Editor's Note: For more information about "Tiny Basic" and other public domain microcomputer software, see the letter bv Jim C. Warren in the Correspondence section of this issue.

January 1976 Tiny BASIC Calisthenics & Orthodontia Box 310, Menlo Park CA 94025

DESIGN NOTES FOR TINY BASIC

by Dennis Allison, happy Lady, & friends

(reprinted from *People's Computer Company* Vol. 4, No. 2)

SOME MOTIVATIONS

A lot of people have just gotten into having their own computer. Often they don't know too much about software and particularly systems software, but would like to be able to program in something other than machine language. The TINY BASIC project is aimed at you if you are one of these people. Our goals are very limited--to provide a minimal BASIC-like language for writing simple programs. Later we may make it more complicated, but now the name of the game is keep it simple. That translates to a limited language (no floating point, no sines and cosines, no arrays, etc.) and even this is a pretty difficult undertaking.

Originally we had planned to limit ourselves to the 8080, but with a variety of new machines appearing at very low prices, we have decided to try to make a portable TINY BASIC system even at the cost of some efficiency. Most of the language processor will be written in a pseudo language which is good for writing interpreters like TINY BASIC. This pseudo language (which interprets TINY BASIC) will then itself be implemented interpretively. To implement TINY BASIC on a new machine, one simply writes a simple interpreter for this pseudo language and not a whole interpreter for TINY BASIC.

We'd like this to be a participatory design project. This sequence of design notes follows the project which we are doing here at PCC. There may well be errors in content and concept. If you're making a BASIC along with us, we'd appreciate your help and your corrections.

Incidentally, were we building a production interpreter or compiler, we would probably structure the whole system quite differently. We chose this scheme because it is easy for people to change without access to specialized tools like parser generator programs.

THE TINY BASIC LANGUAGE

There isn't much to it. TINY BASIC looks like BASIC but all variables are integers There are no functions yet (we plan to add RND, TAB; and some others later). Statement numbers must be between 1 and 255 so we can store them in a single byte. LIST only works on the whole program. There is no FOR-NEXT statement. We've tried to simplify the language to the point where it will fit into a very small memory so impecunious. tyros can use the system.

The boxes shown define the language. The guide gives a quick reference to what we will include. The formal grammar defines exactly what is a legal TINY BASIC statement. The grammar is important because our interpreter design will be based upon it.

IT'S ALL DONE WITH MIRRORS------ OR HOW TINY BASIC WORKS

All the variables in TINY BASIC: the control information as to which statement is presently being executed and how the next statement is to be found, the returnaddressesof active GOSUBS----all this information constitutes the state of the TINY BASIC interpreter.

There are several procedures which act upon this state. One procedure knows how to execute any TINY BASIC statement. Given the starting point in memory of a TINY BASIC statement, it will execute it changing the state of the machine as required. For example,

```
100 LET S = A+6 (cr)
```

would change the value of S to the sum of the contents of the variable A and the interger 6, and sets the next line counter to whatever line follows 100, if the line exists.

A second procedure really controls the interpretation process by telling the line interpreter what to do. When TINY BASIC is loaded, this control routine performs some initialization, and then attempts to read a line of information from the console. The characters typed in are saved in a buffer, LBUF. It first checks to see if there is a leading line number. If there is, it incorporates the line into the program by first deleting the line with the same line number (if it is present) then inserting the new line if it is of nonzero length. If there is no line number present, it attempts to execute the line directly. With this strategy, all possible commands, even LIST and CLEAR and RUN are possible inside programs. 'Suicidal' programs are also certainly possible.

TINY BASIC GRAMMAR

The things in bold face stand for themselves. The names in lower case represent classes of things. '::=' is read 'is defined as'. The asterisk denotes zero or more occurences of the object to its immediate left. Parenthesis group objects. ϵ is the empty set. | denotes the alternative (the exclusive-or).

