In This Issue...

Editorial: History Repeats Itself ... I Hope Jim C. Warren, Jr. 3
Scanning the Industry Periodicals
Information derived from the May 24th issue of Electronic News 4

FEATURE ARTICLES
First Word on a Floppy-Disc Operating System
Command Language & Facilities Similar to DECSYSTEM-10 5

Hardware & Software for Speech Synthesis Lloyd Rice
Detailed discussion of techniques & hardware/software trade-offs 6

SYSTEMS SOFTWARE
MINOL—Tiny BASIC with Strings in 1.75K Bytes Erik T. Mueller
An outstanding implementation by a high school junior 9

System Monitor for 8080-Based Microcomputers Charlie Pack
Keyboard control over program loading, examination, modification & execution 18

DATA
Submitting items for publication 2
Reprint privileges 2
Subscription & information form 33
PCC Bookstore titles 35
TV Dazzler Contest 36
MINOL—Tiny BASIC with Strings in 1.75K Bytes

AN OUTSTANDING JOB
DONE BY A HIGH SCHOOL JUNIOR

Dear Mr. Warren:

May 1, 1976

I have a Tiny BASIC program running on my Altair that I think you might be interested in. I call it MINOL (mine-all). It fits in 1.75K memory. Unlike the other Tiny BASIC's, MINOL has a string-handling capability, but only single-byte, integer arithmetic and left-to-right expression evaluation.

Additions to TB include CALL machine-language subroutines, multiple statements on a line (like TBX), and optional "LET" in variable assignments. Memory locations of the form (H,L) can be used interchangably with variables, permitting DIM-like operations.

Sincerely,

Erik T. Mueller
36 Homestead Lane
Roosevelt NJ 08555

MINOL is an abbreviated form of BASIC with additional features. It has twelve statements: LET, PR, IN, GOTO, IF, CALL, END, NEW, RUN, CLEAR, LIST, and OS.

Variables: A letter from A to Z, or a memory location of the form (H,L), where H is the high address (decimal), and L is the low address. H and L may be expressions.

Number: An integer from 0 to 255.

Expression: A series of terms separated by arithmetic operators.

Terms: Numbers, variables, schars, random.

Schar: A single character enclosed in single quotes. Gives the ASCII value of the character.

Random: "!" (exclamation point) gives a random number between 0 and 255. (Subroutine by Jim Parker.)

Arithmetic Operators: +, -, *, /.

Relational Operators (not permitted in expressions):
= # < (“less than”) >= <= > (greater than)

Arithmetic Evaluation: All expressions are evaluated from left to right (no precedence of operations).

Statements: A statement consists of one or more sub-statements separated by ":" (colon), and terminated by CR.

Lines up to 72 characters. Line numbers from 1 to 254. All statements may be used with or without a line number. Statements without a line number are executed immediately. Statements with line numbers are edited into the existing program.

Substatements: [LET | ] <var> = <expr> Assigns the value of a variable. The "LET" can be left out if desired.

Ex: LET S = 0
    LET (24,0) = P-59
    A=B+C*J-198
    (25,5)=A/7/B

PR <var-list> [: ; ]
<var-list> : Literals, strings, or expressions separated by commas.

Literal: Characters to be printed enclosed in double quotes.

Strings: $(H,L): A series of memory locations starting at H,L which contain characters previously entered.

Expressions: Simple variable or expression.

Ex: PR"YOU SAY YOUR NAME IS"$,$(10,0)
    PR,A,B,(6,0),

PR 56+!A,B
PR
A semicolon at the end of a PR suppresses CRLF. A blank PR produces a CRLF.

PR Format: Numerical values are printed with one leading and trailing space and with all leading zeros suppressed. All strings and literals are printed without leading and trailing spaces. No zone spacing.

GOTO <expr>

Transfers control to the specified statement. GOTO 0 transfers control to beginning of unnumbered statement.

Ex: GOTO A*10
    GOTO 78

IF <expr> <rel> <expr> ; <statement>

Executes the statement following the ";" (semi-colon) if the specified relation is true. If it is untrue, control is transferred to the next statement on the line (if present).

