call 5E66 for the routine. If you do choose the call at 5E6A, 5E76 (LSB) and 5E77 (MSB) to the address of the jump where you want the routine to go upon acknowledging such a parameter error.

## THE INNARDS OF BASIC

### PART 2. COMMANDS, FUNCTIONS, AND OVERLAYS 101.

Harry Holloway, PO Box 2263, Ann Arbor, MI 48106.

In part 1 we looked at BASIC's use of low RAM. Now we continue with the BASIC interpreter in high RAM (6000-7FFF). Instead of giving a list of routines in order of memory locations I will try to group features that have a common function or a common application in modifications of BASIC. This time the emphasis will be on the information that is needed for the user to write overlay programs that will modify BASIC commands or functions. The descriptions will apply to Level II and mostly also to MV's Graphics version. A few users may find that they have an older version of Level II that has different code for the OUTPUT command and addresses above about 7820 that differ slightly from those given here. As before, all numerical values will be hexadecimal unless they have the suffix d.

#### Holes for the user's code

Part 1 showed several places that user code might be inserted in low RAM. There are also some substantial holes in the interpreter code that may be occupied rent free. The biggest holes correspond to code or data that are used by BASIC only for initialisation. After the first entry to the interpreter the program changes a flag byte at 6239 (25145d) from 00 to 01 so that subsequent entry (via RESET and R) will not cause reinitialisation. The holes are:

- (i) 6234-6238 Initialisation code.
- (ii) 623A-6297 Initialisation code.
- (iii) 62A1-6304 Startup messages, "BYTES FREE" etc. Graphics BASIC has the message area starting at 62AE because there is a test for the video banner with a bomb on failure inserted at 6298-62A4. (This is in addition to the test in the start code 4C03 JMP 4000 instead of JMP 6000.)
- (iv) 64D9-6552 Image of a storage area with embedded data that is moved down to 4C0D. (See part 1.)
- (v) 7FF8-7FFF Unused.

One more hole may be created at no cost by disabling the tests for PEEK and POKE and making a minor rearrangement of the code. The patch is:

```

76FE LDAXD
76FF JMP 727C ;exit from PEEK without doing address
              check.
7702 CALL 6E9A ;this moves up the POKE code to make
              the hole continuous.
7705 CALL 6AB2

```

## THE INNARDS OF BASIC, cont.

```

7708  PUSHD
7709  CALL 6835
770C  INRL
770D  CALL 75BF
7710  POPD
7711  STAXD
7712  RET

```

We can also save 3 bytes by eliminating the code for CLS, which is just a jump to the ROM routine at 0573. Now we need to change the dispatch addresses for POKE and CLS by changing the following memory locations:

```

6484  02
6494  73
6495  05

```

These changes create a hole from 7713 to 7740. It isn't big, but with overlays every extra byte of space is precious.

Reserved words and dispatch tables.

BASIC replaces the commands, functions and operators in the user's program with tokens, which are bytes with bit 7 set (i.e. >80=128d) to provide a distinction from ASCII values. The tokens fall into three groups:

- (i) Tokens 80-A1. Command words END through NEW.
- (ii) Tokens A2-B2. Miscellaneous operators TAB( through <.
- (iii) Tokens B3-CF. Functions SGN through POINT. Tokens from CB on require two arguments and all of these except CF require that the first argument be a string. The functions all take arguments in parentheses, but note that TAB and SPC only look like functions. For these the left paren. is actually part of the reserved word.

The reserved word list consists of a sequence of ASCII values with the beginning of each word marked by setting bit 7 of the ASCII for the first letter; e.g. ABS is represented by the bytes C1,42,53. The end of the table is marked with the byte 80. New words may be substituted for the existing ones provided that the new table is no longer than the old one and that the end marker is placed appropriately. Care is needed if the table contains a word that is the beginning of another reserved word, e.g. TO and TONE. In such cases the longer word must precede the shorter one or BASIC's logic will never find it.

The reserved word list is accompanied by tables of dispatch addresses, for command routines, for functions, and for arithmetic and logical operators. (The addresses are stored, as usually, low byte first.)

THE INNARDS OF BASIC, cont.

- 6305-633E Dispatch table for functions.
- 633F-6459 Reserved word list.
- 645A-649D Dispatch table for commands.
- 649E-64B2 Dispatch table for arithmetic and logical operators. Each address is preceded by a byte that is used to determine precedence.

The words, their tokens and the table, word, and dispatch addresses are:

WORD	TOKEN	TABLE	DISPATCH	WORD
END	80	6459	69EE	633F
FOR	81	645C	68F5	6342
NEXT	82	645E	6E4F	6345
DATA	83	6460	6B97	6349
INPUT	84	6462	6D56	634D
DIM	85	6464	70A3	6352
READ	86	6466	6D85	6355
LET	87	6468	68AE	6359
GOTO	88	646A	6B54	635C
RUN	89	646C	6B37	6360
IF	8A	646E	6C26	6363
RESTORE	8B	6470	69C2	6365
GOSUB	8C	6472	6B43	636C
RETURN	8D	6474	6B72	6371
REM	8E	6476	6B99	6377
STOP	8F	6478	69EC	637A
ON	90	647A	6C08	637E
PLOT	91	647C	7741	6380
SOUND	92	647E	779C	6384
tone	93	6480	77B1	6389
DEF	94	6482	72B1	638D
POKE	95	6484	7725	6390
PRINT	96	6486	6C4A	6394
CONT	97	6488	6A1D	6399
LIST	98	648A	6B9B	639D
COLOR	99	648C	775E	63A1
CLEAR	9A	648E	6AF1	63A6
CLOAD	9B	6490	762E	63AB
OSAVE	9C	6492	75D0	63B0
CLS	9D	6494	773E	63B5
REWIND	9E	6496	7B17	63B8
WINDOW	9F	6498	7B9D	63BE
OUTPUT	A0	649A	7B20	63C4
NEW	A1	649C	66D4	63CA
TAB	A2			63CD
TO	A3			63D1
FN	A4			63D3
SPO	A5			63D5
THEN	A6			63D9
NOT	A7			63DD
STEP	A8			63E0
+	A9	649F	7C97	63E4

THE INNARDS OF BASIC, cont.

WORD	TOKEN	ADDRESSES		
		TABLE	DISPATCH	WORD
-	AA	64A2	78BB	63E5
*	AB	64A5	79F9	63E6
/	AC	64A8	7A5A	63E7
^	AD	64AB	7DA8	63E8
AND	AE	64AE	6FFC	63EA
OR	AF	64B1	6FFB	63EC
>	BO			63EE
=	B1			63EF
<	B2			63F0
SGN	B3	6305	7B15	63F1
INT	B4	6307	7BD9	63F4
ABS	B5	6309	7B2B	63F7
USR	B6	630B	4C10	63FA
FRE	B7	630D	724B	63FD
FIRE	B8	630F	77E7	6400
JOY	B9	6311	77F1	6404
POS	BA	6313	7279	6407
SQR	BB	6315	7D9F	640A
RND	BC	6317	7E7E	640D
LOG	BD	6319	79BA	6410
EXP	BE	631B	7DED	6413
COS	BF	631D	7EF3	6416
SIN	CO	631F	7EF9	6419
TAN	C1	6321	7F5A	641C
ATN	C2	6323	7F6F	641F
PEEK	C3	6325	76FB	6422
LEN	C4	6327	74FD	6426
STR\$	C5	6329	7315	6429
VAL	C6	632B	7597	642D
ASC	C7	632D	750C	6430
CHR\$	C8	632F	751D	6433
POT	C9	6331	77DF	6437
INSTR\$	CA	6333	7879	643A
DUMMY	CB	6335	0000	6440
LEFT\$	CC	6337	752D	6445
RIGHT\$	CD	6339	755D	644A
MID\$	CE	633B	7567	6450
POINT	CF	633D	77CB	6454

User-installed BASIC commands.

Now with some of the tedious details out of the way we can get on with the problem of modifying BASIC. One reserved word that we can easily do without is LET because if we write

X = 2      instead of      LET X = 2

the BASIC interpreter will correctly deduce that we are making an assignment and jump to the correct code. Thus, we can replace LET with the name of another command (for example, in FASTLINE it is replaced by BOX) provided that we don't use more than three letters. Remember that bit 7 must be set for the ASCII of the first letter and that if we use less than three letters the extra space in the table must be closed up.

## THE INNARDS OF BASIC, cont.

The code for our new command may be put into any of the available holes and the dispatch address will be changed to point to the new code. There are only two rules that must be followed with the new code. First, when we have done our thing we need a RET instruction to return to the BASIC interpreter. Second, BASIC uses the contents of the HL register pair to keep track of its place in the user's program, so we must preserve HL. In most cases the new code will include PUSH H and end with POP H, RET.

A second target on our hit list is DUMMY. This seems to have been included by Microsoft to permit insertion of a new function, but in its present form it's about as useful as udders on a bull because it is only accepted with two arguments, the first of which must be a string. The obvious way out is to convert DUMMY to a command (This was done to get the LINE command in FASTLINE.) The patch that is needed is:

```

7704 JMP PATCH ;PATCH is the location of the user's
                    code. This replaces a jump to the
                    syntax error routine that occurs if a
                    function is found where a command is
                    expected. The token less 80 is in A.

PATCH CPI 48 ;Is it really DUMMY?
JNZ 65C1 ;No. Bounce back to syntax error.
XCHG
LXIB UADDR ;Where UADDR is the user's routine
                    address.
JMP 69B0 ;Patch back into BASIC's command
                    handler.

```

With this patch the dispatch address for DUMMY is bypassed and need not be changed.

Handling command arguments.

Most user-installed commands will be followed by one or more numerical arguments. After the jump to the command code, HL will be pointing past the command token to the next character in the user's program area, which will usually be the beginning of the argument string. The BASIC interpreter has several subroutines that may be used to read in the arguments. In each case HL will be updated to point past the argument.

```

GETBYT 75BF Reads a literal number or a variable name or
                    evaluates an expression for an unsigned 8-
                    bit integer. Returns with the value in A and
                    in E.

NEXTPRM 7792 Like GETBYT, but skips over a comma to get
                    the value.

FRMNUM 6E9A Evaluates an argument and puts the result in
                    the FAC. (See part 1.)

```

## THE INNARDS OF BASIC, cont.

DEINT 6AB2 Converts the contents of the FAC to a 16-bit signed integer and puts the result into DE.

CONINT 75C2 Converts the contents of the FAC to an 8-bit unsigned integer in both A and E.

Modifying a function.

Treatment of functions follows similar lines. As an example we will consider `USR`, which has been partially described in previous issues of *Interaction*. An assignment of `USR`, such as `B = USR(O)`

will give an unconditional jump to the address that is poked into the low-RAM locations 4C11-2 (19473-4d) or into the dispatch table at 630B-C (25335-6d). In previously published examples the parenthetic argument was ignored, but it is quite easily used. When the BASIC interpreter gets to the code for a function of a single argument the expression for the argument has already been evaluated and the result has been put into the FAC, from where we may use it as desired. As an example we may index into a series of different user routines by starting our user code with

```
CALL 6AB2 ;Get the argument into DE.
PUSHH ;Save BASIC's program pointer.
LXIHL JTABLE ;Point to the user's table of addresses.
DADD ;Add in the offset.
DADD ;Now HL points to the required address,
MOVEM ;which goes via DE
INXH
MOVDM
XCHG ;to HL
FCHL ;to PC and away we go.
```

Note that we will need to recover the program counter with `POPH` before returning from the subroutines.

If `USR` or any other nonstring function is to be used as such, BASIC's convention is that the value of the function will be left in the FAC before the return is made from the function code. With integer arguments one may use

GIVDBL 7267 Subtracts the contents of DE from those of HL and puts the result as a signed number into the FAC.

SNGFLT 727C Converts the contents of A to an unsigned number and puts the result into the FAC.

That's it for now. Next time some floating point routines.

Two addresses have changed recently of interest to all:

George Leggett 20562 Woodward Mt Clemens, MI 48043

Slagh System Services Box 53 Dearborn, MI 48121  
Phone 313-581-8593



## PROGRAM CORRECTIONS

In Vol. II No. 6 (December, 1981), there was an incorrect line listing in the VEGA\$ program. The Slagh BASIC overlay replaces the DUMMY command with a PORT command allowing control of the RS-232 port. Thus when you use the overlay on any program having a DUMMY command it will replace it with PORT and will not execute either command because of the resulting syntax errors.

Thus line 1380 should read -  
1380 PRINT DUMMY ("RESET",Ø)

## FIRE 3 (ALSO IN VOL. II, NO. 6) CORRECTION

In the FIRE 3 instructions (last paragraph), I suggested not using the method of POKEing 19709 and 19710 to save the machine language subroutine with the program. Many subscribers, including Kevin TenBrook, have told me this method works. Kevin in fact says that is how he accomplished the tape copy he sent me. I had difficulty accomplishing a CSAVE and thus recommended not using that method, apparently I was making some error in doing it. For those not on Micro Video's mailing list I quote from their recommendations "When you're certain your program is finished, load in your subroutine at the specified address and POKE the ending address of your subroutine into 19709-19710 (again LSB and MSB). Now when you CSAVE the program, the whole works will be saved in one step."

## FOREST FIRE

By Gregg Pittenger 3430 Dresden Columbus, OH 43224

Three fires have started in the Interact forest. Your goal is to save as many of the 81 sectors as possible. You can fight the fire in two ways. You can drop chemicals to help extinguish the fires. As in real life the chemical is not always effective but if they are they reduce the burning time by one third. Unchecked fires burn out after nine turns.

In the second method by typing B when prompted (B)ACKFIRE? - you can start a backfire if the sector is wooded. The backfire will not spread and will burn out in the next turn, forming a barrier against the spreading fire.

When all the fires are out, your rating will be the number of sectors remaining. Saving even half the trees can be difficult.

## FOREST FIRE, cont.

```

1 PRINT CHR$(8)
2 CLS:PRINT"  FOREST FIRE  -----":PRINT:PRINT"FROM 'STIMULA
TING"
3 PRINT"SIMULATIONS', 2ND ED. BY C.W. ENGEL":PRINT:PRINT"  ADAPTED FOR
"
4 PRINT"  INTERACT BY:  GREGG W.PITTENGER":PRINT:PRINT"SEE 'AFOREMENTI
OND"
5 PRINT"BOOK FOR GAME  INSTRUCTIONS.":PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
6 C0=0:C1=1:C2=2:C3=3
7 WINDOW11
8 COLOR 7,0,2,1
10 DIM L(9,9)
20 FOR R=1 TO 9:FOR C=1 TO 9
30 L(R,C)=10
40 NEXT C,R
50 FOR I=1 TO 3
60 R=INT(9*RND(1)+1)
70 C=INT(9*RND(1)+1)
80 L(R,C)=9
90 NEXT I:CLS
95 REM PRINT GRID
100 OUTPUT"123456789",19,72,C1
110 FOR R=1 TO 9
120 OUTPUT R,6,72-R*6,C1
130 FOR C=1 TO 9
135 Y1=72-R*6:X1=13+C*6
140 IF L(R,C)=10 THEN OUTPUT"Y",X1,Y1,C2:OUTPUT" ",X1,Y1,C2:GOTO 170
150 OUTPUT CHR$(1),X1,Y1,C0
160 IF L(R,C)>0 AND L(R,C)<10 THEN OUTPUT"@",X1,Y1,C3
170 NEXT C
180 NEXT R
195 REM INPUT ROUTINE
200 PRINT:PRINT"ROW?";:R$=INSTR$(1):PRINTR$
210 R=VAL(R$):IFR<1ORR>9THEN200
220 PRINT"COLUMN?";:C$=INSTR$(1):PRINTC$
230 C=VAL(C$):IFC<1ORC>9THEN220
240 PRINT"(B)ACKFIRE?";:B$=INSTR$(1):IFB$="B"THEN360
250 PRINT:PRINT"DROPPING CHEMICAL";
260 FORI=-1TO1:FORJ=-1TO1
270 A=R+I:B=C+J
280 IFA<1ORA>9ORB<1ORB>9THEN340
290 IFL(A,B)<1ORL(A,B)=10THEN340
300 IFRND(1)>.5THEN340
310 L(A,B)=L(A,B)-3
340 NEXTJ:NEXTI
350 GOTO400
360 PRINT:PRINT"BACKFIRE STARTED";
370 IF L(R,C)=10 THEN L(R,C)=2
395 REM SPREAD FIRE
400 FOR R=1 TO 9:FOR C=1 TO 9
410 IF L(R,C)<1 OR L(R,C)>9 THEN 500

```

## FOREST FIRE, cont.

```

420 IF L(R,C)<3 THEN 500
430 I=INT(3*RND(1)-1)
440 J=INT(3*RND(1)-1)
450 A=R+I;B=C+J
460 IF A<1 OR A>9 OR B<1 OR B>9 THEN 500
470 IF L(A,B)<>10 THEN 500
480 IF RND(1)<.3 THEN 500
490 L(A,B)=11
500 NEXTC:NEXTR
505 REM BURN FIRE AND COUNT
510 F=0
520 FOR R=1 TO 9
530 FOR C=1 TO 9
540 T=L(R,C)
550 IF T=11 THEN T=9
560 IF T>0 AND T<10 THEN T=T-1:F=F+1
570 L(R,C)=T
580 NEXTC:NEXTR
590 IF F<1 THEN 620
600 GOTO 110
615 REM COUNT WOODS RATING
620 C=0
630 FOR R=1 TO 9:FOR C=1 TO 9
640 IF L(R,C)=10 THEN W=W+1
650 NEXTC:NEXTR
660 R=W
680 PRINT"YOUR RATING: ";R;
685 FOR Z=1 TO 1000:NEXTZ
690 PRINT"PLAY AGAIN?";Y$=INSTR$(1):PRINT Y$
700 IF Y$="Y" THEN 20
710 CLS:WINDOW77

```

Ok

## PRODUCT REVIEWS

SABRE PORT for the INTERACT computer  
 Manufactured by: Sabre, Inc.  
 1415 Creek Hollow Dr.  
 Seabrook, Tx. 77586

Review by: Nick Speaks  
 6732 Stonecrest Dr.  
 Charlotte, N.C. 28212  
 A.C.704-535-4468

I feel that it is in the best interest of the Interact community to submit my experience on installing and using the Sabre Port. Without getting into a lengthy and technical disertation on the operating theory of the Sabre port (which I am not qualified to do anyway) I will attempt to relay my experience in installing and using the port.