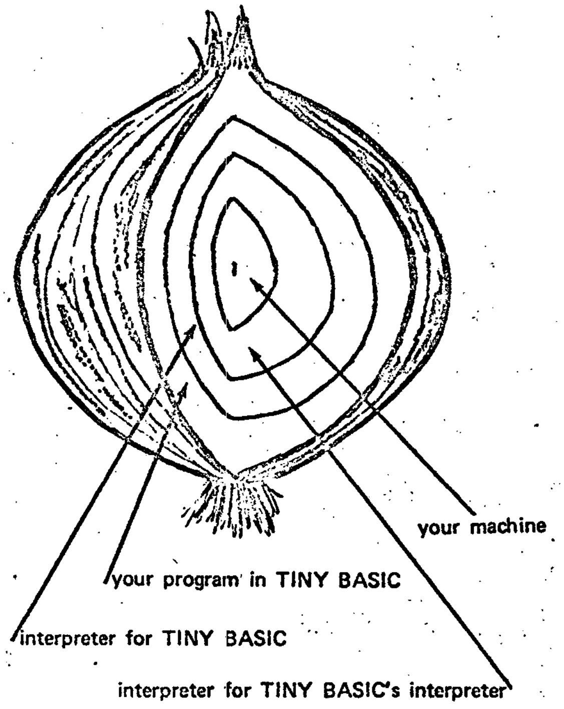
```
line::= number statement (cr) | statement (cr)
statement::= PRINT expr-list
             IF expression relop expression THEN statement
             GOTO expression
             INPUT var-list
             LET var = expression
             GOSUB expression
             RETURN
             CLEAR
             LIST
             RUN
             END
expr-list::= (string | expression) (, (string | expression) *)
var-list::= var (, var)*
expression::= (+|-|ε) term ((+|-) term)*
term::= factor ((* | /) factor)*
factor::= var | number | (expression)
var::= A | B | C ... | Y | Z
number::= digit digit*
digit::= 0| 1 | 2 |... | 8 | 9
relop::=< (>| = | ε) | >(< | = | ε) | =
```

A BREAK from the console will interrupt execution of the program.

IMPLEMENTATION STRATGIES AND ONIONS

When you write a program in TINY BASIC there is an abstract machine which is necessary to execute it. If you had a compiler it would make in the machine language of your computer a program which emulates that abstract machine for your program. An interpreter implements the abstract machine for the entire language and rather than translating the program once to machine code it translates it dynamically as needed. Interpreters are programs and as such have their's as abstract machines. One can find a better instruction set than that of any general purpose computer for writing a particular interpreter. Then one can write an interpreter to interpret the instructions of the interpreter which is interpreting the TINY BASIC program. And if your machine is microprogrammed (like PACE), the machine which is interpreting the interpreter interpreting the interpreter interpreting BASIC is in fact interpreted.

This multilayered, onion-like approach gains two things: the interpreter for the interpreter is smaller and simpler to write than an interpreter for all of TINY BASIC, so the resultant system is fairly portable. Secondly, since the major part of the TINY BASIC is programmed in a highly memory efficient, tailored instruction set, the interpreted TINY BASIC will be smaller than direct coding would allow. The cost is in execution speed, but there is not such a thing as a free lunch.


your machine
your program in TINY BASIC
interpreter for TINY BASIC
interpreter for TINY BASIC's interpreter

LINE STORAGE

The TINY BASIC program is stored, except for line numbers, just as it is entered from the console. In some BASIC interpreters, the program is translated into an intermediate form which speeds execution and saves space. In the TINY BASIC environment, the code necessary to provide the transformation would easily exceed the space saved.

QUICK REFERENCE GUIDE FOR TINY BASIC

LINE FORMAT AND EDITING

- Lines without numbers executed immediately
- Lines with numbers appended to program
- Line numbers must be 1 to 255
- Line number alone (empty line) deletes line
- Blanks are not significant, but key words must contain no unneeded blanks
- '←' deletes last character
- X^c deletes the entire line

EXECUTION CONTROL

CLEAR delete all lines and data
RUN run program
LIST list program

EXPRESSIONS

Operators

Arithmetic		Relational	
+	−	>	>=
*	/	<	<=
		=	<>,><

Variables
A.....Z (26 only)

All arithmetic is modulo 2^{15}
(±32762)