Ex: IF X=5 ; GOTO 20
    IF A="Y" ; PR"SURE, WHY NOT?"
    IF A+B*C # 20 ; GOTO 20 ; PRA+B*C
    IF Y # 6 ; S=1

IN [<var> [<str>]] [,[<var> [<str>]]]*

This statement permits two data to be entered from the terminal: a) Numeric data; and b) Alphanumeric data; either a single letter, or a string of n characters.

Using a <var>: The input data is tested. If it is numeric, the number is deposited into the variable. If the data is not a number, the ASCII value of the first character typed is deposited.

Using a <str>: (of the form $(H,L) The inputted characters are deposited into memory sequentially starting at location H,L. 255 is placed in memory after the last character before CR. All spaces inputted are ignored unless enclosed by quotes. Note that (H,L) refers to a single location, but $(H,L) refers to a series of locations beginning at H,L. (H,L) can be used in expressions as a variable, but $(H,L) can only be used in I/O statements (IN, PR).

CALL (H,L)

Calls users subroutine starting at location H,L decimal.

END: Terminates program.

NEW: Deletes all lines of a program.

CLEAR: Sets all variables (A-Z) equal to zero.

RUN: Starts execution of program at lowest numbered statement.

LIST: Lists program in memory.

OS: Transfers control to user's operating system.

Line editing and correction:

Typing X^d deletes the last character typed.

X^e deletes an entire line.

X^c stops executing program.

Prints: BREAK AT LL (LL is the line that was to be executed before the interrupt occurred.)

To delete a line, type the line number followed by CR.

To change a line, type in the line with changes. The new line will replace the old one.

ERROR MESSAGES IERR L AT XX

1. Label does not exist
2. Input is over 72 characters.
3. Unrecognizable statement type.
4. Illegal variable.
5. Syntax error.
6. Out of memory.

EM MINOL 2.1 SYNTAX Apr. 1976

```
<line> ::= <number> <statement> or | <statement> or
<statement> ::= <substatement>* : <substatement>
<substatement> ::= [LET | φ] <var> = <expr>
               IN <var-str-list>
               IF <expr><rellop><expr> ; <statement>
               GOTO <expr>
               CALL <memloc>
               END
               NEW
               CLEAR
               OS
<number> ::= <digit> +2 <digit>
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<var> ::= A | B | ... | Y | Z | <memloc>
<rellop> ::= # = <
<expr-list> ::= [literal] [expr] [str] [, [literal] [expr] [str]]
<var-str-list> ::= [var] [str] [, [var] [str]]
<expr> ::= <term> <arop> <term>
<term> ::= [var] <arop> <number> | <schar> |
<literal> ::= "<char>"
<schar> ::= <char>
<str> ::= $ <memloc>
<memloc> ::= <higaddr> <loaddr>
<higaddr> ::= <number>
<loaddr> ::= <number>
<char> ::= any character except " and cr
```

Notes: < encloses an element of MINOL
      φ is the empty set
      * repeat limited by length of line
      +2 repeat from 0 to 2 times

MINOL

Memory Allocation:
(All locations are split octal)
000 000 - 000 115  I/O Routines, etc.
System Reset: 000 000 061 LXI SP
               000 377
               000 017
               000 317 RST CRLF
               004 303 JMP, MINOL
               005 116
               006 000
CRLF: 010 A subroutine to output a CR followed by a LF.
INPUT: 020 Moves a character from input device to the A regi-
        ster. Parity equals 1.
        Must output an echo check of the inputted character.
        No registers may be

000 116 - 006 377 MINOL Interpreter
000 253 L Highest memory location available
261 H for MINOL programs.
001 142 L Address of user's operating system,
143 H or monitor.
All input text is stored at 006 210 - 006 320
Free space is left for short strings at 006 333 - 006 377
Variables (A-Z) are stored at 005 007 - 005 042
007 000 + Program storage

Executing MINOL:
To start MINOL and initialize program area, EXAMINE 002 350, RUN.
To start without initialization, EXAMINE 000000, RUN.