## PRODUCT REVIEWS, cont.

Considering that the Interact was never really designed for or intended for expansion by its' original manufacturer several enterprising individuals have attempted and successfully completed expansion projects for the same. This review is about the Sabre Port, designed and built by Kevin Tenbrook and his associates. I purchased one a couple of weeks ago and have been using it successfully ever since. I am going to be brief and to the point in this product review by first listing the advantages of the port and then listing the disadvantages as "I" see them. Being fairly new to computers "my" disadvantages may not be disadvantages at all(keep this in mind). The advantages of the Sabre Port are as follows:

ADVANTAGES

- 1.The cost is \$26.00 complete
- 2.Includes all documentation relating to the installation and operation of the port.
- 3.The installation instructions are very simple to read and easy to follow(by the way, the installation instructions contain a complete pictorial layout of the Interact PC board main electronics with all the ICs and most of the passive components labeled which I consider to be worth \$10 by itself).
- 4.Included in the purchase is an assembly language routine to use with machine language programs(this is an area that I am unfamiliar with at this time and would mean more to experienced programmers).
- 5.The Level II basic overlay printer program tape is included with the purchase.
- 6.The operating instructions are very detailed and extremely flexible.
  - 1)Baud rates from 110 to 9600
  - 2)Line widths from 1 to 255 characters
  - 3)Automatic line feed option
  - 4)Manual or software control of the port
  - 5)Very flexible word length(start,stop bits,etc.)
  - 6)Auto Paging feature
  - 7)Stable operation with CTS(clear to send) loop.

DISADVANTAGES(possible?)

- 1)There is a small amount of soldering(6 connections) required to the main Interact PC board(this was not a problem for me, but it could be for someone who has never soldered). If you do not know how to solder, it would be wise to have a friend that knows how, to do this part(the instructions also recommend this).
- 2)The Sabre port does not give you the ability to put information to the TV screen and the printer at the same time. You have to output to the TV or the printer, but not both simultaneously. I've got a suspicion this could be corrected with software that would put to the screen and the printer alternately and would appear to be putting to both simultaneously.

SUMMARY

In summary, the SABRE Port is a very flexible and useful addition to the Interact computer. The fact that it is inexpensive and reliable leads me to recommend it for printer support to the Interact.

## PRODUCT REVIEWS, cont.

## ANOTHER SABRE PORT REVIEW

## REVIEWED BY

Jerry Goerz KY Utilities One Quality St. Lexington, KY 40507

At a price of only \$24.99 plus \$2.00 postage for a serial printer port, BASIC overlay tape, overlay listing and instructions, I could hardly believe the ad which appeared on page 19 on Interaction Volume II #5. So I wrote to SABRE with my questions, and they promptly responded with a data sheet and a handwritten reply. Yes, the speed is selectable by using POKES, from 110 to 9600 baud. Yes, clear-to-send "handshaking" is provided. No, you cannot use the port for duplex communication, as it is output-only. Yes SABRE would provide an overlay compatible with both LEVEL II and MICRO-SOFT 8K FAST GRAPHICS BASIC. Yes, other serial ports and additional ROM or RAM may be added without conflict with the SABRE port. I had only one reservation when placing my order: SABRE warns up-front that the port is potted in epoxy, so I knew it could never be serviced and no schematic would be provided.

The port arrive promptly, accompanied by easily-understood installation instructions and large drawings of the main CPU board to which six wires must be soldered on the top side. A very fine-tipped low-voltage soldering iron must be used - I used a Wahl Isotip with ultra-fine tip. A drill with 1/4 and 5/32 inch bits is also required, for mounting a miniature and subminiature jack to the right side of the Interact. The port's epoxy module tucks into a corner of the main electronics' assembly. Total installation time is less than two hours. I found only one flaw in the installation instructions: SABRE recommends removing a portion of the metal top panel (cover) of the main electronics assembly, to make future access easier, but I believe the FCC would prefer that we simply cut the cover in half with a tin-snips so that, when re-installed, no RF signals will escape.

Once the port is installed, a 3-wire cable must be fabricated to mate the computer to the printer. The Signal Ground wire connects the ring of the miniature plug to pin 7 of the DB25 male plug which mates with my COMET Model 8300R printer. The Received Data wire connects the tip of the miniature plug to pin 3 of the DB25. The "clear-to-send" wire connects the tip of the subminiature plug to pin 20 of the DB25, which is defined by C. Itoh Electronics as Data Terminal Ready (DTR). Pinouts and signal names may vary with other printers.

After fumbling with POKES and printer DIP-switches for several hours, I realized that a clever printer and a clever port were being mashed together by a dumb operator. For example, a DIP switch on the printer selects either one or two stop bits per "word". The SABRE port defaults to two stop bits, but this may be changed from 1 to 255 with a POKE. I accepted the default value and set my printer accordingly. Likewise, the line width, baud rate, line feed, bits-per-character, and page length must all

## PRODUCT REVIEWS, cont.

be set to agree at both the port and printer. I had a lot of trouble finding the source of undesired form-feeds, and finally discovered that the port has a default value of 58 lines per form-feed, which can be changed or defeated with POKES, and the port's line counter must be reset to zero with CONTROL-R whenever aligning a new page in the printer.

The SABRE port is extremely versatile. To divert output from the screen to the printer, simply type in a Control-O. To return output to the screen, type another Control-O. To switch output between screen and printer in a program, POKES are used. Whenever an "OK" appears at the end of a list or run, output returns automatically to the screen. I don't think it is possible to use one PRINT instruction to output to both the screen and the printer simultaneously, but an identical PRINT instruction can be repeated following a POKE to create such an effect.

SABRE seems to be here to stay. They sent me a personalized letter generated on the SABRE word processor, which they hope to advertise soon in Interaction for under \$10.00. I was impressed!

## FIVE DICE GAMES

by Vincent Chobot 2716 S. Cuyler Berwyn, IL 60402

This program consists of 5 dice games. They are all strictly games of chance for 1 to 5 players (1 plays against the computer). On the tone and indicator hit any key to throw the dice. The names and rules are

- 1 "50" Score 5 points for each pair except 2 sixes score 25 and 2 threes wipe out the total. The first player to score 50 wins.
- 2 "Chicago" Try for a total of 2 in the first round, 3 in the second, up to 12 in the eleventh round. If the points are made it is added to your score. The High Score wins.
- 3 "Drop Dead" The score is a total of the points on the dice, except if a 2 or 5 is on any dice then the score is zero and those dice are eliminated in further throws. You keep throwing until you're out of dice. The High Score wins.
- 4 "Round the Clock" Try to make the sequence 1 to 12. The first to reach 12 wins.
- 5 "Hearts" Score 5 points for each die in sequence from 1 to 6 i.e.  $314565 = 5$  points  $125365 = 15$  pts except all from 1 to 6 scores 35 points; also 3 ones wipes out your total. The High Score wins.

## FIVE DICE GAMES, cont.

```

5 CLS:COLOR2,7,0,3:PRINTCHR$(8):CLEAR(75)
10 REM DICE V.CHOBOT 4/81
20 GOSUB5000:FORX=1TO1000:NEXT
30 H=1:I=1:X1=5:Y1=16:D(I)=1:PRINT:PRINTTAB(2)"FIFTY"TAB(10)"CHICAGO"
35 GOSUB2025:X1=55:D(I)=2:GOSUB2025
40 PRINT:PRINTTAB(2)"DROP DEAD":X1=5:D(I)=3:GOSUB2025
45 PRINT:PRINTTAB(2)"ROUND THE CLOCK":D(I)=4:GOSUB2025
50 PRINT:PRINTTAB(2)"HEARTS":D(I)=5:GOSUB2025
60 PRINT:PRINT"GAME "CHR$(35);:INPUTG:IFG>5OR6<1THEN60
90 IFP2=1THEN110
100 CLS:ONGGOTO300,400,500,600,700
110 CLS:ONGGOTO310,410,510,610,710
200 FORX=1TON:N$(J)=" ":NEXT
210 PRINT:PRINT"HOW MANY PLAYERS?"
220 INPUT"MAXIMUM OF 5";N:IFN>5THEN220
225 PRINT:PRINT"TYPE YOUR NAME"
230 FORJ=1TON
240 INPUTN$(J)
245 NEXT:CLS
247 IFN=1THENN=2:N$(2)="COMPUTER"
250 FORJ=1TON:S(J)=0:NEXT:M=0:W=0:W1=0
253 FORJ=1TON
255 L=LEN(N$(J))*6+20
256 V=42-6*J
260 OUTPUTN$(J),20,V,3
270 FORX=LTO84STEP6:OUTPUT" ",X,V,1:NEXTX,J:RETURN
300 PRINT:PRINT"FIFTY":GOSUB200
305 GOTO320
310 GOSUB250
320 H=2:X1=50:Y1=57
340 OUTPUT"FIFTY",45,72,3:OUTPUT"SCORE",72,45,3
360 GOSUB800
380 GOTO360
400 PRINT:PRINT"CHICAGO":GOSUB200
405 GOTO420
410 GOSUB250
420 H=2:X1=46:Y1=53:P=2:COLOR1,7,0,3
430 OUTPUT"CHICAGO",35,72,3:OUTPUT"POINT",72,64,2
440 OUTPUT"SCORE",72,45,3
460 FORP=2TO12
470 OUTPUTP-1,75,57,0:OUTPUTP,75,57,1
475 GOSUB800
480 NEXT
485 IFM=0THEN3590
487 IFW1>0THEN3600
490 J=W:GOTO3500
500 PRINT:PRINT"DROP DEAD":GOSUB200
505 GOTO520
510 GOSUB250
520 H=5:X1=15:Y1=55:COLOR4,7,0,3
540 OUTPUT"SCORE",72,45,3:OUTPUT"TOTAL",72,64,2:GOSUB590
560 GOSUB800
565 IFM=0THEN3590
570 IFW1>0THEN3600

```

## FIVE DICE GAMES, cont.

```
580 J=W:GOTO3500
590 OUTPUT"DROP DEAD",35,72,3:RETURN
600 PRINT:PRINT"ROUND THE CLOCK":GOSUB200
605 GOTO620
610 GOSUB250
620 H=2:X1=49:Y1=57:R=1:COLOR5,0,7,2
630 FORJ=1TON:S(J)=1:NEXT
640 OUTPUT"ROUND THE CLOCK",15,72,3:OUTPUT"POINT",72,45,3
650 OUTPUT"ROUND",20,45,3
660 GOSUB800
670 R=R+1:OUTPUTR-1,46,45,0:OUTPUTR,46,45,2
680 GOTO660
700 PRINT:PRINT"HEARTS":GOSUB200
705 GOTO720
710 GOSUB250
720 H=6:X1=25:Y1=57:Q=1:COLOR6,7,0,3
740 OUTPUT"HEARTS",42,72,3:OUTPUT"SCORE",72,45,3
750 GOSUB800
760 GOTO750
800 FORJ=1TON
805 V=42-6*J
810 OUTPUTCHR$(91),14,V,1
820 IFN=2ANDJ=2ANDN$(2)="COMPUTER"THENGOSUB950:GOTO860
840 SOUND4,16:SOUND4,17
850 A$=INSTR$(1)
860 GOSUB2000
865 IFG=3ANDZ=1THEN1200
870 ONGGOSUB1000,1100,1200,1300,1400
880 FORX=85TO100STEP5:OUTPUTCHR$(1),X,V,0:NEXT
890 OUTPUTS(J),80,V,1
895 OUTPUTCHR$(91),14,V,0
900 NEXTJ
920 RETURN
950 FORX=1TO1000:NEXT:FORX=1TO50:SOUND5,X:NEXT:SOUND7,4096:RETURN
995 FORX=1TO100:SOUND5,X:NEXT:SOUND7,4096:GOTO860
1000 IFD(1)<>D(2)THENRETURN
1010 IFD(1)+D(2)=6THENS(J)=0:GOSUB4000:RETURN
1020 IFD(1)+D(2)=12THENS(J)=S(J)+20
1030 S(J)=S(J)+5
1040 IFS(J)<50THENRETURN
1050 GOTO3500
1100 IFD(1)+D(2)=PTHENS(J)=S(J)+P
1110 IFS(J)=MANDJ<>WTHENW1=J
1120 IFS(J)>MTHENM=S(J):W=J:W1=0
1130 RETURN
1200 Q=1:Z=0
1205 E=10*(H+1)
1210 FORI=1TOH
1220 IFD(I)=2ORD(I)=5THENH=H-1:Q=0
1230 S=S+D(I)
1240 NEXT
1245 S=S*Q
1250 OUTPUTS,E,55,1
1260 S(J)=S(J)+S:S=0
```



## FIVE DICE GAMES, cont.

```

1262 IFS(J)=MANDJ<>WTHENW1=J
1263 IFS(J)>MTHENM=S(J):W=J:W1=0
1265 FORX=78TO96STEP5:OUTPUTCHR$(1),X,58,0:NEXT:OUTPUTS(J),78,58,2
1270 IFH=0THENH=5:OUTPUTS,E,55,0:GOTO1290
1280 Z=1:GOTO820
1290 FORX=1TO5:OUTPUT"DROP DEAD",35,72,0:GOSUB590:SOUND0,9100:NEXT
1295 SOUND4,17:RETURN
1300 IFS(J)>6THEN1370
1310 IFD(1)=S(J)ORD(2)=S(J)THEN1350
1330 IFD(1)+D(2)=S(J)THEN1380
1340 RETURN
1350 IFD(1)+D(2)<>2*S(J)+1THEN1380
1360 S(J)=S(J)+2:RETURN
1370 IFD(1)+D(2)<>S(J)THENRETURN
1380 S(J)=S(J)+1
1390 IFS(J)<=12THEN RETURN
1395 GOTO3500
1400 FORK=1TO6:C(K)=0:NEXT
1410 FORI=1TOH:K=D(I):C(K)=C(K)+1:NEXT
1420 IFC(1)>=3THENS=0:Q=0:GOSUB4000:GOTO1460
1430 FORK=1TO6
1440 IFC(K)=0THENS=(K-1)*5:GOTO1460
1450 NEXT:S=35
1460 OUTPUTS,80,58,1
1470 S(J)=(S(J)+S)*Q:Q=1
1480 IFS(J)>=100THEN3500
1490 RETURN
2000 FORI=1TOH:D(I)=0:NEXT
2005 FORX=X1TOX2+20STEP5:FORY=Y1TOY1+2STEP2:OUTPUTCHR$(1),X,Y,0:NEXTY,
X
2010 FORI=1TOH
2020 D(I)=INT(6*RND(1)+1)
2025 Y2=Y1-3:Y3=Y1-1:Y4=Y1+1
2030 X2=X1+(I-1)*10:X3=X2+1:X4=X2+5
2100 FORX=X2TOX2+2STEP2
2110 FORY=Y1TOY1+2STEP2
2120 OUTPUTCHR$(1),X,Y,1
2130 NEXTY,X
2140 OND(I)GOTO2170,2160,2160,2150,2150,2145
2145 PLOTX3,Y3,2:PLOTX4,Y3,2
2150 FORY=Y2TOY4STEP4:PLOTX3,Y,2:PLOTX4,Y,2:NEXT
2155 OND(I)-3GOTO2180,2170,2180
2160 PLOTX3,Y4,2:PLOTX4,Y2,2
2165 IFD(I)=2GOTO2180
2170 PLOTX3+2,Y3,2
2180 IFI=HTHENRETURN
2190 NEXT
3500 FORX=1TO1000:NEXT:CLS
3505 TONE50,500:TONE100,250
3510 PRINT"CONGRATULATIONS!!"
3520 PRINT:PRINT:PRINTTAB((17-LEN(N$(J)))/2)N$(J)
3530 PRINT:PRINT:PRINTTAB(5)"YOU WIN":FORX=1TO1000:NEXT
3550 PRINT:INPUT"GAME: KEY 0=QUIT,1=SAME,2=OTHER";P1
3555 IFP1=0THENEND

```

FIVE DICE GAMES, cont.

```

3560 PRINT:INPUT"PLAYERS: 1=SAME 0=DIFFERENT";P2
3570 IFP1=1ANDP2=1THEN110
3580 IFP1=1THEN100
3585 GOTO30
3590 CLS:PRINT"SHUCKS NOBODY GOTA POINT":FORX=1TO500:NEXT:CLS:GOTO3640
3600 CLS:PRINT"IT'S A TIE!!
3610 PRINT"TO PLAYOFF, HIT ANY KEY":A$=INSTR$(1):CLS
3620 IFW1=1THENN$(2)=N$(W):N=2:GOTO3640
3630 N$(1)=N$(W):N$(2)=N$(W1):N=2
3640 ONG-1GOTO410,510
4000 TONE300,133:TONE500,50:RETURN
5000 A$="DI":H=1
5010 OUTPUTA$,0,5,1
5020 FORL=0TO10:FORK=1TO5:IFPOINT(L,K)=1THENGOSUB5050
5030 NEXTK,L:IFA$="CE"THENRETURN
5040 FORX=1TO1000:NEXT:CLS:A$="CE":GOTO5010
5050 X1=8*L+20:Y1=8*K+20:GOSUB2010:RETURN
    
```

Ok

INTERACTION ON TAPE

As in the past, George Leggett will continue to supply copies of the BASIC programs on cassettes. The price is \$6 per issue. George can also supply you with previous years programs at the same per issue price. All copies are guaranteed loadable and are supplied in Level II BASIC. Contact George at his NEW address: George Leggett 20562 Woodward Mt. Clemens, MI 48043

WORD GUESS

BY Bob Draganski 14301 Harrison Livonia, MI 48154

This guessing game for two players gradually uncovers parts of a word or phrase. The first player to press his fire button gets the chance to type in his guess. If he is correct, he is awarded points; more points for a quicker guess..A point is deducted for incorrect guesses. The first to get twenty points wins. The data statements (lines 1-8 and 300-499) can be changed. Be sure to check that the number of data items is larger than the maximum value of H in line 13 and that if you change the last data item, change line 26.