INPUT / OUTPUT

```
PRINT X,Y,Z
PRINT 'A STRING'
PRINT 'THE ANSWER IS'
INPUT X
INPUT X,Y,Z
```

ASSIGNMENT STATEMENTS

```
LET X=3
LET X= -3+5*Y
```

CONTROL STATEMENTS

```
GOTO X+10
GOTO 35
GOSUB X+35
GOSUB 50
RETURN
IF X>Y THEN GOTO 30
```

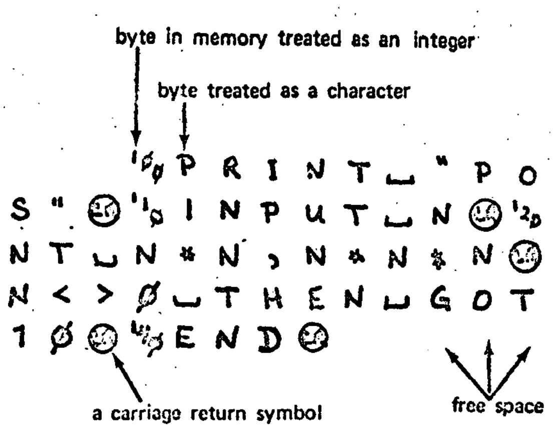
When a line is read in from the console device, it is saved in a 72-byte array called LBUF (Line BUFfer). At the same time, a pointer, CP, is maintained to indicate the next available space in LBUF. Indexing is, of course, from zero.

Delete the leading blanks. If the string matches the BASIC line, advance the cursor over the matched string and execute the next IL instruction. If the match fails, continue at the IL instruction labeled lbl.

The TINY BASIC program is stored as an array called PGM in order of increasing line numbers. A pointer, PGP, indicates the first free place in the array. PGP=0 indicates an empty program; PGP must be less than the dimension of the array PGM. The PGM array must be reorganized when new lines are added, lines replaced, or lines are deleted.

Insertion and deletion are carried on simultaneously. When a new line is to be entered, the PGM array searches for a line with a line number greater than or equal to that of the new line. Notice that lines begin at PGM (0) and at PGM (j+1) for every j such that PGM (j)=[carriage return]. If the line numbers are equal, then the length of the existing line is computed. A space equal to the length of the new line is created by moving all lines with line numbers greater than that of the line being inserted up or down as appropriate. The empty line is handled as a special case in that no insertion is made.

TINY BASIC AS STORED IN MEMORY


byte in memory treated as an integer
byte treated as a character
10 P R I N T ␣ " P O W E R
S " ☺ 11 I N P U T ␣ N ☺ 12 P R I
N T ␣ N * N , N * N * N ☺ 13 I F
N < > Ø ␣ T H E N ␣ G O T O ␣ 1
1 Ø ☺ 14 E N D ☺
a carriage return symbol
free space

ERRORS AND ERROR RECOVERY

There are two places that errors can occur. If they occur in the TINY BASIC system, they must be captured and action taken to preserve the system. If the error occurs in the TINY BASIC program entered by the user, the system should report the error and allow the user to fix his problem. An error in TINY BASIC can result from a badly formed statement, an illegal action (attempt to divide by zero, for example), or the exhaustion of some resource such as memory space. In any case, the desired response is some kind of error message. We plan to provide a message of the form:

! mmm AT nnn

where mmm is the error number and nnn is the line number at which it occurs. For direct statements, the form will be:

! mmm

since there is no line number.

Some error indications we know we will need are:

1 Syntax error
2 Missing line
3 Line number too large
4 Too many GOSUBs
5 RETURN without GOSUB
6 Expression too complex
7 Too many lines
8 Division by zero

THE BASIC LINE EXECUTOR

The execution routine is written in the interpretive language, IL. It consists of a sequence of instructions which may call subroutines written in IL, or invoke special instructions which are really subroutines written in machine language.