Dear Jim:
May 24, 1976
I am enclosing the listing of MINOL—manually typed!
There are several features of my program, both positive and negative, that I might point out.
On the plus side, MINOL uses only 1.75K of memory, including the input-output subroutines (although since writing
it I see how I can make it even smaller.) Memory locations of the form (H,L) can be used similarly to one- or two-dimensional
DIMs in higher BASIC's. Simple input or output strings are possible by specifying a series of memory locations—of the
form S(H,L) where H,L is the first location where characters
are to be deposited. I am enclosing three programs to illustrate
these features.

On the negative side, the program is not designed for arithmetic functions, having no
 grouping of operations, and being limited to a value of 255.
The relational operators are restricted to =, #, and <, although
("greater than") can be done by reversing the logical expressions.
   Fewer error messages are provided than usual. MINOL is written completely in machine language without using IL.

When I can supply MINOL on a cassette I'll let you know. You might like to know that I am in my third year of
high school.
Yours truly,
Erik T. Mueller
Britton House
Roosevelt NJ 08555

Additions/changes since the May 1st letter:
Spaces are ignored:
a. During line/statement entry
   unless enclosed by quotes.
b. When inputting variables.
c. When inputting strings if the L
   address is zero.
Spaces are accepted:
a. When inputting strings if the L
   address is non-zero.
b. When enclosed by quotes.

Instead of GOSUB/RET statements, use the following substitute statements to perform the same function:
First initialize the GOSUB stack pointer Y:Z:
2 Y=14; Z=255 (Y and Z are the H,L address of
   some free space in memory.)
Instead of a GOSUB statement, substitute the
following: LET(Y,Z)=<RETURN label> :z=z-1:GOTO
<subroutine label>
Instead of a RET, substitute: Z=Z+1:GOTO(Y,Z)
Free space is left for very short user's strings from 006366 to 006377.

LIST
10 PR"GIVE ME A SENTENCE":IN$(14,1)
20 PR"STRING TO SEARCH FOR?":IN$(14,101)
21 A=Ø
22 A=A+1:IF(14,A)#255;GOTO22
23 B=Ø
24 B=B+1:IF(14,100+B)#255;GOTO24
30 C=1:D=1:S=Ø
40 IF(14,D+1ØØ)#(14,C);GOTO7Ø
50 D=D+1:C=C+1:IFD(B;GOTO4Ø
60 LETS=S+1
65 C=C-1
70 LETD=1
80 C=C+1:IFC(A;GOTO4Ø
90 PR"";$$(14,101);"" OCCURS";S;
95 IPS=1;GOTO1ØØ
97 PR"TIMES IN ";$$(14,1);"" :END
100 PR"TIME IN ";$$(14,1);"" :END

RUN
GIVE ME A SENTENCE
? THE BLUE BIRD IN THE BLUE SKY
STRING TO SEARCH FOR?
? BLUE
'BLUE' OCCURS 2 TIMES IN 'THE BLUE BIRD IN THE BLUE SKY'

LIST
10 "***NUMBER-A NUMBER GUESSING GAME (NUMØ5)
20 PR:PR:"WHAT IS YOUR NAME";:IN$(14,1)
30 X-1:S-Ø:PR:"HI,";$$(14,1);"" . WELCOME TO THE GAME OF NUMBER"
40 PR:"I'M THINKING OF A NUMBER FROM Ø TO 255"
50 PR:"GUESS MY NUMBER!"
60 PR:PR:"YOUR GUESS";:ING:S=S+1
65 IFX=X:GOTO90
70 IFX<Ø:PR:"TOO SMALL. TRY A BIGGER NUMBER."
80 IFX>Ø:PR:"TOO BIG. TRY A SMALLER NUMBER."
85 GOTO6Ø
90 PR:"THAT'S RIGHT,";$$(14,1);"!! YOU GOT IT IN";S;"GUESSES"
100 PR:"PLAY AGAIN";:INA:IFA='Y';GOTO3Ø
110 PR:"OK....HOPE YOU HAD FUN." :END