- 1 DATAGO FLY A KITE,SUPER MAN,FIRST BASE,LUCKY STIFF,NICE TO SEE YOU
- 2 DATATHE BIG TOP,VICTORY CRY,I LOVE CANDY,GOING TO THE DOGS,NOTHING
- 3 DATAFULL SPEED AHEAD,SLOW DOWN,STAR TREK,YOUNGSTER,GOOD DAY TO YOU
- 4 DATAROCK A BYE BABY,HOME ON THE RANGE
- 5 DATACHILDREN,POLICEMAN,MAILMAN,TOMORROW,PRINCIPLE,TOUGH LUCK,RIGHT O
- N
- 6 DATAGOOD NIGHT,QUEEN FOR A DAY,WEEKEND TRIP,ADVANTAGE

## WORD GUESS, cont.

```

7 DATATONSILS,NOT TOO FAST,SPEED KILLS,DATING GAME,YOU BIG DUMMY,MIGHT
Y
8 DATAMARRY ME PLEASE,THE ALIEN,GOT TO GO NOW,ENERGY
9 A(1)=10:A(2)=8:A(3)=12:A(4)=9
10 RESTORE:CLS:OUTPUT"PUSH FIRE BUTTON",6,40,1
11 IFFIRE(0)=0ORFIRE(1)=0THEN13
12 H=RND(1):GOTO11
13 H=INT(63*RND(1)):FORI=1TOH:READA$:NEXT
14 CLS:COLOR4,3,0,4
15 POKE19215,25
20 READA$:PRINTCHR$(8)
21 L=LEN(A$)
22 FORP1=1TOL:IFMID$(A$,P1,1)=" "THEN24
23 OUTPUTCHR$(1),6*P1,57,2
24 NEXT
25 OUTPUTR,6,70,1:OUTPUTQ,80,70,1
26 IFA$="BULL FROG"THENRESTORE
27 OUTPUTA$,6,12,3
28 IFR>14ORQ>14THEN200
30 K=5:FORU=1TO4:K=K-1
40 FORX=6TOL*6+6
50 IFPOINT(X,A(U))=3THENTONE30,100:GOTO55
51 PLOTX,45+A(U),0
55 IFFIRE(0)=0THENOUTPUT"<--",30,70,2:Z=1:GOSUB100
56 IFFIRE(1)=0THENOUTPUT"-->",50,70,2:Z=2:GOSUB100
60 NEXT:TONE100,100:NEXT
70 OUTPUT"NO SCORE",30,30,1:TONE90,100:OUTPUTA$,6,57,1:TONE100,750:CLS
80 GOTO20
100 OUTPUT"^",6,52,2:FORP1=1TOL:IFMID$(A$,P1,1)=" "THEN102
101 OUTPUTCHR$(1),6*P1,57,2
102 NEXT
103 FORI=1TOL
104 IFMID$(A$,I,1)=" "THEN140
105 B$=INSTR$(1)
110 IFB$=MID$(A$,I,1)THENOUTPUTB$,I*6,57,1:TONE40,100:GOTO140
120 OUTPUT"WRONG",30,30,1:TONE100,750:IFZ=1THENR=R-1
121 IFZ=2THENQ=Q-1
122 OUTPUTB$,6*I,63,1:TONE100,750:OUTPUTA$,6,57,1:TONE100,750:CLS
139 GOTO20
140 OUTPUT"^",6*I,52,0:OUTPUT"^",6*I+6,52,2:NEXT
141 IFZ=1THENR=R+K
142 IFZ=2THENQ=Q+K
145 OUTPUT"CORRECT",30,30,1:TONE70,100:TONE100,750:CLS:GOTO14
200 OUTPUT"A WINNER",30,30,1:Q=0:R=0
210 IFFIRE(0)=0ORFIRE(1)=0THENCLS:GOTO10
220 GOTO210
300 DATA COOL CATS, FROM THE GRAVE, WALK ON WATER, LET THE SUN SHINE
305 DATA YOU'RE CUTE, LET'S DANCE, PROBLEMS PROBLEMS, YESTERDAY, NEIGHBORHO
OD
306 DATA AIRPLANE, SPACESHIP, QUICKSAND
310 DATA KILL THE UMPIRE, JUST FRIENDS, SLOW MOTION, VAMPIRE BAT, FASTER FO
OL
320 DATA JUST FOR FUN, LOCOMOTION, I GOT THE MONEY, HOT LIPS, HEAVEN CAN WA
IT

```

WORD GUESS, cont.

- 330 DATAI HATE TO GO, MOVE IT, YOU SAID IT, QUESTION PLEASE?, LOW BRIDGE
- 340 DATASPEED TO BURN, LITTLE BO PEEP, PRETTY BABY, HOLY SMOKES .
- 350 DATAPRESIDENT, GOING FISHING, WASHINGTON, QUIT TALKING, BULL FROG
- 360 DATABJ AND THE BEAR, DON RICKLES, ROCK AND ROLL, BO DEREK
- 370 DATAINTERACT COMPUTER, JELLY BEAN, EASTER BUNNY, CAR POOL
- 380 DATAASH WEDNESDAY, AUGUST, BOWLING, WALLPAPER, BEDROOM SET, LOCAL YOKEL
- 499 DATABULL FROG

DISAPPOINTING CALLS

During the time this winter when I was unable to do any newsletter work, I got a number of calls. While I encourage everyone to call me (when you can catch me at home) many of these callers while telling me how much they liked the newsletter have never been paid subscribers. Publishing INTERACTION is not my little gold mine, the money left over after the bills are paid is not really an incentive to keep doing the newsletter. It is at times extremely frustrating to think about the people who, in a very real sense, are stealing from me. Rather than giving copies of INTERACTION to your friends, encourage them to subscribe. The real losers from my frustration will be everyone, both innocent and guilty. Now that Interact production is truly dead, it can only be a dwindling market and we should encourage everyone to be productive, not destructive to the Interact's future.



BASIC WRITTEN  
WORD PROCESSOR  
for your INTERACT!

Page Titling, Tab Set, Indent, Numbering for Outline Data, Line Spacing, Line Feed Suppress, Line Feed Unsuppress,

Left Justified                                Centered                                Right Justified

Line Break, Initialization of Port and Printer, UPPER/lower case, Underlining, Well Defined Subroutines, Good Documentation with examples, Continuing Support, and Emphasized Printing.

This word processor was written for the Micro-Video RS-232 Port using an EPSON MX-80 printer. I will be glad to answer questions about the program.

COST-\$15.00

Write to: Joseph Kleczka, 357 Joya Loop, Los Alamos, N.M. 87544  
or Phone: Area Code (505)-672-1532

MEMORY

4116 (16K x 1 dynamic RAM) IC's  
\$14 per set of 8 postpaid  
from Joe Gilbert 320 Gold Mine Dr San Francisco, CA 94131

\*\*\*\*\*

INTERACT COMMUNICATIONS SYSTEM

The command menu has the following commands:

- # 1 Start terminal emulation using the UBOM RS-232 port.  
(Exit command mode, and begin polling port and keyboard)
- # 2 Re-initialize the terminal profile and the translation table.  
(Restores settings, after modifications, to default values)
- # 3 Select the escape & break codes, change the translation table,  
and select automatic carriage return or line feed on input.  
(Defines the Interact's keys)
- # 4 Select the bits per second (or baud rate) and lines scrolled.  
(Sets the data communication channel rate)
- # 5 Select even, odd or no parity & one or two stop bits  
& seven or eight bit character modes.  
(Sets the data format on the communication channel)
- # 6 Generate, offset, and transmit data using the RS-232 port.  
(Output the contents of memory in ASCII Intel hex format)
- # 7 Receive, validate, offset, and load data using the RS-232 port.  
(Input ASCII Intel hex format into memory)
- # 8 Edit RAM memory using a full screen Hexadecimal editor.  
(Display and/or modify the Interact memory)
- # 9 Jump program counter to any four byte hexadecimal address.  
(Start the CPU executing machine code at a selected address)

FULL DOCUMENTATION INCLUDED WITH SHIPMENT -

CASSETTE \$14.00  
+shipping \$1.00

NEW ADDRESS

SLAGH SYSTEM SERVICES BOX 53 DEARBORN, MI 48121  
NEW TELEPHONE NUMBER : 313-581-8593

\*\*\*\*\*

SABRE PORT

An RS232 level serial printer port for the Interact. Price includes all hardware, operating and installation instructions port driver program listing, and a BASIC Overlay program which overlays both Level II and 8K Graphics BASIC. All for just \$24.99 plus \$2.00 for shipping and handling from:

SABRE 1415 Creek Hollow Dr. Seabrook, TX 77586 (713)870-8315

\*\*\*\*\*



**\*\*\* INTERACT OPERATING SYSTEM UPDATE \*\*\***

GOOD NEWS!!! Recent changes in the INTERACT OPERATING SYSTEM (IOS) Monitor ROM software has made the IOS memory size independant! This means that no matter what your memory configuration is, the IOS can handle your needs.

For those of you who have not yet heard of the IOS, and even if you have, please read on! The IOS is an EPROM-based monitor for INTERACT computers with the following commands and features:

- 1) Memory dumps in hexadecimal or ASCII formats
- 2) Fill memory with any data byte (\$)
- 3) Move memory blocks up or down in location (\$)
- 4) Number base conversions - hex to decimal and decimal to hex
- 5) Hexadecimal sum and differences
- 6) Substitute memory with hex data (\$)
- 7) Disassemble memory - requires tape loaded data table placed anywhere in core
- 8) Reading and writing tape files
- 9) Tape motor control (rewind, fast forward)
- 10) Screen windowing and clearing
- 11) Memory testing over a user defined range
- 12) 8080 address reference locator
- 13) Memory range checksum for program verification
- 14) Display and/or change a user register set
- 15) Goto any address and execute using the user register set with any number of breakpoints placed into RAM based code
- 16) 60 cycle interrupt - user panic breakpoint or panic exit routine
- 17) IOS only software restart (no reset button needed to reset the IOS software)
- 18) Exit from IOS control to 'DEPRESS L TO LOAD TAPE' mode
- 19) 3 user command jumps to preset RAM addresses and a jump to 4C00 HEX, the start address of all INTERACT RAM software
- 20) 7 separate user selectable color tables in popular combinations
- 21) User control of characters being displayed on the screen, that is, any character which is to be put onto the screen may be manipulated by the user before it is displayed (handy for printing any screen text)
- 22) Initial power-up or reset options which allow any of the followings:
  - a) exiting IOS to 'DEPRESS L TO LOAD TAPE' mode without changing any critical RAM program memory
  - b) clearing all memory starting at the top of the screen to the end of addressable core
  - c) entering full IOS control

(Commands marked with an asterisk (\$) have an operation verification feature)

All of this software is available to you for the price of \$55.00, which includes a complete user's manual, a listing of the IOS and the original ROM in the INTERACT, and also the tape necessary for the disassembler command. Please include \$2.00 postage and handling charges.

**NO INTERACT SHOULD BE WITHOUT AN IOS!**

SEND TO:

RAF SOFTWARE ENTERPRISES  
 C/O RICHARD A. FERRIS  
 16415 FOREST BEND AVENUE  
 FRIENDSWOOD, TEXAS 77546



MTP-I MORSE TERMINAL PROGRAM (COPYRIGHT MARCH 1982 BY J. A. MILLER)

THIS MACHINE LANGUAGE PROGRAM CONVERTS THE INTERACT INTO A MORSE COMMUNICATIONS TERMINAL WITH TRANSMIT AND RECEIVE CAPABILITY. SIMPLY LOAD THE CASSETTE AND THE TERMINAL INITIALIZES READY TO RECEIVE OR TRANSMIT. A FIXED STATUS PARTITION ON THE SCREEN CONTINUOUSLY DISPLAYS PROGRAM STATUS (CODE SPEED, BUFFER DEPTH, ETC.). A FAST SCROLL PARTITION DISPLAYS TRANSMITTED (TYPED) AND RECEIVED CHARACTERS. TWO CHARACTER SIZES (3X5 OR 5X5) CAN BE SELECTED BY KEYBOARD COMMAND. THE 3X5 SIZE PROVIDES FOR 28 CHARACTERS PER LINE.

START TYPING AND THE UNIT BEGINS SENDING WITH PRECISE DOT, DASH, CHARACTER AND WORD SPACING. A SIDETONE IS PROVIDED VIA THE TV MONITOR SPEAKER. TRANSMIT SPEEDS (4-60 WPM) AND OTHER FUNCTIONS MAY BE SELECTED AT ANY TIME WITH CTRL KEYS. A 256 CHARACTER CIRCULAR BUFFER ALLOWS TYPE-AHEAD AND BACKSPACE CORRECTIONS WHILE THE UNIT SENDS FROM THE BUFFER AT THE SELECTED SPEED. A SOFTWARE "KEYBOARD DEBOUNCE" FUNCTION VIRTUALLY ELIMINATES ANNOYING DOUBLE HITS THAT PLAGUE INTERACT KEYBOARDS.

WHEN NOT TRANSMITTING, THE UNIT GOES INTO RECEIVE MODE, WHERE AN ADAPTIVE ALGORITHM WITH NOISE REDUCTION AUTOMATICALLY ADJUSTS TO RECEIVE SPEEDS FROM 8 TO 40+ WPM, RECOGNIZES MORSE DOT/DASH PATTERNS, AND DISPLAYS THE RECEIVED CHARACTERS. AUTOMATIC WORD WRAP IMPROVES SCREEN READABILITY BY REDUCING THE OCCURRENCE OF FRAGMENTED WORDS AT THE SCREEN EDGES. SUPPORT OF EITHER THE U80M OR MICROVIDEO RS-232 PORTS IS PROVIDED FOR HARDCOPY OUTPUT. (PLEASE SPECIFY TYPE.)

TRANSMITTER KEYING IS ACCOMPLISHED BY CONNECTING ONE OF THE TAPE HEAD WRITE LINES TO AN EXTERNAL KEYING TRANSISTOR AND/OR RELAY. MARK/SPACE LOGIC LEVELS FOR RECEPTION ARE INPUT TO THE LEFT JOYSTICK "HIT" BUTTON (A/D CONVERTER). A RELATIVELY SIMPLE OUTBOARD CIRCUIT IS REQUIRED TO FILTER AND DEMODULATE THE RECEIVED AUDIO TONES AND PRODUCE THE NECESSARY ON/OFF LEVELS FOR THIS INPUT (CIRCUIT DIAGRAM INCLUDED).

FOR CASSETTE, OPERATING INSTRUCTIONS AND INTERFACING PLANS, SEND \$23.00  
 JAMES A. MILLER, P. O. BOX 455, MELBOURNE, FL 32901.

TO

**George A. Leggett's MACHINE LANGUAGE SUBROUTINES**

Tape and documentation for Twenty-one Routines in all: ten of which appeared in the book 8080 FOR EVERYONE, and eleven of which have never before been released by me.

Send \$10.00 for MACHINE LANGUAGE SUBROUTINES  
 Or, send \$22.00 for MACHINE LANGUAGE PROGRAMMING PACKET which includes: The book, 8080 FOR EVERYONE, the X-Y PLOTTING MONITOR, & MACHINE LANGUAGE SUBROUTINES  
 8080 FOR EVERYONE \$15.00, X-Y PLOTTING MONITOR \$5.00

George A. Leggett 20562 Woodward Mt. Clemens, MI 48043

LEVEL II BASIC LOAN EVALUATION PROGRAM, SCREEN DISPLAY BUT CAN BE ALTERED FOR HARD COPY. COMPUTES 1) MONTHLY PAYMENT 2) SCHEDULE OF PAYMENTS 3) PAYOFF DATE AND AND PAYMENT 4) PORTION OF PAYMENT THAT IS INTEREST AND PRINCIPAL.

FOR CASSETTE AND LISTING SEND \$8.00 TO -  
 TERRY O'BRIEN 6500 LANGE RD. BIRCH RUN, MI 48415 517-777-5241



# INTERACTION

A NEWSLETTER FOR INTERACT OWNERS

---

JUNE - AUGUST, 1982

VOL. III, no. 3

---

## Leggett's MACHINE SHOP TALK

A lot has happened since our first Machine Shop Talk, but I'm back again with more shop talk. I'd like to begin by clearing up something I should have mentioned in the first article, and which will pertain to all future articles where Machine Language is being used as a USER Function with BASIC. To enter any of the Machine shop examples, I always use a ROM Monitor. You may use any Monitor you wish. When you work with BASIC and Machine Language you enter the Machine Language first via a Monitor. A question I received from one of the readers was, how did I go from a BASIC line number into a Machine Language routine. I first load the Machine Language program and save it on tape. Then I load BASIC, press Reset, and L. I load in the Machine Language program back into the computer, return to BASIC and enter the BASIC program. Those of you who have no Monitor will have to take a longer route to enter the Machine Language. You could poke the addresses from BASIC, but that is a pain in the neck. The best thing for those of you with no Monitor at all is to refer back to the BASIC TO MONITOR Program by Walter Hendrickson in INTERACTIONS Vol. 2 no. 3. That will get you out of it quite nicely. I hope we have covered the subject. If there are any further questions, please notify me.

Now, onward and upward. In this and the next article, I would like to discuss the SOUND and TONE Routines, which are quite familiar in BASIC. I shall attempt to show you that not only can you achieve the same sounds as BASIC, but far more flexibility than BASIC will allow. The easier of the two is the TONE Routine. As you may know if you have any ROM Listings or my book, 808C FOR EVERYONE, the TONE Routine resides in ROM 1. Load B and C with the tone number from 0 to FFFF. (I usually just load B with 0 and C with any number from 0 to FF giving 255 tones or a nice variety.) Then, load D and E with the duration. Thus, it works the same as BASIC. If we wanted to say TONE 64, 50 in Machine Language, we would load 40 Hex in C and 0 in B. Load E with 32 Hex, which equals 50 Decimal, and 00 in D. CALL 07BF. If you have entered the example and are saying "I can't tell any difference from BASIC", well, good! It works! So why use Machine Language. We will now use the TONE Routine to make a distinction as to why a Machine Language Routine even to be incorporated with BASIC, might be a more unique alternative. Now, if you were writing a BASIC Program and you were only using the TONE command to indicate a response or something simple, BASIC will do fine. But how about a sliding tone? You may feel as you look at the BASIC part that it is something you did the first night you brought home the machine, but it is only presented as a comparison. You will be shocked to hear the difference in the two interpretations of the same thing.



## Machine Shop Talk, cont.

As mentioned above, enter the Machine Language first. I chose the 5600 block of memory because it is out of the way of BASIC. You may put routines wherever you wish to suit your needs.

			SLIDING TONE
Address	HEX	Mnemonic	Description
5600	F3	DI	Disable Interrupts to make clear steady tone.
5601	01	LXI B	Load B and C
5602	01		
5603	00		Load B and C with Tone 1
5604	3E	MVI A	Move next 2 Bytes into A
5605	40		Loop count number (64 Decimal)
5606	11	LXI D	Load D and E
5607	40		
5608	00		Load DE with length of tone in loop.
5609	F5	PUSH PSW	Save Loop count number from A on Stack
560A	CD	CALL	
560B	BF		
560C	07		CALL 07BF TONE Routine in ROM 1
560D	F1	POP PSW	Bring loop Count number back into A from Stack
560E	03	INX B	Increase B and C, adding 1 to your tone.
560F	3D	DCR A	Decrease A by 1, Loop Count number.
5610	02	JNZ	Jump if not C to next 2 Bytes
5611	06		
5612	56		Jump to 5606 and carry out loop again.
5613	3E	MVI A	Move into A
5614	40		New Loop Count number
5615	11	LXI D	Load D and E
5616	40		
5617	00		Load DE with length of all tones in loop
5618	F5	PUSH PSW	Push loop number in A onto Stack
5619	CD	CALL	
561A	BF		
561B	07		CALL TONE Routine in ROM 1.
561C	F1	POP PSW	Bring loop count number back to A from Stack
561D	0B	DCX B	Decrease B and C by 1, Tone number.
561E	3D	DCR A	Decrease A by 1 or Loop Count
561F	02	JNZ	Jump if not C
5620	15		
5621	56		Jump to 5615 to carry out loop again
5622	FB	EI	Turn Interrupts back on.
5623	09	RET	Return to BASIC

## CORRECTION

the INNARDS OF BASIC, Part 2 April - May, 1982

The description of the routine NXTPRM should have included the statement

'The parameter is returned in A only.'