Two different things are going on at the same time. The routines must determine if the TINY BASIC line is a legal one and determine its form according to the grammar; secondly, it must call appropriate action routines to execute the line. Consider the TINY BASIC statement:

GOTO 100

At the start of the line, the interpreter looks for BASIC key words (LET, GO, IF, RETURN, etc.) In this case, it finds GO, and then finds TO. By this time it knows that it has found a GOTO statement. It then calls the routine EXPR to obtain the destination line number of the GOTO. The expression routine calls a whole bunch of other routines, eventually leaving the number 100 (the value of the expression) in a special place, the top of the arithmetic expression stack. Since everything is legal, the XFER operator is invoked to arrange for the execution of line 100 (if it exists) as the next line to be executed.

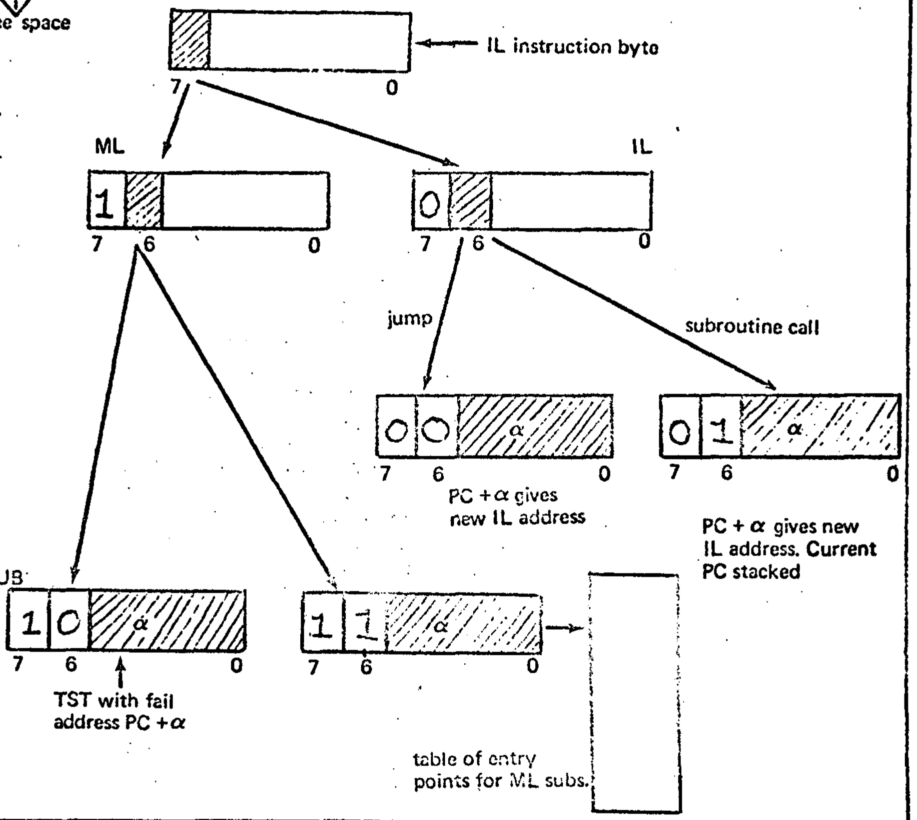
Each TINY BASIC statement is handled similarly. Some procedural section of an IL program corresponds to tests for the statement structure and acts to execute the statement.

ENCODING

There are a number of different considerations in the TINY BASIC design which fall in this general category. The problem is to make efficient use of the bits available to store information without loosing out by requiring a too complex decoding scheme.

In a number of places we have to indicate the end of a string of characters (or else we have to provide for its length somewhere). Commonly, one uses a special character (NUL = 00H for example) to indicate the end. This costs one byte per string but is easy to check. A better way depends upon the fact that ASCII code does not use the high order bit; normally it is used for parity.

ONE POTENTIAL IL ENCODING


IL instruction byte
7
0
ML
IL
1
7 6 0
0
7 6 0
jump
subroutine call
0 0 α
7 6 0
PC + α gives new IL address
0 1 α
7 6 0
PC + α gives new IL address. Current PC stacked
1 0 α
7 6 0
TST with fail address PC + α
1 1 α
7 6 0
table of entry points for ML subs.

on transmission. We can use it to indicate the end (that is, last character) of a string. When we process the characters we must AND the character with 07FH to scrub off the flag bit.