10 PR"NAME";:IN$(14,1)
20 IF(14,1)=J';IF(14,2)=I';IF(14,3)=M';PR"IT'S JIM!"
30 IF(14,1)#J';PR"IT'S NOT JIM.";GOTO1Ø

RUN
NAME?ERIK
IT'S NOT JIM.
NAME?JIM
IT'S JIM!
NAME?XØ
BREAK AT 1Ø
### MINOL 2.1

<table>
<thead>
<tr>
<th>TAG</th>
<th>ADDRESS</th>
<th>ADDRESS 11</th>
<th>ADDRESS 12</th>
<th>ADDRESS 13</th>
<th>MMENTHIC</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINOL</td>
<td>000116</td>
<td>076 335</td>
<td>MVI A &quot;J&quot;</td>
<td></td>
<td></td>
<td>Output prompt</td>
</tr>
<tr>
<td>000120</td>
<td>315 052 004</td>
<td>CAL, INPTXT</td>
<td></td>
<td></td>
<td></td>
<td>Get input line</td>
</tr>
<tr>
<td>000124</td>
<td>210 006</td>
<td>LXXH.LXTT</td>
<td></td>
<td></td>
<td></td>
<td>Point to input text with HL</td>
</tr>
<tr>
<td>000127</td>
<td>176</td>
<td>MOV A,M</td>
<td></td>
<td></td>
<td></td>
<td>Check for label</td>
</tr>
<tr>
<td>000130</td>
<td>315 062 005</td>
<td>CAL,CHEKIN</td>
<td></td>
<td></td>
<td></td>
<td>If no label, execute command</td>
</tr>
<tr>
<td>000133</td>
<td>322 000 001</td>
<td>JNC,DIRXLT</td>
<td></td>
<td></td>
<td></td>
<td>character NT</td>
</tr>
<tr>
<td>000136</td>
<td>043</td>
<td>INX HL</td>
<td></td>
<td></td>
<td></td>
<td>Convert ASCII label to binary</td>
</tr>
<tr>
<td>000137</td>
<td>176</td>
<td>MOV A,N</td>
<td></td>
<td></td>
<td></td>
<td>This section edits (inserts, deletes, changes) lines of the program</td>
</tr>
<tr>
<td>000140</td>
<td>315 062 005</td>
<td>CAL,CHEKIN</td>
<td></td>
<td></td>
<td></td>
<td>EXIL</td>
</tr>
<tr>
<td>000144</td>
<td>332 136 000</td>
<td>JC,FND</td>
<td></td>
<td></td>
<td></td>
<td>Lock at line number KILLINE</td>
</tr>
<tr>
<td>000146</td>
<td>315 256 004</td>
<td>CAL,MRSIN</td>
<td></td>
<td></td>
<td></td>
<td>olini</td>
</tr>
</tbody>
</table>
| 000151 | 021 000 007 | LXXH.DE11 | | | | \begin{tabular}{llll|l|}
| BBL |
| Point to line number greater than or equal to entered label |
| 000174 | 332 154 000 | JC,ZIP |
| 000176 | 176 | MOV A,M | | | | If label alone, delete line ARK |
| 000180 | 376 215 | CPI "CR" | | | | Count length of line and add 2 |
| 000201 | 006 002 | MVI C "02" | | | | If line entered already exists, first delete the old one, then insert the new one DIRECT |
| 000207 | 043 | INX HIL | | | | Delete old line RUN |
| 000212 | 376 215 | CPI "CR" | | | | HL points to first location LPUB |
| 000214 | 302 207 000 | JNZ, JIR | | | | where new line will be placed |
| 000220 | 270 | CMP B | | | | Save position in stack |
| 000222 | 231 000 | JXXH, EBFH | | | | Continue until DE points to end of file BIB |
| 000224 | 232 | PUSH DE | | | | Length of new line in A |
| 000225 | 315 351 000 | CAL,KILLS | | | | xxx = Low address: limit of program memory |
| 000230 | 151 | POP DE | | | | xxx = High address: memory limit |