## Machine Shop Talk, cont.

## SLIDING TONE BASIC PROGRAM

Note: If using Level II BASIC, use Line 5. Any of the newer BASIC Language tapes begin at Line 10.

```
5 POKE 19215, 25
10 FOR X = 1 TO 64
20 TONE X, 64
30 NEXT
40 FOR X = 64 TO 1 STEP -1
50 TONE X, 64
60 NEXT
70 POKE 19473, 0
80 POKE 19474, 86
90 U =USR(0)
```

While this is not the shortest nor the best way to write this program, it does serve as the best way to understand it clearly. We could enter the program in just a couple BASIC lines, but it would not vary the timing. As you can hear, the BASIC tone is gravely and slower in comparison to the Machine Language, which gives a clearer sliding tone. If you were shocked to hear such sound coming from yours you are not alone. My wife and I were shocked too. Try on your own as an experiment to make the POT Control vary the tone. For once, we can finally have a clear smooth tone in the Interact.

## BETTER KEYBOARD RESPONSE for the MICRO-VIDEO COMMUNICATOR

by Robert M. Alpert  
1144 N. 35th St., Apt. 2  
Camden, NJ 08105  
COMPUERVE no. 70525,1213

Those of you who use the Micro Video 'communicator' program to access a time sharing service must have noticed how bouncy and unpredictable the keyboard is. This is very easily remedied with a patch.

After loading the video banner, the program loads 10 bytes at 4C00H and the main body of the program at 6000-6B00. Located at 64D3H is the following series of instructions:

Better Keyboard Response, cont.

(note: mnemonics as used in Hi-Lo Monitor)

```

64D3 CALL 07E7
64D6 RZ
64D7 MOVBA
64D8 LHLD 6276
64DB MOVMA
64DC INRL
64DD SHLD 6276
64E0 LDA 6281
64E3 ORAA
64E4 RZ
64E5 JMP 6610
64E8 NOP
64E9 NOP
64EA NOP
64EB NOP

```

....

The string of NOP's allows us to easily insert a call to a new user routine. Move the code from 64D7-64E5H up 3 bytes to 64DA-64EBH. Insert a CALL instruction at 64D7H. I put the routine at 4C10H. The result should look like this:

```

64D3 CALL 07E7
64D6 RZ
→ 64D7 CALL 4C10
64DA MOVBA
64DB LHLD 6276
... etc.

```

The following routine disables the interrupts and sends a tone to the TV speaker when each key is pressed. This greatly reduces key bounce and provides audio confirmation of each character sent. (This is especially handy when screen output is suppressed, such as typing in a Comuserve password.)

```

4C10 DI
4C11 PUSHHP
4C12 PUSHB
4C13 PUSHD
4C14 PUSHH
4C15 LXIB 0015
4C18 LXID 0060
4C1B CALL 07BF
4C1E POPH
4C1F POPD
4C20 POPB
4C21 POPPS
4C22 EI
4C23 RET

```

By setting up the appropriate tape output list, the much improved communicator program may be written to tape.

(For those who like the idea but lack the facilities to make this patch, I will provide an overlay tape for a nominal price. If interested see ad in this issue.)

NOTE - this article supplements the preceding one

### MICRO-VIDEO Communicator Program

(Add tone to the keystrokes and UPPER/lower case capabilities)

Joseph Kleczka  
357 Joya Loop  
Los Alamos, New Mexico 87544

The Upper/lower case is the easiest and only takes one location change. Replace the hex number 20 with 00 at location 6517. To add the tone to the keystroke takes a little more effort. First substitute this small program at location 4C10.

```

CD Call ASCII key entry routine
E0
07
F5 Push registers
C5
D5
E5
F3 Disable interrupts
01 Load the B C register pair with next two bytes
15
00
11 Load D E register pair with next two bytes
50
00
CD Call tone generator
BF
07
FB Enable interrupts
E1 Pop registers
D1
C1
F1
C9 Return to the calling program

```

Next substitute hex 23 for 20 at location 6514

At location 6520 substitute the following

```

CD Call a section of the routine we substituted at 4C10
13
4C
32 Original code
00
C0
C9

```

Substitute at locations 6399-639B, 63C6-63C8, 644D-644F, 662D-662F, and 66F5-66F7

```

CD Call our program
10
4C

```

## WORD SEARCH

by Stephen Cook

This program generates and prints out word search puzzles. It is a program for a 16K INTERACT (BS232 BASIC) and an Epson MX-80. However it is more suited to a 32K INTERACT. With a 16K machine limit your puzzle size to 20 by 20 and use less than 20 words or you will get Out of Memory or Out of String space errors constantly. Even with those limits you still may get occasional errors. For a 32K machine change line 10 to 10 CLEAR 2500 and you can make up to a 40 by 40 puzzle. The program takes a long time to fill in the empty spaces, so be patient. The program allows you to print out a solution and then makes 1 or more copies of the word list and puzzle on a sheet. It was written for use as a vocabulary exercise for my wife's students.

```

10 CLEAR 1000
20 REM -WORD SEARCH PUZZLE
30 INPUT "COLUMNS";C
40 INPUT "ROWS";R
50 INPUT "NO. OF WORDS";W
60 INPUT "NO. OF PUZZLES";N
70 DIM L$(C,R),W$(W),A$(26),G$(1)
80 FOR K = 1 TO 26
90 A$(K) = CHR$(K + 64)
100 NEXT K
110 FOR I = 1 TO C
120 FOR J = 1 TO R
130 L$(I,J) = "*"
140 NEXT J: NEXT I
150 FOR K = 1 TO W
160 PRINT "ENTER WORD": INPUT W$(K)
170 GOSUB 340
180 NEXT K
190 PRINT "SOLUTION (Y - N)?":G$ = INSTR$(1)
200 IF G$ < > "Y" THEN 220
210 GOSUB 250
220 GOSUB 990
230 LPRINT
240 N = N - 1
250 FOR J = 1 TO R
260 LPRINT TAB(40 - C);
270 FOR I = 1 TO C
280 LPRINT L$(I,J);" ";
290 NEXT I
300 LPRINT
310 NEXT J
320 IF N < 1 THEN END
330 GOTO 220
340 LO = LEN(W$(K))
350 P = 1
360 IF T < 100 THEN 390
370 PRINT "WORDS WON'T FIT!"
380 PRINT "TRY AGAIN": GOTO 10
390 P = 1:PO = 1:Q = 1

```

word Search, cont.

```

400 IF RND(1) < = .5 THEN 420
410 Q = 1
420 QO = - 1
430 IF RND(1) < = .5 THEN 450
440 QO = 1
450 D = 2
460 IF Q < > 1 THEN 480
470 P = 0
480 IF QO < > 1 THEN 500
490 PO = 0
500 IF RND(1) < .75 THEN 520
510 D = 1
520 IF RND(1) > .25 THEN 540
530 D = 0
540 CO = C:RO = R
550 IF D < > 1 THEN 570
560 RO = R - LO
570 IF D < > 0 THEN 590
580 CO = C - LO
590 IF D < = 1 THEN 610
600 RO = R - LO:CO = C - LO
610 IF CO < > C THEN 630
620 PO = 0
630 IF RO < > R THEN 650
640 P = 0
650 X1 = INT(( RND(1) * RO / 100 + .01) * 100 + P * LO)
660 X2 = INT(( RND(1) * CO / 100 + .01) * 100 + PO * LO)
670 IF D = 1 THEN 890
680 IF D = 0 THEN 790
690 FOR I = 1 TO LO
700 IF L$(X2 + (I - 1) * QO,X1 + (I - 1) * Q) = "*" THEN 720
710 IF L$(X2 + (I - 1) * QO,X1 + (I - 1) * Q) < > MID$(W$(K),I,1) THE
N 350
720 NEXT I
730 T = 0
740 FOR I = 1 TO LO - 1
750 L$(X2 + I * QO,X1 + I * Q) = MID$(W$(K),I + 1,1)
760 NEXT I
770 L$(X2,X1) = MID$(W$(K),1,1)
780 GOTO 980
790 FOR I = 1 TO LO
800 IF L$(X2 + (I - 1) * QO,X1) = "*" THEN 820
810 IF L$(X2 + (I - 1) * QO,X1) < > MID$(W$(K),I,1) THEN 350
820 NEXT I
830 T = 0
840 FOR I = 1 TO LO - 1
850 L$(X2 + I * QO,X1) = MID$(W$(K),I + 1,1)
860 NEXT I
870 L$(X2,X1) = MID$(W$(K),1,1)
880 GOTO 980
890 FOR I = 1 TO LO
900 IF L$(X2,X1 + (I - 1) * Q) = "*" THEN 920
910 IF L$(X2,X1 + (I - 1) * Q) < > MID$(W$(K),I,1) THEN 350

```

Word Search, cont.

```

920 NEXT I
930 T = 0
940 FOR I = 1 TO LO - 1
950 L$(X2,X1 + I * Q) = MID$(W$(K),I + 1,1)
960 NEXT I
970 L$(X2,X1) = MID$(W$(K),1,1)
980 RETURN
990 FOR I = 1 TO C: FOR J = 1 TO R
1000 IF L$(I,J) < > "*" THEN 1030
1010 Z1 = INT((RND(1) * .26 + .01) * 100)
1020 L$(I,J) = A$(Z1)
1030 NEXT J: NEXT I
1040 LPRINT CHR$(12)
1050 LPRINT CHR$(14);
1060 LPRINT TAB(16);"WORD LIST"
1070 FOR K = 1 TO W
1080 LPRINT TAB(35);W$(K)
1090 NEXT K
1100 RETURN

```

### IMPORTANT NOTICE

All listings in this issue use a special listing program and contain extraneous spaces. While they are more readable, please type them in without extra spaces to conserve memory. This is especially true for WORD SEARCH.

### PRODUCT REVIEWS

#### SKETCH PAD

David T. Schwab  
10 Jay Lee Court  
Ann Arbor, MI 48104

Last year I decided I'd like a banner to stick on tapes I send to people. To make it, I first used COMPUTE-A-COLOR to draw a picture and letters and then saved it on tape. I then used the HI-LO Monitor to make a copy that would clear the screen, set the color registers and draw the banner. This required a tape output list to create a banner properly that did not have a stop code.

I could have saved myself over an hour of work doing this, if I had had SKETCH PAD. The same banner (with 1982 instead of 1981) took me less than 15 minutes with Dave's new BASIC program. With it you can draw open circles, filled circles, lines, boxes and triangles and print characters which had to be individually drawn with COMPUTE-A-COLOR. Once finished you can make a tape copy which will load and stop or will load and then continue to load the program following your picture on the tape.

For \$8, this is a very good utility program to help make all your programs look and load professionally. It uses the keyboard to pick the type of figure and the joystick to place it on the screen.

Reviewed by Stephen Cook

## Product Reviews, cont.

## COLOR BAR PROGRAM by George Leggett

Reviewed by Jim Dinkey 3380 Cork Oak Way Palo Alto, CA 94303

This machine language program is very useful for aligning and working on TV sets. If you have never been in the back of a TV I suggest you don't start now, but the program has all of the characteristics that you need for aligning and adjusting the set. It is nice to be able to have a steady picture with which to work. It even helps out considerably on black & white. I'd like additional colors and identification, but still the program is quite an aid. I recommend it.

## Product Review: COMET 8300R Printer

By Jerry Goerz

KY Utilities 1 Quality St. Lexington, KY 40507

When the wholesale price of the COMET 8300R dropped to \$225 in December 1981, I could no longer resist. The COMET is a rugged dot-matrix tractor-feed line printer, with 22 DIP-switches to set such parameters as Baud Rate (110 to 9600), line length (80 or 132), page length, bi-directional printing, word format, plus one option which will really delight the engineer: the slash through a zero!

The COMET is about the size and weight of a record turntable, and arrives carefully packed and tested. It takes forms up to 10 inches wide with two carbons, and comes with a purple ribbon, but a black ribbon is available. Unfortunately, the user's manual does not provide pinout data for the DB25 connector, but this information is available in a service manual. I'll save you about \$35 by revealing that pin 3 is Received Data, 7 is Signal Ground, and 20 is Data Terminal Ready which tells the computer that the printer is on, the line is selected, the paper is in the printer, and the buffer is empty.

The COMET is a line printer, which means that characters are stored in a buffer until a carriage-return is sent to signify the end of a line. The print head then prints the line and continues to traverse the paper, coming to rest at the opposite side. The printer can print bi-directionally, which increases speed from 50 to 63 lines per minute. The paper can be advanced forward manually, but can be moved backward only by releasing the paper from the tractor feed.



Product Reviews, cont.

To access the DIP-switches, the DB25 plug must be removed, something soft placed under the rear of the printer to protect the table top from being marred by the protruding hinges, and the printer reclined on its rear. A plastic panel is then removed from the underside, revealing a surprisingly large PC board, with 24 DIP switches, of which only 22 have any function, and many of those don't matter if you are using a simple port such as the SABRE. The switch positions which match the SABRE default conditions (1200 baud, transmitted automatic line feed on CR, 80 characters per line, 58 lines per page, 2 stop bits, 8 data bits, no parity) are as follows, from left to right beginning with SW1-1 and ending with SW5-4:

X C O X O C O X O C C O O C C C O C O O X O O X

Where X is don't care, C is Closed, O is Open. Remember to reset the SABRE line counter with CONTROL-R, whenever aligning a new page in the printer.

The Comet 8300R is marketed in the U. S. by C. Itoh Electronics, Inc., 280 Park Avenue, New York, N. Y. 10017



# THE EDU-BASIC OVERLAY

Now you can turn Edu-BASIC into a powerful programming language. With the Edu-BASIC Overlay, you can have such features as

- \*A real, live cursor (like other computers have!)
- \*Printing text anywhere on the screen, in any color
- \*Built-in printer commands for Slagh Systems printer interface
- \*PEEK, POKE, USR, and three new machine language commands
- \*A greatly expanded vector
- \*Simple adaption to memory expansions
- \*Of course, 8K of program memory

The Edu-BASIC Overlay comes with 50 pages of documentation (this alone is worth the price!) which fully explain the 14 new functions and commands, and include a memory map, subroutine list, and other information to help YOU customize YOUR OWN version of Edu-BASIC.

To purchase, send \$15.00 to  
R.P. Williams  
6710 Virgilian St.  
New Orleans, LA 70126

For more information call  
(504) 242-6961 between 6 and  
8 P.M. CST.

EDITING FORTH

by Russ Schnapp 8062 Gold Coast San Diego, CA 92126

In my previous article on FORTH, I alluded to the difficulty of the editing task. In this article, I shall discuss some FORTH editing techniques, and some enhancements.

First of all, for those of you struggling with the editor, let me introduce you to the simple (read: minimal functionality) line editor. In order to gain access to the editor vocabulary, one invokes the word EDITOR. The single most used editor word is P. This word places text following it onto one of the sixteen lines (numbered 0 through 15) in the edit screen. For example, lets enter some word definitions into the edit screen:

```
EDITOR(cr) OK
0 P ( UTILITY WORDS; C?, DUMP; RLS, 5/20/82 ) (cr) OK
1 P (cr) OK
2 P : C? C^ 2 .R ; (cr) OK
3 P (cr) OK
4 P : DUMP4 HEX DUP 0 4 D.R ." : " (cr) OK
5 P 4 0 DO DUP C? 1+ LOOP DROP ; (cr) OK
```

We can now save this screen onto tape by loading a cassette, pressing WRITE/READ, and typing PUTSCR(cr). We can display a line by using the word, T:

```
2 T (cr)
_ : C? C^ 2 .R ; 2 OK
```

The entire edit screen can be listed, using the word L. L displays two lines and then waits for a keystroke; If this key is not a (cr), the next two lines are displayed, otherwise the screen listing is terminated. The listing is terminated after line 15 in any event.

In addition to adding and displaying lines, we can edit the screen to some degree. A line can be deleted, and subsequent lines moved up, by using D:

```
2 T 3 T (cr)
_ : C? C^ 2 .R ; 2
3 OK
2 D 2 D 2 T 3 T (cr)
_ : DUMP4 HEX DUP 0 4 D.R ." : " 2
_ 4 0 DO DUP C? 1+ LOOP DROP ; 3 OK
```

The text of the most recently Deleted, Put, or Typed line is recorded in PAD (as in scratch-), for use by words I and R. The word " also records text following it at PAD. R replaces the indicated line with the text at PAD. I inserts the text in PAD at the indicated line, after shifting that line, and following lines, down by one:

```
" : C? C^ . ; " 2 I (cr) OK
" " 3 I (cr) OK
```

## Editing FORTH (continued)

You now know how to edit a screen. The real problem arises when you try to replace an edited screen onto a cassette. This is trivial when there is only one screen on the cassette--you merely REWIND, and then PUTSCR. However, when there are many screens on the cassette, there are two ways to go about this:

- 1) Use one tape as "old" and the another as "new." Copy all screens preceding the one to be modified, from "old" to "new." Copy the edited screen to "new," copy the remaining screens from "old" to "new." This is a slow, arduous task, but it is a relatively safe method, as long as you don't confuse the "old" and "new" cassettes.
- 2) After reading the screen to be edited, key in REWIND, and alternate on the REWIND and READ keys on the cassette drive until you hear the end of the screen previous to the edited one. Stop the REWIND, edit the screen, and PUTSCR it over the old copy. This is a relatively quick, easy, and DANGEROUS technique (though I used to use it exclusively).

At the beginning of this article, I promised an editor enhancement: By adding extra screen buffers, and a few words to manipulate them, we can mitigate the multiscreen editing problem. The new words are:

- PUT :     n ---  
Take the edit screen, and place it in buffer number n (buffers are numbered from 0)
- GET :     n ---  
Copy buffer number n to the edit screen.
- THIS :     --- n  
Return the number of the buffer to which a GET or PUT has most recently been applied.
- NEXT :     --- n  
Return THIS + 1
- GETN :    n ---  
Read n screens from the cassette into buffers 0 thru n-1.
- PUTN :    n ---  
Write n screens from buffers 0 thru n-1 to the cassette.
- LOAD :    ---  
Begin compiling from buffer 0. The definition of --> has been redefined to skip to the next buffer, when in LOAD mode.