The interpreter opcodes can be encoded into a single byte. Operations fall into two distinct classes---those which call machine language subroutines, and those which either call or transfer within the IL language itself. The diagram indicates one encoding scheme. The CALL operations have been subsumed into the IL instruction set. Addressing is shown to be relative to PC for IL operations. Given the current IL program size, this seems adequate. If it is not, the address could be used to index an array with the ML class instructions.

TINY BASIC INTERPRETIVE OPERATIONS

TST lbl, 'string' delete leading blanks
If string matches the BASIC line, advance cursor over the matched string and execute the next IL instruction. If a match fails, execute the IL instruction at the labeled lbl.

CALL lbl Execute the IL subroutine starting at lbl. Save the IL address following the CALL on the control stack.

RTN Return to the IL location specified by the top of the control stack.

DONE Report a syntax error if after deletion leading blanks the cursor is not positioned to read a carriage return.

JMP lbl Continue execution of IL at the label specified.

PRS Print characters from the BASIC text up to but not including the closing quote mark. If a cr is found in the program text, report an error. Move the cursor to the point following the closing quote.

PRN Print number obtained by popping the top of the expression stack.

SPC Insert spaces to move the print head to next zone.

NLINE Output CRLF to Printer.

NXT If the present mode is direct (line number zero), then return to line collection. Otherwise, select the next sequential line and begin interpretation.

XFER Test value at the top of the AE stack to be within range. If not, report an error. If so, attempt to position cursor at that line. If it exists, begin interpretation there; if not report an error.

SAV Place present line number on SBRSTK. Report overflow as error.

RSTR Replace current line number with value on SBRSTK. If stack is empty, report error.

CMPR Compare AESTK(SP), the top of the stack, with AESTK(SP-2) as per the relation indicated by AESTK(SP-1). Delete all from stack. If condition specified did not match, then perform NXT action.

INNUM Read a number from the terminal and push its value onto the AESTK.

FIN Return to the line collect routine.

ERR Report syntax error and return to line collect routine.

ADD Replace top two elements of AESTK by their sum.

SUB Replace top two elements of AESTK by their difference.

NEG Replace top of AESTK with its negative.

MUL Replace top two elements of AESTK by their product.

DIV Replace top two elements of AESTK by their quotient.

STORE Place the value at the top of the AESTK into the variable designated by the index specified by the value immediately below it. Delete both from the stack.

TSTV lbl Test for variable (i.e. letter) if present.. Place its index value onto the AESTK and continue execution at next suggested location. Otherwise, continue at lbl.

TSTN lbl Test for number. If present, place its value onto the AESTK and continue execution at next suggested location. Otherwise, continue at lbl.

IND Replace top of stack by variable value if indexes.

LST list the contents of the program area.

INIT Performs global initialization
Clears program area, emptys GOSUB stack, etc.

GETLINE Input a line to LBUF.

TSTL lbl After editing leading blanks, look for a line number. Report error if invalid; transfer to lbl if not present.

INSRT Insert line after deleting any line with same line number.

XINIT Perform initialization for each stated execution.
Empties AEXP stack.

A STATEMENT EXECUTOR WRITTEN IN IL

This program in IL will execute a TINY BASIC statement. The operators TST, TSTV, TSTN, and PRS all use a cursor to find characteristics of the TINY BASIC line. Other operations (NXT, XFER) move the cursor so it points to another TINY BASIC line.

THE IL CONTROL SECTION

```
START;  INIT                    ;INITIALIZE
        NLINE                   ;WRITE CR/LF
CO:     GET LINE                ;WRITE PROMPT & GET A LINE
        TSTL       XEC          ;TEST FOR LINE NUMBER
        INSRT                   ;INSERT IT (MAY BE DELETE)
        JMP        CO
STMT:   XINIT                   ;INITIALIZE FOR EXECUTION
```