| 000231 | 142 | MOV H,D | | | | Out of memory error |
| 000232 | 153 | MOV I,E | | | | | Increment until DE points to new end-of-file position, and HL points to where file updating begins |
| 000233 | 133 | MOV C,E | | | | BC points to end of file |
| 000234 | 023 | INX DE | | | | Increment until DE points to new end-of-file position, and HL points to where file updating begins |
| 000235 | 032 | POP PSHW | | | | BC points to end of file |
| 000236 | 376 XXX | CPI XXX | | | | |
| 000237 | 032 265 000 | JWX,HIL | | | | |
| 000238 | 072 | MOV A,D | | | | |
| 000239 | 066 | DEX A | | | | |
| 000240 | 312 224 004 | JZ,ERR6 | | | | |
| 000241 | 361 | POP PSHW | | | | |
| 000242 | 075 | DEC A | | | | |
| 000243 | 322 | LXH,RX | | | | |
| 000244 | 173 | MOV A,E | | | | |
| 000245 | 043 | INX HIL | | | | |
| 000246 | 365 | PUSH PSHW | | | | |
| 000248 | 173 | MOV A,E | | | | |
| 000250 | 376 XXX | CPI XXX | | | | |
| 000251 | 032 | POP PSHW | | | | |
| 000252 | 376 XXX | CPI XXX | | | | |
| 000254 | 302 | MOV A,D | | | | |
| 000255 | 066 | DEX A | | | | |
| 000256 | 322 | LXH,RX | | | | |
| 000257 | 173 | MOV A,E | | | | |
| 000258 | 043 | INX HIL | | | | |
| 000259 | 365 | PUSH PSHW | | | | |
| 000260 | 173 | MOV A,E | | | | |
| 000261 | 312 224 004 | JZ,ERR6 | | | | |
| 000262 | 361 | POP PSHW | | | | |
| 000263 | 075 | DEC A | | | | |
| 000264 | 322 | LXH,RX | | | | |
| 000265 | 173 | MOV A,E | | | | |
| 000266 | 043 | INX HIL | | | | |
| 000267 | 365 | PUSH PSHW | | | | |
| 000268 | 173 | MOV A,E | | | | |
| 000269 | 043 | INX HIL | | | | |
| 000270 | 365 | PUSH PSHW | | | | |
| 000271 | 173 | MOV A,E | | | | |
| 000272 | 043 | INX HIL | | | | |
| 000273 | 365 | PUSH PSHW | | | | |
| 000274 | 173 | MOV A,E | | | | |
| 000275 | 043 | INX HIL | | | | |
| 000276 | 365 | PUSH PSHW | | | | |
| 000277 | 173 | MOV A,E | | | | |
| 000278 | 043 | INX HIL | | | | |
| 000279 | 365 | PUSH PSHW | | | | |
| 000280 | 173 | MOV A,E | | | | |
| 000281 | 043 | INX HIL | | | | |
| 000282 | 365 | PUSH PSHW | | | | |
| 000283 | 173 | MOV A,E | | | | |
| 000284 | 043 | INX HIL | | | | |
| 000285 | 365 | PUSH PSHW | | | | |
| 000286 | 173 | MOV A,E | | | | |
| 000287 | 043 | INX HIL | | | | |
| 000288 | 365 | PUSH PSHW | | | | |
| 000289 | 173 | MOV A,E | | | | |
| 000290 | 043 | INX HIL | | | | |
| 000291 | 365 | PUSH PSHW | | | | |
| 000292 | 173 | MOV A,E | | | | |
| 000293 | 043 | INX HIL | | | | |
| 000294 | 365 | PUSH PSHW | | | | |
| 000295 | 173 | MOV A,E | | | | |
| 000296 | 043 | INX HIL | | | | |
| 000297 | 365 | PUSH PSHW | | | | |
| 000298 | 173 | MOV A,E | | | | |
| 000299 | 043 | INX HIL | | | | |
| 000300 | 365 | PUSH PSHW | | | | |
| 000301 | 173 | MOV A,E | | | | |
| 000302 | 043 | INX HIL | | | | |
| 000303 | 365 | PUSH PSHW | | | | |