## Editing FORTH (continued)

In the accompanying listings, the constant HIBUF is set to the number of buffers to be allocated. On 16K systems, HIBUF can't be set to more than 4, or so. On Interacts with 32K or more, and having FORTHS with serial numbers > 14, HIBUF can be set to 10 or more (By the way, anyone having FORTH serials < 15 may exchange it for the latest release, if they send \$3.00, and agree to return their old copy upon receipt. Pre-serial 15 tapes don't automatically use all available memory).

```

_( MULTI-BUFFERS, 1 OF 2; ?BUFNUM,PUT; RLS, 5/18/82 )      0
_FORTH DEFINITIONS DECIMAL                                1
_ 9 CONSTANT HIBUF ( NUMBER OF SCREEN BUFFERS-1 )        2
_0 VARIABLE PREVIOUS ( MOST RECENTLY ACCESSED SCREEN BUFFER ) 3
_0 VARIABLE SCREEN_BUFFERS 1024 HIBUF 1+ * 2 - ALLOT 0 C,  4
_: ?BUFNUM ( N->N; IF N>HIBUF ADVISE AND QUIT )           5
_ DUP HIBUF > IF DROP ." BUF NUMBER TOO HI" DING CR QUIT ENDIF ; 6
_: GET ( N->; MOVE BUFFER N TO EDIT SCREEN )              7
_ ?BUFNUM UPDATED C^                                     8
_ IF DING ." PUT OR DISCARD!" DROP                        9
_ ELSE DUP 1024 * SCREEN_BUFFERS + FIRST 1024 CMOVE     10
_ 0 UPDATED C! PREVIOUS ! ENDIF ;                        11
_: PUT ( N->; MOVE EDIT SCREEN TO BUFFER N )              12
_ ?BUFNUM 0 UPDATED C! DUP PREVIOUS ! 1024 * SCREEN_BUFFERS 13
_ + FIRST SWAP 1024 CMOVE ;                               14
--->                                                      15

_( MULTI-BUFS, 2 OF 2; THIS,NEXT,PUTN,GETN,LOADING,LOAD,--> ) 0
_: THIS PREVIOUS ^ ;                                     1
_: NEXT THIS 1+ ;                                       2
_: PUTN 0 DO I GET PUTSCR LOOP ;                          3
_: GETN 0 DO GETSCR I PUT LOOP ;                          4
_0 VARIABLE LOADING ( LOADING=1 ONLY DURING "LOAD" )     5
_: --> ( OVERLOAD --> DEFINITION )                       6
_ ?LOADING 0 IN !                                         7
_ LOADING ^                                               8
_ IF NEXT GET ELSE GETSCR ENDIF ;                          9
_: LOAD ( ->; START COMPILING FROM BUF 0--WIPES OUT EDIT SCR ) 10
_ 1 LOADING ! 0 GET SLOAD 0 LOADING ! ;                  11
_12
_13
_14
_15

```

The described enhancement allows you to read in a set of screens, edit, recompile, and debug them; and then write them out. This is a considerable improvement, you would probably agree.

Any comments on FORTH or this article are welcome. Next article: FORTH I/O.

## LITTLE LETTERS / LITTLE NUMBERS

by Alvy Albert 1704 Cadillac Circle S. Melbourne, FL 32935

Here's one solution to getting more characters per line on an Interact. While this BASIC program does allow you to display 26 characters on a line, the program itself takes almost 3600 bytes leaving little in a regular INTERACT. However with a 32K INTERACT, the program could prove more useful. Also a second program is given which displays little numbers only which may be useful for large amounts of numerical data. It takes only 900 bytes of memory.

## LIST

```

1 POKE 19215,25
2 CLS :X = 18465: GOTO 100
10 CLS :X = 18465
11 A$ = INSTR$(1)
12 IF A$ = CHR$(8) THEN X = X - 1:PA = 0:PB = 0:PC = 0:PD = 0:PE = 0:
GOTO 82
13 IF A$ = "^" THEN PRINT :X = 18465: RETURN
14 IF A$ = " " THEN PA = 0:PB = 0:PC = 0:PD = 0:PE = 0: GOTO 82
15 IF A$ = CHR$(13) THEN PRINT : STOP
16 Q = ASC(A$) - 32
17 REM
18 REM
19 ON Q GOTO 22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,4
1,42
20 ON Q - 21 GOTO 43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60
,61,62
21 ON Q - 41 GOTO 63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79
22 PA = 4:PB = 4:PC = 4:PD = 0:PE = 4: GOTO 80
23 PA = 17:PB = 17:PC = 0:PD = 0:PE = 0: GOTO 80
24 PA = 17:PB = 21:PC = 17:PD = 21:PE = 17: GOTO 80
25 PA = 21:PB = 5:PC = 21:PD = 20:PE = 21: GOTO 80
26 PA = 1:PB = 16:PC = 4:PD = 1:PE = 16: GOTO 80
27 PA = 21:PB = 17:PC = 4:PD = 17:PE = 21: GOTO 80
28 PA = 16:PB = 4:PC = 0:PD = 0:PE = 0: GOTO 80
29 PA = 4:PB = 1:PC = 1:PD = 1:PE = 4: GOTO 80
30 PA = 4:PB = 16:PC = 16:PD = 16:PE = 4: GOTO 80
31 PA = 0:PB = 17:PC = 4:PD = 17:PE = 0: GOTO 80
32 PA = 0:PB = 4:PC = 21:PD = 4:PE = 0: GOTO 80
33 PA = 0:PB = 0:PC = 0:PD = 16:PE = 4: GOTO 80
34 PA = 0:PB = 0:PC = 21:PD = 0:PE = 0: GOTO 80
35 PA = 0:PB = 0:PC = 0:PD = 0:PE = 4: GOTO 80
36 PA = 0:PB = 16:PC = 4:PD = 1:PE = 0: GOTO 80
37 PA = 21:PB = 17:PC = 17:PD = 17:PE = 21: GOTO 80
38 PA = 4:PB = 4:PC = 4:PD = 4:PE = 4: GOTO 80
39 PA = 21:PB = 16:PC = 21:PD = 1:PE = 21: GOTO 80
40 PA = 21:PB = 16:PC = 20:PD = 16:PE = 21: GOTO 80
41 PA = 17:PB = 17:PC = 21:PD = 16:PE = 16: GOTO 80
42 PA = 21:PB = 1:PC = 21:PD = 16:PE = 21: GOTO 80
43 PA = 1:PB = 1:PC = 21:PD = 17:PE = 21: GOTO 80
44 PA = 21:PB = 16:PC = 16:PD = 16:PE = 16: GOTO 80
45 PA = 21:PB = 17:PC = 21:PD = 17:PE = 21: GOTO 80
46 PA = 21:PB = 17:PC = 21:PD = 16:PE = 16: GOTO 80
47 PA = 0:PB = 4:PC = 0:PD = 4:PE = 0: GOTO 80

```

Little Letters, cont.

```

48 PA = 0:PB = 4:PC = 0:PD = 4:PE = 1: GOTO 80
49 PA = 16:PB = 4:PC = 1:PD = 4:PE = 16: GOTO 80
50 PA = 0:PB = 21:PC = 0:PD = 21:PE = 0: GOTO 80
51 PA = 1:PB = 4:PC = 16:PD = 4:PE = 1: GOTO 80
52 PA = 21:PB = 16:PC = 20:PD = 0:PE = 4: GOTO 80
53 PA = 21:PB = 17:PC = 16:PD = 17:PE = 21: GOTO 80
54 PA = 21:PB = 17:PC = 21:PD = 17:PE = 17: GOTO 80
55 PA = 21:PB = 17:PC = 5:PD = 17:PE = 21: GOTO 80
56 PA = 21:PB = 1:PC = 1:PD = 1:PE = 21: GOTO 80
57 PA = 5:PB = 17:PC = 17:PD = 17:PE = 5: GOTO 80
58 PA = 21:PB = 1:PC = 5:PD = 1:PE = 21: GOTO 80
59 PA = 21:PB = 1:PC = 5:PD = 1:PE = 1: GOTO 80
60 PA = 21:PB = 1:PC = 17:PD = 17:PE = 21: GOTO 80
61 PA = 17:PB = 17:PC = 21:PD = 17:PE = 17: GOTO 80
62 PA = 4:PB = 4:PC = 4:PD = 4:PE = 4: GOTO 80
63 PA = 16:PB = 16:PC = 16:PD = 17:PE = 21: GOTO 80
64 PA = 17:PB = 17:PC = 5:PD = 17:PE = 17: GOTO 80
65 PA = 1:PB = 1:PC = 1:PD = 1:PE = 21: GOTO 80
66 PA = 17:PB = 21:PC = 21:PD = 17:PE = 17: GOTO 80
67 PA = 17:PB = 21:PC = 21:PD = 21:PE = 17: GOTO 80
68 PA = 21:PB = 17:PC = 17:PD = 17:PE = 21: GOTO 80
69 PA = 21:PB = 17:PC = 21:PD = 1:PE = 1: GOTO 80
70 PA = 21:PB = 17:PC = 17:PD = 21:PE = 16: GOTO 80
71 PA = 21:PB = 17:PC = 5:PD = 17:PE = 17: GOTO 80
72 PA = 21:PB = 1:PC = 21:PD = 16:PE = 21: GOTO 80
73 PA = 21:PB = 4:PC = 4:PD = 4:PE = 4: GOTO 80
74 PA = 17:PB = 17:PC = 17:PD = 17:PE = 21: GOTO 80
75 PA = 17:PB = 17:PC = 17:PD = 17:PE = 4: GOTO 80
76 PA = 17:PB = 17:PC = 21:PD = 21:PE = 17: GOTO 80
77 PA = 17:PB = 17:PC = 4:PD = 17:PE = 17: GOTO 80
78 PA = 17:PB = 17:PC = 4:PD = 4:PE = 4: GOTO 80
79 PA = 21:PB = 16:PC = 4:PD = 1:PE = 21: GOTO 80
80 REM
81 IF X > 18491 THEN PRINT :X = 18465
82 POKE X,PA: POKE X + 32,PB: POKE X + 64,PC: POKE X + 96,PD: POKE X +
128,PE
83 IF A$ = CHR$(8) THEN X = X - 1
84 X = X + 1
85 IF X > 18491 THEN PRINT :X = 18465
86 IF X > 18492 THEN PRINT :X = 18465
87 RETURN
100 READ B$
110 IF B$ = "0" THEN PRINT : FOR P = 1 TO 2000: NEXT : CLS : GOTO 100
115 IF B$ = "END" THEN END
120 FOR G = 1 TO LEN(B$)
130 A$ = MID$(B$,G,1)
140 GOSUB 12
150 NEXT
160 GOTO 100
200 DATA INTRODUCING LITTLE LETTERS^FOR THE INTERACT COMPUTER!^
205 DATA ^
210 DATA ABCDEFGHIJKLMNOPQRSTUVWXYZ
215 DATA ^^1 2 3 4 5 6 7 8 9 0^^
216 DATA ", ", ". ? ; * / = - + ,": ", ' $ % ! ( ) < >
220 DATA 0,^

```

Little Letters, cont.

```

250 DATA 0," ",PROGRAM BY^^," ",ALVY L. ALBERT^
255 DATA " ",1704 CADILLAC CR. S.^," ",MELBOURNE," ",FL.
260 DATA ^ 32935,0,^
270 DATA YOU MAY NOW ENTER YOUR OWN^DATA STARTING AT LINE 200.^^
275 DATA TO SCROLL UP ONE LINE ENTER AN EXPONENT IN THE NEXT^DATA LINE.^
280 DATA REMEMBER TO PUT COMMAS^INSIDE QUOTES.^^
290 DATA TO CLEAR THE SCREEN ENTER A ZERO IN THE NEXT DATA LINE.^^
295 DATA REMEMBER TO MAKE THE LAST^DATA LINE -END- TO ^
296 DATA TERMINATE THE PROGRAM^NEATLY AND RETURN COMMAND^TO THE KEYBOARD.^
297 DATA ^,0
300 DATA GOOD LUCK AND I HOPE THE ^PROGRAM IS USEFUL TO YOU!,0
310 DATA END

```

### LITTLE NUMBERS

```

1 POKE 19215,25
2 CLS :X = 18465
10 GOTO 50
11 CLS :X = 18465
12 A$ = INSTR$(1)
13 IF A$ = CHR$(8) THEN X = X - 1:PA = 0:PB = 0:PC = 0:PD = 0:PE = 0:
GOTO 33
14 IF A$ = "^" THEN PRINT :X = 18465: RETURN
15 IF A$ = " " THEN PA = 0:PB = 0:PC = 0:PD = 0:PE = 0: GOTO 33
16 IF A$ = CHR$(13) THEN PRINT : STOP
17 IF A$ = CHR$(9) THEN 11
18 REM
19 Q = ASC(A$) - 47
20 ON Q GOTO 21,22,23,24,25,26,27,28,29,30
21 PA = 21:PB = 17:PC = 17:PD = 17:PE = 21: GOTO 31
22 PA = 4:PB = 4:PC = 4:PD = 4:PE = 4: GOTO 31
23 PA = 21:PB = 16:PC = 21:PD = 1:PE = 21: GOTO 31
24 PA = 21:PB = 16:PC = 20:PD = 16:PE = 21: GOTO 31
25 PA = 17:PB = 17:PC = 21:PD = 16:PE = 16: GOTO 31
26 PA = 21:PB = 1:PC = 21:PD = 16:PE = 21: GOTO 31
27 PA = 1:PB = 1:PC = 21:PD = 17:PE = 21: GOTO 31
28 PA = 21:PB = 16:PC = 16:PD = 16:PE = 16: GOTO 31
29 PA = 21:PB = 17:PC = 21:PD = 17:PE = 21: GOTO 31
30 PA = 21:PB = 17:PC = 21:PD = 16:PE = 16: GOTO 31
31 REM
32 IF X > 18491 THEN PRINT :X = 18465
33 POKE X,PA: POKE X + 32,PB: POKE X + 64,PC: POKE X + 96,PD: POKE X +
128,PE
34 IF A$ = CHR$(8) THEN X = X - 1
35 X = X + 1
36 IF X > 18491 THEN PRINT :X = 18465
37 IF X > 18492 THEN PRINT :X = 18465
38 RETURN
39 READ B$
40 IF B$ = "END" THEN PRINT : PRINT : END
45 IF B$ = "0" THEN 2
50 FOR S = 1 TO LEN(B$)
51 A$ = MID$(B$,S,1)
52 GOSUB 13: NEXT : GOTO 50

```

## CSAVE BASIC AND PROGRAMS

by Dave Schwab 10 Jay Lee Court Ann Arbor MI 48104

I liked the file management system for BASIC programs that was in the newsletter. However, I prefer, and some others might too, to save BASIC itself along with my programs so that I only have to load one tape. This can be done by modifying BASIC's tape output list before you issue a CSAVE. The POKES to save 4A00-5FA0 and 6000-7FFF on tape are:

Decimal	(Hex)	To restore normal CSAVE POKE _____,
POKE 19556,0	(00)	90
19557,74	(4A)	76
19558,160	(A0)	2
19559,21	(15)	0
19560,0	(00)	90
19561,74	(4A)	76
19563,0	(00)	253
19564,96	(60)	76
19565,0	(00)	2
19566,32	(20)	0
19567,0	(00)	253
19568,96	(60)	76
19569,253	(FD)	255

I use SKETCH PAD to generate a Banner file without a stop code, and then save my BASIC programs like this. I find it very convenient to be able to switch from machine language programs to BASIC programs without having to load the BASIC interpreter all the time.

## INTERACTION STORAGE

Robert H. Murphy 109 Poplar Street Conneaut, OH 44030

One of my main problems with Interaction is where to put it (I can here the snickers now) but seriously, I did not want to punch holes in them for a notebook as this is inconveinent and risks tearing and mutilating it. I found a solution, a special notebook called and Accugrip. In-stead of punches it has a spring loaded bar on one side and all you do is line it up and push the bar down and it is held into place. It will hold all lose material and is ideal for keeping the Interactions whole and in order.

I would like to find out if any Interact owners have created a version of Dungons and Dragons. Now I know it can be done as I know somebody with a little 1K Sinclair who made a version of it so with the 4½K the Interact has it should be able to handle it. If you want to sell or trade please contact me.



BACK TO BASIC?

by George Leggett 20562 Woodward Mt. Clemens, MI 48043

The first half of 1982 has been quite a busy one in many ways. With our second child coming this fall and my ever-expanding computer and electronics and now wood work, we needed to move to a house that could handle it all! For those of you who write me, please make note of my new address so I receive all your mail quickly. I am sorry for any delays earlier in the year in getting materials sent out. Moving has been hectic, but things have leveled out now somewhat, and things are going out fine now.

I have gone into some new endeavors with the VIC-20 computer. However, have no fear. I am not leaving Interact. Interact is still my first love, and for now my only love, and if that sounds insane, it is not. For those of you who have purchased my book, 8080 FOR EVERYONE and MACHINE LANGUAGE RAM ROUTINES I am now on the side you were in trying to learn a new CPU, the 6502. This is coming along slowly, and although the VIC is a great machine and I might even recommend it as a good second choice in my opinion, the 8080 is like riding in a Mark VI vs. the 6502 being a 1960 VW. What more can I say, but for machine language programming I still maintain the Interact is the best machine around.

You may be shocked because I have said in the past that I do not do any BASIC in the Interact. Since I've had the VIC, BASIC is all I do! The luxury of screen editing has made programming in BASIC really enjoyable in the VIC. Perhaps one of you out there could come up with a screen editing program for the Interact. An article like my latest Machine Shop Talk would be nothing big for the VIC, since the VIC makes sliding tones like that naturally in a FOR Loop.

Although my 6502 Machine Language is coming along, all of my software in the VIC-20 has been written in BASIC, which doesn't make a lot of difference, since BASIC is in ROM, and there is no need to load a language tape. I feel that the Interact is the best machine for what it does that the 8080 ever saw, and that compared to the 6502, the 8080 is vastly superior.

\*\*\*\*\*  
 PLANET CONQUEST - As the commander of a space shuttle, your mission is to defeat Martians and colonize Mars. For 1 or 2 players. Joysticks needed. Specify Level II Or Fastline BASIC Version.

SLOT MACHINE - Why buy a slot machine when you can buy this fantastic program? 1 joystick is needed.

METRIC CONVERSIONS - Since the U.S. is switching over to the Metric System this program is a must! Does 40 Conversions.

The programs require Level II, and they are \$4.50 each or \$10.00 for all three plus plans to show a novice how to build a multi-alarm system for under \$30. Contact: (302) 658-0516 after 3pm EST

David K. Nichols 509 East 5th St. Wilmington, DE 19801

\*\*\*\*\*  
Announcing a new utility, the

UNIVERSAL PRINTER OVERLAY

This program adds a hardcopy capability to either Level II or to BK Graphics BASIC for users whose Interacts have RS232 ports. Both the Slagh and Microvideo ports are supported and the printer overlay is compatible with both the Leonardo and Fastline graphics overlays. This compatibility is attained at a small cost in programming space. (Less than 500 bytes.)