### D.R. Dobbs' Journal of Computer Calisthenics & Orthodontics, Box 310, Menlo Park, CA 94025

April, 1976

**Copy editor's note:**
- Relocate file leaving space for new line. Retrieve pointer.
- Point to first non-numeric character. Put line no. in A store line no. in file Store text line in file
- Go back to monitor section Delete line
- If deleting line that does not exist, return
- Point to next line Relocate file, deleting line
- Direct execution of a statement Set LNE (current line no.)=0 Execute statement Run statement: Start from beginning of program. Get next statement.
- If not a new line, go execute statement If statement no.=377 (end of program), go back to monitor If not 377, store current line no. at LNE Check for keyboard program interruption Check for "m" in second column (variable assignment)
- If "m" in first column (memory location assignment) go to LET Check for "n" If not, go on
- CALL Statement CLEAR Statement If neither, report error
Check for "E" as in END.

Check for "indicating REM statement

Address of user's monitor

NXTE 001340  376 242  CPI "***
HR   001342  302 022 002  JNZ,VAR
MR   001343  043   INX HL
MR   001346  176   MOV A,M
MR   001347  376 342  CPI "**
MR   001351  312 366 001  JZ,MREND
MR   001354  315 043 005  CAL,TERM
MR   001357  332 217 004  JCR,ERR
MR   001362  347   RST OUT
MR   001363  303 345 001  JMP,HR
MR   001366  043   INX HL
MR   001367  176   MOV A,M
MR   001370  315 043 005  CALL,TERM
MR   001373  332 016 002  JCR,DCR
MR   001376  376 273  CPI ";"
MR   001379  302 217 004  JNZ,ERR
MR   001380  043   INX HL
MR   001381  176   MOV A,M
MR   001382  315 043 005  CALL,TERM
MR   001387  332 016 002  JCR,DCR
MR   001393  376 240  CPI "A"SP
MR   001394  347   RST OUT
MR   001395  176   MOV A,M
MR   001396  043   STX DE
MR   001397  043   INX HL
MR   001398  043   INX DE
MR   001404  376 274  CPI ";"
MR   001408  302 035 002  JNZ,ER
MR   001409  043   DCX HL
MR   001410  043   DCX DE
MR   001414  022   STAX DE
MR   001415  163 120 003  MOV A,C
MR   001416  376 268  CPI "SP
MR   001417  303 345 001  JMP,MREND
MR   001418  176   MOV A,M
MR   001419  171   MOV A,C
MR   001420  315 043 005  CALL,TERM
MR   001421  303 130 001  JMP,LPUB
MR   001422  251  CPI ";"
MR   001423  302 212 004  JNZ,ERR
MR   001426  053   DCX HL
MR   001429  250  CPI ";"
MR   001432  053   DCX HL
MR   001433  176   MOV A,M
MR   001434  376 250  CPI ";"
MR   001435  302 306 001  JNZ,JH
PR   001439  171   MOV A,C
MR   001440  365  PUSH PSW
MR   001441  315 062 006  CAL,VAL
MR   001442  361  POP PSW
MR   001443  002   STAX BC
MR   001444  002   STAX BC
MR   001445  303 013 001  JMP,LPUB
MR   001446  250  CPI ";"
MR   001447  302 013 001  JMP,LPUB
MR   001448  176   MOV A,M
MR   001449  333  INP
MR   001450  376 244  CPI "B"
MR   001451  312 247 002  JZ,STRIN
MR   001452  376 250  CPI "*

Check for literal end.
If not, go on.
Print text until "found
If terminator before closing quotes, print error
If end of statement without semicolon ";", go do CR
Get next thing to print
Check if string
Output leading space
Transfer expression text from expression buffer
Print expression's value
Get start address of string in BC. Print string of
Input statement
If sense switches down, print"**
Check for variable
Check for input string
Check for single memory
**Calculates the value of an expression stored in memory.**

**DE is the cursor.**

**Check for end of expression.**

**RTOP**

**Save Operation in stack.**

**Get Term/Factor**

**Single character value**

**Memory location**

**Random number**

**Constant (number)**

**Variable**

**Retrieve Operation**

**Add C=C+B**

**Input a line of 72 characters**

**Do not accept space if outside quotes**

**If X הוד红线**

**If X Español回 go back a character**

**Low top address**
If over 72 characters, report error

If over 72 characters, report error

Print "ERR" message

"AT*

Get value of ASCII Numbers

BCDBIN

BBD to BIN subroutine

Print text pointed to by DE

Print text pointed to by DE

Check if a character is a letter

Check for statement terminator (OR + 1)
MINOL Errata & Praise

Dear Jim: July 5, 1976
I have just received a letter from Joseph F. Gaffney listing a zillion errors or typos in the MINOL listing. Below is a list of the corrections that should be made. Apparently, the listing has been published, but I still haven't received the issue or any issues after the third. Please check with the subscription department for me. **ERRORS**. (Most of them were pointed out by Joseph F. Gaffney, 321 Lyndhurst Ave., Lyndhurst NJ 07071.)

Changes are underlined.

GSM 001106
002345 303 013 001

ACT 003123
003211 CPI "C"
003206 312 371 003
003317 320 315 003
003327 016 000 MVI C 0
003333 312 345 003

NXGT 003134
004005 176 MOV A M

INPTXT 004052
004135 302 144 004

CHEKN 005062
005256 312 276 005 JZ, INU
006027 376 243 CPI "#"
006353 311 RET
006053 322 013 001 JC, LPUBB
006056 303 041 001 JMP, EXEC
006101 302 066 006 JNZ, SHME

Sincerely yours,
Erik T. Mueller
36 Homestead Lane
Roosevelt NJ 08555

Thanks for the errata. Your subscription was entered on May 19th. Issues no. 4 and no. 5 were mailed a week and a half apart, about a month prior to your letter. I encourage you to complain to your local congressional reps (complaining to the Post Office appears to be useless). I also mailed an extra copy of the issue in which MINOL appeared, separately.

--JCW

Dear Mr. Warren: July 19, 1976
Erik Mueller's MINOL version of Tiny BASIC in the April issue is fantastic, and I'm really enjoying it! I relocated it to fit with my monitor (a modified 'JAMON' [MITS User's Group]), and it's running with a Model 33 Teletype. Some of the MINOL subroutines are useful in other programs as well, and are easily called (particularly useful is PRINTXT). MINOL is fun, certainly, but it is also very amazing (how can it be so smart and yet so small?).

There were a few typographical errors which were easy to correct. Corrections (at the original addresses) are shown below.

Address Was Change to
001/350 342 242
002/050 274 273
002/346 OMITTED 013,001
003/207 271 371
003/317 320 302
003/320 OMITTED 315,003
003/327 OMITTED 016,000
003/334 OMITTED 345,003
004/005 OMITTED 176
004/060 OMITTED 107
004/137 OMITTED 004
005/256 OMITTED 312
005/257 DISPLACED 276,005
006/353 OMITTED 311

As the program stands, the processor will enter an endless loop if you try to divide by zero. This doesn't hurt anything, but it does hang it up. To cure this, you might wish to add the following routine to test for division by zero. It adds Error 7.

Change:

003/326 315,000,004 CALL DIV0

DIV0 MOVAB 004/000 170 MOVE B TO A
OARAA 004/001 267 SET STATUS
MOVAC 004/002 171 MOVE C TO A;
STATUS UN-AFFECTED

MVIC 004/003 016,000 CLEAR C
RNZ 004/005 300 RETURN NOT
ZERO

MVIB 004/006 006,067 ERR 7'
JMP 004/010 303,226,004 JMP ERR

*This is my 'relocated' code. Any convenient locations will do.

Yours truly,

Phillip L. Hansford
6841 Haywood St.
Tujunga CA 91042

EUGENE STORE:
THE REAL OREGON COMPUTER CO.

For Bob: 4/26/76

Indeed we are running a store and would love it if you mentioned us. The store opened May 8.

Thanks,
John Montgomery
The Real Oregon Computer Co.
205 W 10th
Eugene OR 97401

NEW CLUB CONTACTS: VENTURA COUNTY COMPUTER SOCIETY

VCSS is a Chapter of the Southern California Computer Society. Its mailing address is P.O. Box 525, Port Hueneme, CA 93041. For more direct responses, contact their Secretary, Fred Moeckel, 4240 Harbor Blvd. No.208, Oxnard, CA 93030. 