Provision is made for the user to write to tape a copy of the enhanced BASIC so that subsequent loads may be of a single cassette.

Available from Harry Holloway, PO Box 2263, Ann Arbor, MI 48106. Price \$12.50 postpaid within the US or Canada. MI residents please add \$0.50 sales tax.

\*\*\*\*\*

16K INTERACT FOR SALE includes raised keyboard, auxiliary commercial keyboard reconfigured for the Interact, 2 controllers, RF TV adapter, dust cover, PLUS 60 programs including Packrat & Alien Invaders, also complete financial package, games & 8K Fast Graphics BASIC, Monitor PLUS Complete Documentation, all newsletter issues, "Basically Speaking" Manual, catalogs etc. Total worth much more than sacrifice asking \$550 complete Call: B. Shumaker (313) 661-2232 after 6pm

\*\*\*\*\*

FORTH FOR THE INTERACT

This is an adaptation of the F.I.G. standard Forth for Interact computers. It comes complete with an assembler, a graphics package, and a line editor. Forth is faster and more powerful than BASIC.

Included is documentation of the differences from, and additions to the F.I.G. standard. A Forth manual is NOT included (I suggest "Starting Forth," by Leo Brodie, available from the Forth Interest Group, POB 1105, San Carlos, CA 94070).

By the way, Forth now makes use of all memory available on your Interact, whether you have 16K, 32K, or even 48K bytes.

For your copy of Forth for the Interact, send \$12 to:  
R. Schnapp, 8062 Gold Coast Drive, San Diego, CA 92126

- Also available are:
- A super-fast LIFE program, using the left joystick.... \$6
- A smooth PING-pong program, uses both joysticks..... \$6

\*\*\*\*\*

ADVENTURE CLUB. After you have fully completed an adventure, it's not much fun to go through it again. You're ready for the next adventure. For only \$25 per year, you can have 6 new adventure games. One will be mailed to you about every 60 days. Most will be in Level II BASIC but at least one per year will be full length (10K+) in machine language. Please note that these are standard non-graphic adventure games. If you have never played an adventure game before, we suggest you order Enchanted Adventure from our regular program list first. Send for a list of our other programs.

Richard Jones, Route 2, Box 191, Cole Camp, MO 65325

### The SABRE Port

Still available for only \$24.99 plus \$2.00 for shipping. See last INTERACTON Newsletter for details.

Now available from SABRE -- The ML4 Word Processor, Full cursor control, 18 commands for text handling, upper and lower case handling capability, reasonably priced, a true machine language text processor.

ML4A - For IOS equipped Interacts - \$9.00

ML4B - Non IOS equipped Interacts - \$10.00

Requires the SABRE Port. Please include \$1.00 for shipping.

SABRE, 1415 Creek Hollow Dr., Seabrook, TX, 77586, (713)498-3038

### MACHINE LANGUAGE RAM ROUTINES

Tape and book includes all routines from 8080 FOR EVERYONE plus fifteen new routines. One of these is a full WRITE Routine with notes and updates since the release of the first Machine Language book. Send \$10.00.

8080 FOR EVERYONE \$15.00

Programmer Pack, including 8080 FOR EVERYONE, MACHINE LANGUAGE RAM ROUTINES and X - Y PLOTTING MONITOR \$22.00.

For a free product list including materials for the Interact and new software for the VIC-20 computer, send a self-addressed stamped envelope to:

George Leggett 20562 Woodward Mt. Clemens, MI 48043

\*\*\*\*\*  
COMMUNICATOR  
overlay...

Greatly improves keyboard response  
of Micro-Video 'Communicator' program.

\$3.00-- if you send blank tape

\$5.00--if I provide tape

(includes postage)

Robert M. Alpert

1144 N. 35th St., Apt. 2

Camden, NJ 08105

\*\*\*\*\*

VENTURE!!! Not just any ADVENTURE, but now in full-color graphics.  
8K Graphics BASIC is needed. First, voyage into THE FORGOTTEN  
CAVE, and uncover Ziliad's most hidden secrets, but watch out for  
tricks, traps, and ancient monsters who keep his lair wellguarded.

Then, enter BLACK FOREST, where you must find the missing  
doctor who has the only cure for your disease, but you better be  
quick! You only have three days to find him before...

\$5.00 for both games to:

an Carey

58 Algonquin Trail

Medford Lks., NJ 08055

\*\*\*\*\*

ETCH PAD - BASIC program with extensive machine language subroutines for  
creating, modifying, and saving screen displays. Draws open and filled  
circles, rectangles, lines, and letters with super-fast joystick posi-  
tioning. Saves screen on tape with or without stop code (to create pro-  
gram banners). Hours of fun for all ages.....\$8.00

DISASSEMBLER in BASIC and QUEST in EDU-BASIC still.....\$5.00 each\*

David J. Schwab 10 Jay Lee Court Ann Arbor, MI 48104

\*\*\*\*\*

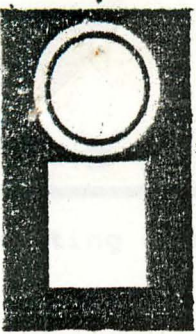
URED.. Maximum of \$75 labor (plus parts) most are less

ES 3380 Cork Oak Way Palo Alto, CA 94303

venture style game that lets you use  
your slave to find trinkets and  
could become filthy rich, rich, or  
depends upon your ingenuity.  
steal actually) at \$5.95



3 Big Lake Rd. Davisburg, MI 48019



*Laird  
from  
Laird*

# INTERACTION

A NEWSLETTER FOR INTERACT OWNERS

SEPT. - OCT., 1982

VOL. III, no. 4

## PARTING OF THE WAYS

As the year has progressed, I have found it increasingly difficult to keep up with the work on INTERACTION. Because there are so many (though only 300 paid subscribers this year) who still need the support of a newsletter, including those whose projects would never be considered for sale by MicroVideo, I have decided to give the newsletter over completely to George Leggett. One of my stipulations to George was that he would continue the newsletter for at least another year. George had no difficulty in agreeing to this and believes that there are many more years of usefulness for the Interact.

Unless you support George and the newsletter, many projects and ideas may wither. Without the newsletter you would have never seen the HILO Monitor or The Monitor ROMs. Russ Schnapp's FORTH rejected by MicroVideo because they did not know how to use it or support it would have never been brought to market. Because MicroVideo has a memory expansion, Walt Jopke would be wasting his time developing a 32K memory system, S-100 conversion and a floppy disk system without a newsletter to promote this items. The same goes for anyone who even sells one program through the newsletter.

I would like to thank everyone who has contributed over the years and all of the unprinted programs, articles and ideas will be turned to George for his consideration for publication. Please keep the flow of information still incoming, so INTERACTION will continue to be a success.

## PRINTING THE INTERACT SCREEN

Harry Holloway, PO Box 2263, Ann Arbor, MI 48016

During the past year I have had many enquiries about the use of the Epson MX80 graphics features. The result is these notes, which describe two techniques for getting a printout of the Interact's video display. The examples are set up for the MX80, but they could be reworked for several other printers by changing the control codes. The program examples are in BASIC. This gives a rather slow printout. (About 8 minutes for the first method and about 12 minutes for the second.) Machine code would be faster, but less accessible to the average user.

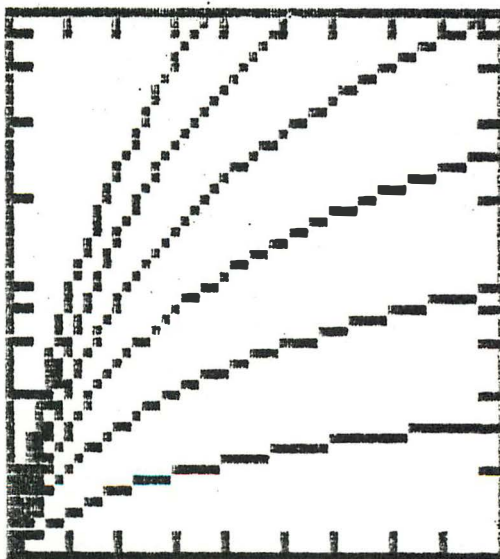
Printing the INTERACT screen, cont.

I assume that the reader has a serial output port and a BASIC with suitable printer drivers (ie. Level II or 8K plus a printer overlay or RS232 BASIC). The RS232 port should be set up to transmit 8 data bits (the printer overlay should have documentation that describes how to do this either with an explicit command or with a POKE) and the serial board in the MX80 should be set to pass all 8 data bits. The method for this is different for different serial boards and it should be described in the documentation. My early 2K buffered serial board has an 8-fold dipswitch with switch 4 passing the high-order bit when off and zeroing it when on. (CAUTION! The documentation that I received from Epson had all of the switch positions reversed from their actual values.) A later version of the same board has two dipswitches and accurate documentation.

Suppose that BASIC and a suitable printer driver have been loaded and that the serial port has been initialised. Now set the overlay program so that further PRINT statements go to the printer, but not to the screen. (For RS232 BASIC replace the following PRINT statements with LPRINT.) Generate a picture on the screen, either by loading a banner from tape or by loading and running a program. Now the picture may be dumped to the printer by loading and running one of the following programs.

The first example is based on the TRS80-style block graphic symbols that are available with the Epson MX80 with its original ROM or with the early Graftrax ROMs, but not with the later Graftrax plus ROMs. The block graphic characters are formed by making all permutations of filled squares in a 2 by 3 array that occupies a single character position. Fortunately, the 64 possibilities are arranged in a convenient sequence. The base value for the ASCII code that corresponds to all of the squares blank is 160. To this we must add a value for each nonblank square

1 . 0EE



1 . 0E0

1 . 0E0

Printing the INTERACT screen, cont.

```

-----
:      20=1   :      21=2   :
:      :      :      :
:-----
:      22=4   :      23=8   :
:      :      :      :
:-----
:      24=16  :      25=32  :
:      :      :      :
:-----

```

As shown in the diagram, the numbers to be added double as we read across and down the block. This leads to a simple screen dump in which we print out the screen 6 pixels at a time.

```

100 PRINT: REM MAKE SURE AT LEFT MARGIN
110 D=2: REM 2 ASSIGNED TO VARIABLE FOR SPEED
120 CO=160: REM BASE FOR CHARACTER SET
130 FOR Y=75 TO 3 STEP -3:REM SCAN DOWN IN 3-ROW BLOCKS
140 FOR X=2 TO 108 STEP 2:REM SCAN ACROSS IN 2-COLUMN BLOCKS
150 C=CO: F=1
160 FOR J=0 TO 2:REM SUM CONTRIBUTIONS FROM 6 PIXELS
170 FOR I=0 TO 1
180 C=C+F*SGN(POINT(X+I,Y-J))
190 F=D*F
200 NEXT I,J
210 PRINT CHR$(C);:REM PRINT THE 6-PIXEL BLOCK
220 NEXT: REM LOOP ACROSS THE TRIPLE ROW
230 PRINT:REM STEP DOWN FOR NEXT ROW
240 NEXT:REM LOOP FOR NEXT TRIPLE ROW.

```

The logic of this needs no comment. The screen gets transferred, but we lose the color information. The background comes out white and everything else is black. This works out fine for graphs, as shown in the example, but it isn't too attractive for pictures.

With either of the sets of Graftrax ROMs we can use bit-mapped graphics to make a screen dump that distinguishes colors. A program that does this is

```

0 DIM C(3,5): REM ARRAY FOR DOTS IN COLORED PIXELS
5 POKE 19215,25: REM ENABLE PEEK AND POKE
10 FOR I=0 TO 3: REM READ VALUES FOR DOT POSITIONS IN PIXEL
20 FOR J=0 TO 5
30 READ C(I,J)
40 NEXT J,I
50 X1=2: X2=108: REM RANGE OF X TO DUMP
60 Y1=5: Y2=75: REM RANGE OF Y TO DUMP
70 NC=6*(X2-X1+1):REM NUMBER OF DOTS IN A ROW
80 LH=INT(NC/256): REM HIGH BYTE FOR NC

```

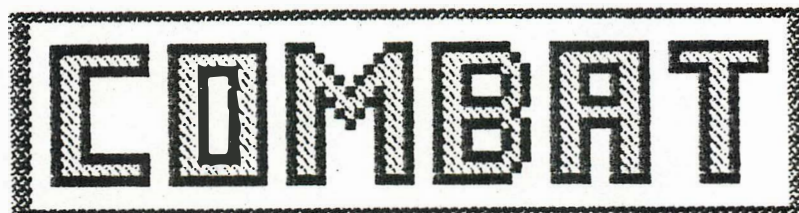
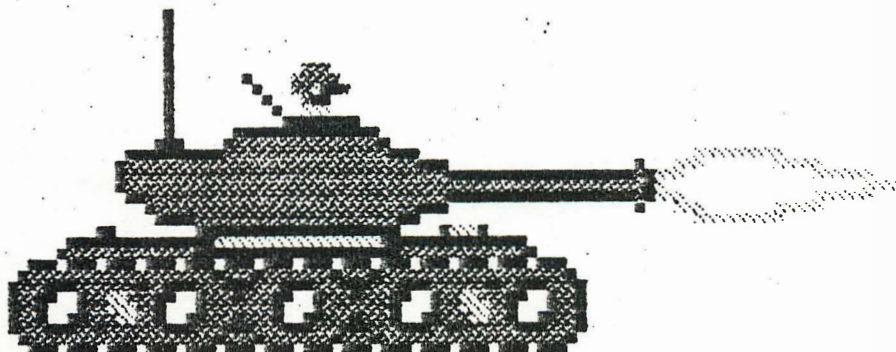
Printing the INTERACT screen, cont.

```

90 LL=NC-256*LH: REM LOW BYTE FOR NC
100 A=-16384: REM PORT OUTPUT ADDRESS FOR MV PORT
110 PRINT: REM GET TO LEFT MARGIN
120 PRINT CHR$(27);"3";CHR$(24);:REM SET 24/216" LINE SPACING
130 FOR Y=Y2 TO Y1 STEP -2: REM 2 PIXELS PER ROW
140 PRINT CHR$(27);"L";CHR$(LL);CHR$(LH);:REM SET UP FOR NC BYTES
150 FOR X=X1 TO X2: REM STEP ACROSS THE DOUBLE ROW OF PIXELS
160 CH=POINT(X,Y):REM COLOR OF TOP PIXEL
170 CL=POINT(X,Y-1):REM COLOR OF BOTTOM PIXEL
180 FOR I=0 TO 5:REM 6 BYTES ACROSS A PIXEL
190 CC=C(CL,I)+16*C(CH,I): REM 4 DOTS VERTICALLY FOR EACH PIXEL
200 POKE A,CC: REM SEND TO PORT
210 NEXT I,X: REM FINISH ROW
220 PRINT: REM STEP DOWN TO NEXT DOUBLE ROW
230 NEXT: REM LOOP THROUGH ROWS
240 PRINTCHR$(27);CHR$(64);:REM RESET PRINTER DEFAULT VALUES
500 DATA 0,0,0,0,0,0: REM VALUES FOR COLOR 0 = BACKGROUND
510 DATA 1,8,0,4,0,2: REM COLOR 1
520 DATA 5,9,10,6,3,12: REM COLOR 2
530 DATA 15,15,15,15,15,15: REM COLOR 3

```

The example shown is a banner that was stripped from a game tape (with Tapemaster).

The routine does require a little additional comment. Each poke puts out a column of 8 dot positions some of which may be blank. If the number poked is expressed in binary, the dot positions that are printed poked correspond to the one bits with the low-order bit corresponding to the lowest dot position. The top 4 dots are in one pixel and the other 4 dots are in the pixel immediately beneath it. Six adjacent columns are used to print the pair of pixels.



Printing the INTERACT screen, cont.

At first glance it might seem that one could use a PRINT statement instead of a POKE to put out the column. Usually this will not work because we will need to put out a row of over 600 columns. Most PRINT routines will not allow such a long line of characters without interjecting unwanted carriage return/line feeds.

The address that is POKEd is the output register of the port. Here we have used in line 100

A = -16384 for the Microvideo port.

The alternative is

A = -9 for the Slagh (Protecto) port.

In principle, we should test that the port is ready before sending out each value. However, BASIC runs so slowly that we can get away without this complication. The BASIC equivalents of the tests that would be needed in a machine language program are

```
193 IF PEEK(A+6) AND 16 = 0 GOTO 193
196 IF PEEK(A+5) AND 32 = 0 GOTO 193
for the MV port and
193 IF PEEK(A-1) AND 10 <> 2 GOTO 193
for the Slagh port.
```

The DATA statements 500-530 give the pixel maps that I chose for the colors. For colors 0 through 3, the 6\*4 block for a square pixel contains 0,4,12, and 24 dots.

To adapt this program for the Sabre port the PRINT statements must be replaced with calls to a machine-language subroutine via the USR function. The resulting program will not need a printer overlay to be present.

In the program above delete line 100 and replace or insert the following lines

```
7 GOSUB 300
110 B=USR(13):B=USR(10)
120 B=USR(27):B=USR(51):B=USR(24)
140 B=USR(27):B=USR(76):B=USR(LL):B=USR(LH)
200 B=USR(CC)
240 B=USR(27):B=USR(64)
300 POKE 19473,9:REM POKE IN USR ADDRESS
310 POKE 19474,74
320 FOR I=18953 TO 19028:REM POKE IN USR ROUTINE
330 READ J
340 POKE I,J
350 NEXT
360 RETURN
390 REM BYTES FOR USR ROUTINE
400 DATA 205,178,106,123,243
402 DATA 229,213,197,245,245
404 DATA 58,7,56,254,239
406 DATA 194,19,74,33,219
408 DATA 95,126,246,128,119
```

Printing the INTERACT screen, cont.

```
410 DATA 62,255,50,0,48
412 DATA 22,1,205,60,74
414 DATA 241,22,8,205,59
416 DATA 74,175,22,1,205
418 DATA 60,74,195,114,1
420 DATA 47,15,95,230,128
422 DATA 174,50,0,16,1
424 DATA 36,1,13,194,71
426 DATA 74,5,194,71,74
428 DATA 21,200,123,195,60
430 DATA 74
```

The USR routine is the standard Sabre port driver prefixed with the two instructions

```
CALL 6AB2
MOV A,E
```

that transfer the argument of the USR function to the A register for use by the Sabre routine. Three of the data values above are underlined. These are user changeable. The first, in line 412, specifies the number of stop bits. The others, in line 424, are the low-order and high-order bytes of the baud rate divisor (set here for 2400 baud).

I have put the USR routine at 4A09. This will be out of the way of everything except the CLOAD\* and CSAVE\* routines, which will clobber the code if they are used.

## INTERACT HEAD ALIGNMENT

by Alvy Albert 1704 Cadillac Circle S. Melbourne, FL 32935

In trying to solve a problem of head alignment, I came across a neat solution. I think that some people may have used the Interact alignment tape which is at best crude. What I did was with my IOS ROM, I filled 4000 to 49FF with 01, 4A00 to 4FFF with AA and 5000 to 5F7F with FF. Then W 4000,5F7F. This gives a constant data feed that is easy to adjust to at 3 different audio levels..It can also be done with the HILO Monitor or Monitor 1.1

## INTERACT I/O IN FORTH

by Russ Schnapp, 8062 Gold Coast Drive, San Diego, CA 92126

I have received numerous questions on how one may access Interact i/o devices from FORTH. This article will, I hope, answer most of these questions, and stimulate further interest in the language. The most general answer to this question of, "How do I interface FORTH to my i/o device?" is this: The same way you would do it in assembler language; you write device driver routines, and use those. Granted, not all of you are systems programmers, and so you might not know what I mean by a device driver, but I'll try to clear that up in this article.

Basically, a driver is a routine (or set of routines) which provide a simple interface to a (possibly quite complex) device. For example, the routine (in ROM) that implements the BASIC TONE statement is a driver for the "software sound" output device. In many cases, the device drivers already reside in the ROM, as for the case of TONE. A few devices require you to supply driver routines, such as the TI sound chip, or an RS-232 port. Some devices are provided with an extremely easy-to-use interface; These have driver routines in ROM that are executed automatically, 60 times each second. These devices are the keyboard, joysticks, potentiometers, and the 60Hz clock. The results of reading these devices are left in fixed memory locations for your program's perusal, at its leisure. Let's talk about these devices first:

**KEYBOARD:** The keyboard's memory locations are (in hex):

5FD0: Keyboard status; 0=no keystroke ready yet

5FD1: Keyboard data; one byte of ASCII

Some existing FORTH routines are KEY, ?TERMINAL, EXPECT.

A useful word is : ?KEY 5FD0 C^ ; ( ->f;f<>0 when key is struck)

**JOYSTICKS:**

5FF1: Left joystick

5FF2: Right joystick

The values in these locations have identical meanings to the BASIC JOY definition. Thus a useful word is:

: JOY 5FF1 + C^ ; ( n->v; n=0 left Joy, n=1 right Joy )

**ANALOG PORTS** (updated 10 times per second):

5FF7: Left Fire button ( <16 when button has been pushed )

5FF8: " Potentiometer (paddle)

5FF9: " Spare

5FFA: Right Fire

5FFB: " Potentiometer

5FFC: " Spare

You probably get the idea by now;

: FIRE IF 5FFA ELSE 5FF7 ENDIF C^ 10 < ; ( j->f; j=0 is left)

: POT IF 5FFB ELSE 5FF8 ENDIF C^ ; ( j->v)

The analog ports can be turned off, yielding a small speed improvement, by storing a nonzero value in the word at 5FF5.

## FORTH I/O (continued)

## 60 HZ CLOCK:

This clock resides in a word at 5FEF. The clock's value is incremented by one during each interrupt. It can be used to time an event with relatively low precision.

The remaining devices are not quite as easy to use. Some already have drivers in ROM, some in FORTH...

## TAPE CASSETTE DRIVE:

The easiest way to use the tape is to use the words already defined in FORTH, such as GREC, ?GREC, PREC, READLEAD, TAPEON, TAPEOFF, REWIND, etcetera. These really are adequately defined in the documentation accompanying FORTH for the Interact. The second most commonly asked question is, "How can I write a variable to cassette?" Let's say you have a variable, of length less than or equal to 256 bytes, named DATA. To write it out:

```
: WRITE_SHORT ( address recordsize-> ; write to cassette )
  WRITELEAD PREC TAPEOFF DROP ;
DATA HEX 40 WRITE_SHORT
```

To read that same variable back:

```
: READ_SHORT ( addr recsize-> ; read from cassette )
  READLEAD GREC ?GREC TAPEOFF DROP ;
DATA HEX 40 READ_SHORT
```

By the way, this addition for impatient souls:

```
: SHORT 600 60DA ! ; ( ; WRITELEAD will write a fast leader )
: LONG 1800 60DA ! ; ( ; WRITELEAD writes normal leaders )
```

SCREEN: The screen is adequately handled by EMIT, PTC, and the GRAPHICS vocabulary.

## SOFTWARE SOUND PORT:

A well-known routine resides in ROM to generate tones of varying frequencies and durations. To access this routine requires the use of an assembler-derived word:

```
CODE TONE ( period duration-> ) DI, D POP, H POP, B PUSH,
  B H MOV, C L MOV, 7BF CALL, B POP, EI, NEXT JMP,
```

## HARDWARE SOUND PORT:

This is one of the most difficult devices to use. It consists of 3 bits of one port, and 2 bits from each of 8 more ports. It is also very poorly documented. The following words help out:

```
: 4* DUP + DUP + ; ( n->4*n ; )
: MIXER ( mix-> ; select mix of sounds )
  7 AND 5FC0 C^ 0F8 AND OR DUP 3000 C! 5FC0 C! ;
: SOUND ( n-> ; send n to the sound chip )
  DUP 2000 ! 4* DUP 2002 ! 4* DUP 2800 ! 4* 2802 ! ;
```

The sound value has the following meaning ( bit 0 is the least significant bit) (a "?" means a poorly understood function):

FORTH I/O (continued)

BIT DESCRIPTION

- 0 if bit0=1, the sound chip is disabled. Transition to 0 begins attack, if bit10=0
- 9,1 ? Envelope Select. 00=VCO, 01=One Shot, 10=Mixer, 11=Alternating. ES=01 for attack/delay operation.
- 2 VCO source. 0=V, 1=V+SLF (V determined by bits 5,4)
- 3 Noise. 0=White (fine), 1=Pink (coarse)
- 5,4 VCO tone period. 00=High frequency, 11=Low
- 7,6 Attack period. 00=Fast, 11=Slow.
- 8 ? VCO Ext control. 0=f up, 1=f down. ???
- 10 One Shot. 0=Attack shape, 1=Decay shape.
- 11 Unused.
- 12 Decay period. 0=Fast, 1=Slow.
- 13 Noise inhibit. 0=Allow noise, 1=Inhibit noise.
- 15,14 SLF (Super Low Frequency VCO) period. 00=High freq, etc

Mixer input meanings:

- 0=VCO                      1=Noise                      2=SLF+Noise                      3=SLF+VCO
- 4=SLF                      5=VCO+Noise                      6=SLF+Noise+VCO                      7=Tape only

RS-232:

This is the single most commonly asked-about port. There are at least 3 different versions of RS232 (Sabre, Slash, Microvid). I will include here only the Slash interface, since I can't test any others, personally:

```

_( RS232 INTRFC, 1 OF 2; RD232, WR232, (PRINT , 232INIT ... )      0
_FORTH DEFINITIONS HEX                                           1
_: ?232 FFF6 C^ 1 AND ; ( IS THERE AN INPUT FROM RS232? )      2
_: RD232 BEGIN ?232 UNTIL FFF7 C^ 7F AND ; ( GET BYTE FM RS232 ) 3
_: (WR232) BEGIN FFF6 C^ 0A AND 2 = UNTIL FFF7 C! ;             4
_: WR232 DUP (WR232) 0D = IF 0A (WR232) ENDIF ;                5
_2 VARIABLE (PRINT) ( 1-RS232, 2-SCREEN, 3-BOTH )              6
_: (EMITT) ( REPLACES CURRENT EMIT DEFINITION )                7
_(PRINT) ^ DUP 1 AND IF OVER WR232 ENDIF                        8
_2 AND IF LBR ' EMIT ^ , RBR ELSE DROP ENDIF ;                 9
_' (EMITT) CFA ' EMIT ! ( PATCH IN NEW (EMIT )                 10
_: 232INIT 03 FFF6 C! 1505 FFF5 ! ( 4800 BAUD, 155C FOR 300 BD ) 11
_ 2 (PRINT) ! CR ;                                             12
_232INIT ( INITIALIZE RS232 INTERFACE RIGHT NOW )              13
_' 232INIT CFA ' ABORT 6 + ! ( PATCH ABORT TO INIT RS232 )    14
--->                                                            15

_( RS232 INTERFACE, 2 OF 2; FINISH PATCHING, EDITOR WORD PR )  0
_' CR ' LIT CFA OVER ! 2 + 0D OVER ! 2 + ' EMIT CFA OVER !    1
_2 + ' ;S CFA SWAP ! 4EF3 ( DOCOLON ) ' CR CFA ! ( PATCH CR )  2
_EDITOR DEFINITIONS                                           3
_: PR ( PRINT CURRENT SCREEN )                                 4
_ (PRINT) ^ 1 (PRINT) ! CR                                     5
_ 10 0 DO FORTH I EDITOR T LOOP                                6
_ CR (PRINT) ! ;                                              7
    
```

## FORTH I/O (continued)

RD232: - n  
Input a byte from the RS232 port.

WR232: n -  
Write n to the RS232 port.

(PRINT): - addr  
Variable controlling print mode; 0=EMIT throws away output,  
1=RS232 out only, 2=Screen only, 3=Both screen and RS232 out

PR: -  
Send current edit screen to RS232.

232INIT: -  
Initialize RS232 port. Change "1505" to appropriate value  
for your interface and desired baud rate. Executed auto-  
matically upon reset.

The variations for other ports only apply to ?232, RD232, (WR232), WR232, and 232INIT. The changes are simply for appropriate mask values and addresses. The interface assumes you are using CTS handshaking (by the way the Comet 8300R printer runs very nicely at 4800 baud in this mode). Perhaps the vendors of these interfaces could start supplying the FORTH interfaces.

Well, all this should give you some flavor for using FORTH for i/o. For more specific information on device i/o, I'm afraid we will have to wait for other authors to better describe device characteristics. Perhaps there is some existing documentation I am unaware of? In any event, don't be afraid to experiment. That is what FORTH allows you to do best!

## MACHINE SHOP TALK

by George Leggett 20562 Woodward Mt. Clemens, MI 48043

Hello again, and welcome to another Machine Shop Talk. In this issue, we will continue our discussion with the SOUND Routine. As you recall, when we make tones, the ROM 1 calls up the Tone Routine at 07BF. It would be great if there was a Sound Routine in ROM 1. But unfortunately, there is no such routine. By sound, I mean the special sound effects in the games, the beeps, buzzes, the bell in the Control G in BASIC. The BASIC SOUND Command is simple: SOUND 0-7, 0-32767. As everyone know, there are enormous possibilities and sound combinations. Also, in BASICALLY SPEAKING, there is a long list of different sounds you can make. But, what if you want a Sound in your Machine Language program and there is no Sound Routine in ROM 1? Well, what I did was to extract and modify the Sound Routine from Level II BASIC. For those of you with 8080 FOR EVERYONE or RAM ROUTINES, you have the routine and are familiar with its use. Simply CALL 5DD2. I should like to add here not only calling a single sound, but mixing sounds and using time delay loops, all done in wonderful 8080 Machine Language.

Machine Shop talk, cont.

A sound my wife first found when we got our Interact was a gunshot. In BASIC this is:

```
10 SOUND 1, 20
20 SOUND 1, 21
```

Try it out. For a longer sustain, add a delay loop in Line 15.

```
15 FOR X = 1 TO 500: NEXT
```

If you would like a 6-gun, try:

```
10 FOR X = 1 TO 6
20 SOUND 1, 20
30 SOUND 1, 21
40 FOR D = 1 TO 200
50 NEXT
60 NEXT
```

Lines 40 and 50 are the delay loop between gunshots. You can vary that and hear the results, or put a delay in Line 25 to vary your echo.

Fortunately, all of the Sounds you have used in BASIC are the same in Machine Language, and as we saw with the tones, speed is greater in Machine Language.

The Sound Routine can be found right in Level II BASIC at 7FCA. To use it in Machine Language, load D and E with a number from 0 to FFFF, or the second number of your BASIC SOUND command. Load Register C with 0 to 7, or the first number in our familiar command. CALL 7FCA and there your sound will be!

Here is a sample Sound program, and a BASIC program to use with it if you have no Monitor.

Address	Hex	Mnemonic	Description
5600	11	LXI D	Load Register Pair D and E
5601	4C	*	Hex Equivalent for Decimal
5602	01	*	Number 332 for Sound
5603	0E	MVI C	Move next Byte into C
5604	03	*	Number 3 for SOUND
5605	CD	CALL	
5606	CA	*	Call Sound Routine in
5607	7F	*	Level II BASIC
5608	CD	CALL	
5609	ED	*	Call Keyboard Routine in ROM 1
560A	07	*	waits for key to be pressed.
560B	C9	RET	Return to monitor or BASIC.

If you have a ROM Monitor, you should have no problem running this short routine, as it is using a routine out of BASIC, but not BASIC itself. For those of you using taped monitors such as Micro Video, load your Monitor, enter in the routine, save it on tape, load in BASIC, load in the routine, and run it. You should hear a sound. Here is a BASIC USR program. If you are using Level II, use Line 10, and those with newer BASIC versions of course not use the initial POKE.

```
10 POKE 19215, 25
20 POKE 19473, 0
30 POKE 19474, 80
40 U = USR(0)
```

Machine Shop talk, cont.

Run this, and you should hear the sound. With BASIC, when you press a key, the OK from BASIC will appear. By pressing another key, the sound will stop. When using a monitor, one key prompt will end the routine.

You will probably want to extract this Sound Routine for your own personal library of Machine Language Routines. With a ROM Monitor or Micro Video Monitor, simply Write from 7FCA to 7FF7 onto tape. The Micro Video Monitor can be loaded right over BASIC, and the routine saved on tape. You can then use your monitor to move the routine anywhere you wish, or leave it at 7FCA. Remember to change the Jumps or Calls. For those of you who are not able to Write the routine directly to tape in this manner, use Hendrickson's Basic To Monitor, (INTERACTIONS Vol. 2 no. 3) to look at the routine. Write it out on a sheet of paper, then load it into your computer at whatever address is compatible with your Monitor.

To turn sound off, load D with 10 and E with 00, and C with 7. For those with RAM ROUTINES, CALL 5DC5 for SOUND OFF.

Try using sounds, tones, with the Delay Loop Routine in ROM 1 to have fun making sound effects in your next program.

## RANDOM CRAZY

by George Newcomer 413 Pearl Charlotte, MI 48813

This graphic program and the two that follow are modified from TRS 80 programs appearing in Creative Computing. They have been enhanced with color and sound. George hopes you find them enjoyable and use them. This program is for 8K Graphics BASIC only. The others work with either BASIC.

```

10 REM *RANDOM CRAZY*
20 REM * BY GEORGE NEWCOMER
30 CLS : FOR I = 1 TO 25
40 OUTPUT "RANDOM",40,50,7: OUTPUT "CRAZY",42,40,7
50 OUTPUT "RANDOM",40,50,0
60 OUTPUT "CRAZY",42,50,7
70 OUTPUT "CRAZY",42,40,0
80 OUTPUT "RANDOM",40,40,7
90 CLS : SOUND 6,242: NEXT I
100 PLOT 1,1,1,112,77
110 CLS : COLOR 7,1,0,3: REM TRY DIFFERENT COLOR'S HERE
120 PRINT "ARE YOU CRAZY YET": PRINT : PRINT : PRINT : PRINT : PRINT :
PRINT
130 PRINT "LET PROGRAM RUN A FEW MOMENTS ANDTHEN PUSH SPACE BAR"
140 PRINT : PRINT : PRINT : PRINT : PRINT : PRINT
145 PRINT "AFTER A FEW MOMENTS PUSH ANY OTHER KEY": PRINT : PRINT
: PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT
150 PRINT "AFTER A FEW MORE MOMENTS PUSH KEY 'A'"
160 PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRI
NT : CLS

```



Random Crazy, cont.

```

165 PRINT "THEN REPEAT THE ABOVE STEPS, ALSO TRY A DEFERENT SEQUENCE
!"
166 PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRI
NT : CLS
170 M = 55:N = 38.5: PLOT M,N,1
180 X = 55:Y = 38.5: PLOT X,Y,1
190 P = 55:O = 38.5: PLOT P,O,1
200 R = 55:S = 38.5: PLOT R,S,1
210 C = INT(3 * RND(1) + 1)
220 A = INT(4 * RND(1) + 1)
230 H = INT(55 * RND(1) + 1)
240 FOR B = 1 TO H
250 ON A GOTO 260,310,360,410
260 X = X + 1: IF X > 102 THEN X = 102
270 R = R + 1: IF R > 102 THEN R = 102
280 M = M - 1: IF M < 10 THEN M = 10
290 P = P - 1: IF P < 10 THEN P = 10
300 GOTO 450
310 X = X - 1: IF X < 10 THEN X = 10
320 R = R - 1: IF R < 10 THEN R = 10
330 M = M + 1: IF M > 102 THEN M = 102
340 P = P + 1: IF P > 102 THEN P = 102
350 GOTO 450
360 Y = Y + 1: IF Y > 70 THEN Y = 70
370 S = S - 1: IF S < 10 THEN S = 10
380 N = N - 1: IF N < 10 THEN N = 10
390 O = O + 1: IF O > 70 THEN O = 70
400 GOTO 450
410 Y = Y - 1: IF Y < 10 THEN Y = 10
420 S = S + 1: IF S > 70 THEN S = 70
430 N = N + 1: IF N > 70 THEN N = 70
440 O = O - 1: IF O < 10 THEN O = 10
450 PLOT X,Y,C: PLOT M,N,C
460 PLOT P,O,C: PLOT R,S,C
470 IF PEEK(24529) = 32 THEN 540
475 IF PEEK(14339) = 253 THEN 170
476 IF PEEK(14336) = 191 THEN 1000
480 Q = Q + 1: IF Q = 200 THEN Q = 0
490 IF Q < 50 THEN SOUND 6,3500
500 IF Q > 50 AND Q < 100 THEN SOUND 6,460
510 IF Q > 100 THEN SOUND 3,30
520 NEXT B
530 GOTO 210
540 C = INT(7 * RND(1) + 1)
545 IF PEEK(14336) = 191 THEN 1000
550 SOUND 6,456
560 GOTO 220
570 END
1000 CLS : PRINT "DROVE YOU CRAZY  FINALLY!!"
1010 GOTO 1000

```

NOTE - Many of the listings in this and the last newsletter are made using the new MULTI-BASIC EDITOR by Harry Holloway. SEE his ad on page 21. Therefore some programs may contain extra spaces to enhance readability, but not necessary for program usage.

## WORM

```

1 REM -WORM-BY GEORGE NEWCOMER
5 CLS
10 OUTPUT "WORM",40,50,1: FOR I = 1 TO 1000: NEXT
20 CLS :C = 0:N = 0
110 X = INT(100 * RND(55) - 1)
120 Y = INT(100 * RND(38) - 1)
130 PLOT X,Y,C
131 PLOT X,77 - Y,C
132 PLOT 110 - X,Y,C
133 PLOT 110 - X,77 - Y,C
134 C = INT(4 * RND(1) + 1)
135 N = N + 1
136 IF N < 50 THEN SOUND 3,109
140 A = INT(10 * RND(4) + 1)
150 ON A GOTO 160,170,180,190
160 X = X + 1: GOTO 200
170 Y = Y + 1: GOTO 200
180 X = X - 1: GOTO 200
190 Y = Y - 1
200 IF X < 0 THEN X = 55: GOTO 130
210 IF X > 110 THEN X = 0: GOTO 130
220 IF Y < 0 THEN Y = 38: GOTO 130
230 IF Y > 77 THEN Y = 0: GOTO 130
235 IF X = Y THEN SOUND 3,264
236 IF N < 50 THEN SOUND 3,109
237 IF N = 100 THEN SOUND 3,268
238 IF N > 150 THEN N = 0
240 GOTO 130

```

## MIRROR IMAGE

```

10 CLS
20 REM -BY GEORGE NEWCOMER
30 OUTPUT "MIRROR",37,50,1
40 OUTPUT "IMAGE",40,40,1
50 FOR I = 1 TO 1000: NEXT
60 CLS
110 X = INT(64 * RND(1) + 1):Y = INT(24 * RND(1) + 1)
120 A = INT(3 * RND(1) + 1)
125 K = INT(5 * RND(1) + 1)
126 SOUND 3,600
127 REM -TRY PUTTING A COLOR STATEMENT HERE
130 ON K GOTO 140,150,160,170,170
140 X = X - 1:Y = Y - 1: GOTO 180
150 X = X + 1:Y = Y - 1: GOTO 180
160 X = X - 1:Y = Y + 1: GOTO 180
170 X = X + 1:Y = Y + 1: GOTO 180
180 IF X < 0 OR Y < 0 OR X > 112 OR Y > 77 THEN 230
190 IF POINT(X,Y) = - 1 THEN 300
200 PLOT X,Y,A: PLOT 112 - X,Y,A
210 PLOT 112 - X,77 - Y,A: PLOT X,77 - Y,A
220 GOTO 120
230 B = INT(5 * RND(1) + 1)

```

Mirror Image, cont.

```
240 DN B GOTO 250,260,270,280,290
250 X = 62:Y = 0: GOTO 120
260 X = 0:Y = 38: GOTO 120
270 X = 62:Y = 38: GOTO 120
280 X = 0:Y = 0: GOTO 120
290 X = 32:Y = 12: GOTO 120
300 PLOT X,Y,0: PLOT X,47 - Y,0
310 PLOT 112 - X,77 - Y,0: PLOT 112 - X,Y,0
320 GOTO 120
330 END
```

## RELOCATE THE HI-LO MONITOR for 32K RAM

by Robert M. Alpert  
1144 N. 35th St., Apt. 2  
Camden, NJ 08105  
COMPUERVE no. 70525,1213

The following information will interest those Interact owners who have both the 32-K expansion and the Hi-Lo monitor, particularly the RS-232 version.

(NOTE: The following operations all refer to the HI monitor)

The Hi-Lo monitor by Harry Holloway is a powerful utility for those interested in machine and assembly language programming. However, the program protects itself from disassembly and relocation. The following routine, entered with the Hi-Lo's 'A' command starting at 6000 hex, will relocate the Hi monitor to top end of RAM in the 32-K machine:

```
6000 LXIH 6B00
6003 LXID AB00
6006 MOVAM
6007 XCHG
6008 MOVMA
6009 XCHG
600A INXH
600B INXD
600C MOVAM
600D CPI 80
600F JNZ 6006
6012 RET
```

After entering this code, type 'G 6000' and the entire Hi monitor will be copied from 6B00-7FFF to AB00-BFFF. The monitor may now be disassembled (with the RS-232 version I was able to run out an entire printed listing this way).

Relocate the HILD Monitor, cont.

The resultant listing will still refer to the original address locations. This can be easily interpreted as there is a constant offset:

```
6— = A—
7— = B—
8— = C—
```

To make the new copy functional, we use the 'O' command to offset the sensory references:

```
OAB00,BFFF,6B00,7FFF,AB00
```

(Note that 'O' will display all changed references as it works. Using a partial scroll here will speed things up considerably.)

In addition, there is a table of addresses located at B820 which must be changed manually. Starting at B821 change every other byte to reflect the offset, using the 'S' command. For example:

```
S8821

B821 6F-AF
B822 53-

B823 6D-AD
B824 7D-
B825 73-B3

.....etc.
```

Continue these changes up to B853, inclusive.

Next, we change the protective addressing limits to match the new location. Use 'S' to make the following 3 changes:

```
S861 6B-AB
S862 FF-
S863 7F-BF
...
S867 6B-AB
```

Now we assemble a new start code at 4C00, and a duplicate at B854 (this is used by the 'B' command to reinstate the start code when needed):

```
4C00 LXISP C000 ... B854 LXISP C000
4C03 JMP AB00 ... B857 JMP AB00
```

The info for the following changes was kindly provided by Harry Holloway:

Relocate the HILD Monitor, cont.

These changes will correct a bug that gives incorrect flag values after a 'G' command (16K owners see end of article):

```
ACE2 PUSHF
ACE3 POPH
ACE4 SHLD BE78
ACE7 LXIH 0000
ACEA DADSP
ACEB SHLD BEB2
ACEE LXISP C000
```

The user default stack pointer is set by an LXIH instruction at the beginning of the program, AB00 in this case. This should be changed to an appropriate value, for example:

```
AB00 LXIH AB00
```

will set the user stack just below the start of the monitor.

AT BECF-BED0 is the tape output list spec for the filename buffer. This should be offset from the current value of 7EC4 to BEC4 (low byte first).

To get the 'KS' checksum command to work: after all the above changes have been made, 'G AB00' to jump to the new location. Substitute zeroes for the two bytes at BF7E-BF7F (since the protective addressing limits will prevent the use of 'S' to accomplish this, write a simple routine to move the necessary data in. Alternately, the address limits may be removed completely before the monitor is relocated. See the addendum for 16K Interact owners at the end of this article. I don't necessarily recommend that method for relocation as matching the address limits to the new location can help prevent accidentally assembling or substituting inside the monitor.) When you have zero in those bytes, do a 'KS' and note the returned number. Calculate its two's complement (HO, number will do this). Substitute the two's complement (low byte first) into BF7E-BF7F. 'KS' should now respond with 'OK'.

The monitor may now be written to tape. If you set the address checks to match the new location, the new monitor will not write itself to tape. Jump back to the old monitor with 'G 6B00'. If the checks are totally disabled, you can use the new monitor to make a copy of itself. Set up the tape output list to copy 4C00-4C05 and AB00-BFFF to tape. You now have a functional monitor which can be used to load, examine, and modify 16K programs in one piece.

Relocate the HILO Monitor, cont.

FOR OWNERS OF 16K MACHINES:

The address limits may be totally disabled with the following routine:

```
6000 LXIH FFFF
6003 SHLD 7860 (5760 FOR LO- MONITOR)
6005 RET
```

The monitor may now be examined, modified, and separate Hi and Lo versions may be written to tape (this allows you to overlay a previously loaded program). If you want to correct the bug described in the above article, these are the changes for the Hi and Lo monitors, respectively:

6CE2 PUSHP	4BE2 PUSHP
6CE3 POPH	4BE3 POPH
6CE4 SHLD 7E78	4BE4 SHLD 5D78
6CE7 LXIH 0000	4BE7 LXIH 0000
6CEB SHLD 7E82	4BEB SHLD 5D82
6CEE LXISP 7FFF	4BEE LXISP 5FC0

### Elimination of Spring Contacts to Improve Reliability of Interact Controller

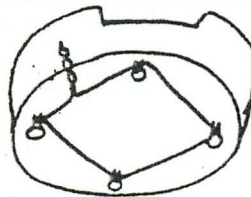
by Jerry W. Goerz 2394 LaRochelle Lexington, KY 40504

- 1) Remove three screws to open controller.
- 2) Remove four screws to release P. C. board.
- 3) Using small soldering iron, remove the four spring-contact switches from the P. C. board. Do not remove the contact points beneath the spring switches.
- 4) Remove the black wire from the P. C. board.
- 5) Fasten the P. C. board to the controller with four screws.
- 6) Find four flat steel or copper washers with an inside diameter of 1/8 inch, or which fit #6 screws.
- 7) Set the washers over the four square contact points which remain after removing the spring switches. Solder the washers to the contact points, to enlarge them.
- 8) Scrape flux from the top of the enlarged contacts. A sharp knife may be used to scrape excess solder from the contacts.

## Elimination of Spring Contacts, cont.

- 9) Slide the joystick out of the controller.
- 10) Remove four Phillips screws to release the back cover of the joystick.
- 11) Cut what is left of the four plastic nubs from the cover.
- 12) Find four pins, such as are found on new shirts, or four wire brads with smooth flat or rounded heads. Test these pins at the pointed end to be sure they will take solder.
- 13) Using a wire-cutters, cut each pin to a length of  $5/16$  inch.
- 14) Holding a pin with a long-nose pliers, heat it with a soldering iron and press it, shank first, into the plastic cover where a nub used to be. Let about  $1/16$  inch of the shank protrude, with the head atop it, from the cover. Repeat this procedure for all pins.
- 15) Skin the insulation from a 4-inch length of small gauge (such as 24-gauge) solid copper wire.
- 16) Connect all four pins together with the wire, winding it once around each pin, and joining both ends of the wire in a splice to the left side of the slot for the potentiometer leads, when holding the cover with the pins down.

- 17) Solder the pins to the wire, and tin the splice with solder.



- 18) Provide a strain-relief for the blue wire which is soldered to the potentiometer. Use a one-inch piece of insulated 24-gauge solid wire, and strip  $1/8$  inch from one end. Solder the bare end to the center terminal of the potentiometer. Wrap the insulated end several times around the blue wire.
- 19) Reassemble the joystick using the four Phillips screws.
- 20) Remove two of the three black rubber gaskets which fit at the hole in the controller which the joystick fits through.

**Elimination of Spring Contacts, cont.**

- 21) Solder the black wire, previously removed from the P. C. board, to the splice in the wire which connects the four pins.
- 22) Slide the joystick through the opening in the controller.
- 23) Reassemble the controller shell with three screws, taking care not to pinch wires between the shell halves.

**Replacement of a Defective Interact Controller Pushbutton  
With a Common Inexpensive Pushbutton**

**by Jerry W. Goerz 2394 LaRochelle Lexington, KY 40504**

- 1) Remove three screws to open controller.
- 2) Gently pry away the two metal sides of the switch from the red switch body. Snip wires. Remove and discard the defective switch body.
- 3) Pull out the white plunger from the switch cap, and use a sharp knife to trim off the "V", leaving a flat surface. Replace the plunger.
- 4) Squeeze the metal sides of the old switch back together, so that the prongs which once wrapped around the red switch body will now fit into the threads of a miniature normally-open pushbutton switch such as a Radio Shack #275-1547, with the nut removed.
- 5) Screw the miniature switch into position, far enough so that pushing the square switch cap will cause the miniature pushbutton switch to make contact. This may be tested with an ohmmeter, or some other method of testing continuity. The plunger of the miniature pushbutton may need to be trimmed with a sharp knife.
- 6) Dab some epoxy onto the prongs where they meet the threads, to hold the switch in position.
- 7) Carefully solder the wires to the switch's solder lugs, using a small soldering iron.
- 8) Reassemble the controller shell with three screws, taking care not to pinch wires between the shell halves.



\*\*\*\*\*  
Announcing a new utility, the

**UNIVERSAL PRINTER OVERLAY**

This program adds a hardcopy capability to either Level II or to 8K Graphics BASIC for users whose Interacts have RS232 ports. Both the Slagh and Microvideo ports are supported and the printer overlay is compatible with both the Leonardo and Fastline graphics overlays. This compatibility is attained at a small cost in programming space. (Less than 500 bytes.)

Provision is made for the user to write to tape a copy of the enhanced BASIC so that subsequent loads may be of a single cassette.

Available from Harry Holloway, PO Box 2263, Ann Arbor, MI 48106. Price \$12.50 postpaid within the US or Canada. MI residents please add \$0.50 sales tax.

\*\*\*\*\*

\*\*\*\*\*

Announcing the

**MULTI-BASIC EDITOR**

This utility has been designed to do for the BASIC programmer what the HILD monitor did for the machine language programmer. (Please don't confuse it with EZEDIT. Both are BASIC editors, but there the resemblance ends.)

The features include: Compatibility with Level II, 8K Graphics and RS232 BASIC, also with the FASTLINE and LEONARDO overlays. Hard copy via Slagh (Protecto), Microvideo, or Sabre ports (all three drivers in one version). Formatted listing inserts spaces for readability of programs that have spaces omitted to save memory. Translate either way between Level II and RS232 BASIC. Auto line numbering. Cross-reference listing for GOTOs and GOSUBs. Delete and extract. Renumber the program (unlike EZEDIT, works with multiple statement lines). Move a block of lines (with renumber). Squeeze out REMs and unnecessary spaces to shorten a program. Join two or more BASIC lines (saves four bytes per join). String find and substitute commands with wildcard constructions. Csave with options of saving only part of the BASIC program and of combining with machine code subroutines. Peek and poke for adding short machine code sections (longer sections may be written separately with a monitor and loaded for generation of a combined BASIC/machine code tape). Join two BASIC programs in append or overlay mode. Output permits embedding of comments in hardcopy record and transmission of escape sequences and other control codes for printer control. With comprehensive documentation.

Available Oct. 1st 1982 from Harry Holloway, PO Box 2263, Ann Arbor, MI 48106. Price \$22.50 includes first class postage within US or Canada. MI residents please add \$0.90 sales tax.

\*\*\*\*\*

FORTH FOR THE INTERACT

This is an adaptation of the F.I.G. standard Forth for Interact computers. It comes complete with an assembler, a graphics package, and a line editor. Forth is faster and more powerful than BASIC.

Included is documentation of the differences from, and additions to the F.I.G. standard. A Forth manual is NOT included (I suggest "Starting Forth," by Leo Brodie, available from the Forth Interest Group, POB 1105, San Carlos, CA 94070).

By the way, Forth now makes use of all memory available on your Interact, whether you have 16K, 32K, or even 48K bytes.

For your copy of Forth for the Interact, send \$12 to:  
R. Schnapp, 8062 Gold Coast Drive, San Diego, CA 92126

Also available are:

- SPACE BATTLE-- You are at the controls of your own space fighter. With a touch of your joystick you roll left, right, or accelerate forward. Hit the fire button to launch a shimmering photon torpedo at your opponent. For two players. ....\$8
- A super-fast LIFE program, using the left joystick.... \$6
- A smooth PING-pong program, uses both joysticks..... \$6

\*\*\*\*\*

QUALITY PROGRAMS FOR THE INTERACT

from: David J. Schwab  
10 Jay Lee Court  
Ann Arbor, Michigan 48104

SKETCH PAD - Basic program with extensive machine language subroutines for creating, modifying, and saving screen displays. Draws open and filled circles (round ones!), triangles, rectangles, lines, and letters with super-fast joystick positioning. Saves screen on tape with or without stop code (to create program banners). Hours of fun for all ages.....\$8.00

EDU-BASIC OVERLAY - Allows for PEEK, POKE, and USR type facilities in EDU-BASIC. Also for use with Slash US0 port to direct output or listings to port. Use this powerful language to its full potential.....\$8.00

QUEST in EDU-BASIC - An 8k adventure program. You must retrieve a treasure from an underground maze. Descriptions are given of each room and you have 6 directions in which to try to proceed. A pirate lurks in the maze and may steal the treasure back.....\$5.00

8080 DISASSEMBLER in BASIC - This Basic program lists addresses, contents, corresponding ASCII character, and standard 8080 mnemonic assembler language op codes and registers for any memory locations. Includes complete instructions and sample output listings form.....\$5.00

## INTERACT EXPANSION

Expand your Interact computer to a total of 48K bytes of RAM and open the door for floppy disc, S-100, high resolution graphics and more.

Our expansion consists of an expansion interface board which is plugged into the Interact microprocessor integrated circuit socket in the same manner as the available RS-232 serial port boards and an enclosure external to the Interact which houses the actual peripheral expansion boards, themselves. The expansion interface (IE board) conditions all of the necessary Interact signals and brings them out on a 40 conductor ribbon cable to the IMB-1 motherboard. The peripheral electronics, then, are plugged onto the motherboard with the use of standard 44 pin edge connectors.

The IE and IBM-1 boards are available for purchase now. The IMEM-1 32K memory board should be available mid November, 1982. This board has space for 16 4116 16K dynamic RAM integrated circuits arranged in two banks which brings the total Interact capacity to 48K. In addition, there is bank select circuitry provided on-board that, with some minimal additional electronics, will allow you to stack up to four of these boards with full software control. This will bring the total Interact capacity up to 144K.

Additionally we have plans for floppy disc, S-100 conversion and the addition of more space for ROM/RAM on the first 16K page. We will be translating these into printed circuit boards in the near future.

We are offering these expansions as kits, although we will assemble and test individual assemblies. We cannot accept your Interact for us to do the entire work and ship back. Additionally, you will need to furnish your own power supply for all of the additional electronics we supply; the Interact power supply is not hefty enough to do the job.

Please see below for the price list. This is included to give you an idea of the cost. Because we cannot include all of the features of these units in this ad, we ask that you send a stamped self-addressed business sized envelope for full information. We will not accept any money until this inquiry is made.

IE board and plans - \$25.50 / with parts - \$78.50 / assembled - \$97.50

IMB-1 board and plans - \$24.50 / 44 pin edge connectors - \$3.95 each

IMEM-1 board and plans - \$44.50 / with parts - \$134.50 / assembled - \$169.50

IEN-1 tailor made enclosure - \$19.95

INTERWORD - a full featured word processor machine language program for the Interact. This is a menu controlled cursor driven word processor that, within the hardware restrictions of the Interact, rivals those written for other small computers. It features full horizontal and vertical scroll, full cursor control (up,down,left,right), lower case letters on the screen, either the normal 17 characters per row or 25, supports both Micro-Video and Slagh RS-232 boards, works with 16K, 32K, or 48K Interacts, includes a mini-monitor which allows you to change default parameters to suit your tastes and make back-up copies, price includes future updates free of charge for one year. Priced advantageously for groups: 1-\$59.50, 2-\$49.50 each, 3-\$39.50 each, 4-\$30.50 each, 5-9 \$25.50 each, 10 or more-\$19.50 each. Please include the name and address of each person ordering.

Send all inquiries for products on this page to: Walter H. Jopke Jr.  
5016 West 99th Street