STATEMENT EXECUTOR

```
STMT:   TST     S1,'LET'        ;IS STATEMENT A LET?
        TSTV    S16             ;YES: PLACE VAR ADDRESS ON AESTK.
        CALL    EXPR            ;PLACE EXPR VALUE ON AESTK.
        DONE                    ;REPORT ERROR IF cr NOT NEXT.
        STORE                   ;STORE RESULT:
        NXT                     ;AND SEQUENCE TO NEXT.
S1:     TST     S3,'GO'         ;GOTO OR GOSUB?
        TST     S2,'TO'         ;YES...TO OR...SUB.
        CALL    EXPR            ;GET LABEL.
        DONE                    ;ERROR IF cr NOT NEXT.
        XFER                    ;SET UP AND JUMP.
S2:     TST     S14,'SUB'       ;ERROR IF NO MATCH.
        CALL    EXPR            ;GET DESTINATION.
        DONE                    ;ERROR IF cr NOT NEXT.
        SAV                     ;SAVE RETURN LINE.
        XFER                    ;AND JUMP.
S3:     TST     S8,'PRINT'      ;PRINT.
S4:     TST     S7,'"':         ;TEST FOR QUOTE.
        PRS                     ;PRINT STRING.
S5:     TST     S6,','          ;IS THERE MORE?
        SPC                     ;SPACE TO NEXT ZONE.
        JMP     S4              ;YES, JUMP BACK.
S6:     DONE                    ;NO, ERROR IF NO cr.
        NLINE
        NXT
S7:     CALL    EXPR            ;GET EXPR VALUE.
        PRN                     ;PRINT IT.
        JMP     S5              ;IS THERE MORE?
S8:     TST     S9,'IF'         ;IF STATEMENT.
        CALL    EXPR            ;GET EXPRESSION.
        CALL    RELOP           ;DETERMINE OPR AND PUT ON STK.
        CALL    EXPR            ;GET EXPRESSION.
        CMPR                    ;PERFORM COMPARISON--PERFORMS NEXT IF FALSE.
        JMP     STMT            ;GET NEXT STATEMENT.
S9:     TST     S12,'INPUT'     ;INPUT STATEMENT.
S10:    CALL    VAR             ;GET VAR ADDRESS.
        INNUM                   ;MOVE NUMBER FROM TTY TO AESTK.'
        STORE                   ;STORE IT.
        TST     S11,','         ;IS THERE MORE?
        JMP     S10             ;YES.
S11:    DONE                    ;MUST BE cr.
        NXT                     ;SEQUENCE TO NEXT.
S12:    TST     S13,'RETURN'    ;RETURN STATEMENT.
        DONE                    ;MUST BE cr.
        RSTR                    ;RESTORE LINE NUMBER OF CALL.[
        NXT                     ;SEQUENCE TO NEXT STATEMENT.
S13:    TST     S14,'END'
        FIN
S14:    TST     S15,'LIST'      ;LIST COMMAND.
        DONE
        LST
        NXT
S15:    TST     S16,'RUN'       ;RUN COMMAND.
        DONE
        NXT
S16:    TST S17,'CLEAR'         ;CLEAR COMMAND.
        DONE
        JMP START

S17:    ERR                     ;SYNTAX ERROR.

EXPR:   TST     E0,'-'
        CALL    TERM            ;TEST FOR UNARY -.
        NEG                     ;GET VALUE.
        JMP     E1              ;NEGATE IT.
E0:     TST     E1,'+'          ;LOOK FOR MORE.
        CALL    TERM            ;TEST FOR UNARY +.
E1:     TST     E2,'+'          ;LEADING TERM.
        CALL    TERM            ;SUM TERM.
        ADD
        JMP     E1
E2:     TST     E3,'-'          ;ANY MORE?
        CALL    TERM            ;DIFFERENCE TERM.
        SUB
        JMP     E3
E3: T2: RTN                     ;ANY MORE?

TERM:   CALL    FACT
TO:     TST     T1,'*'
        CALL    FACT            ;PRODUCT FACTOR.
        MPY
        JMP     TO
T1:     TST     T2,'/'          ;ANY MORE?
        CALL    FACT            ;QUOTIENT FACTOR.
        DIV
        JMP     TO

FACT:   TSTV    FO              ;VARIABLE.
        IND                     ;YES, GET THE VALUE.[
        RTN
F0:     TSTN    F1              ;NUMBER, GET ITS VALUE.
        RTN
F1:     TST     F2,'('          ;PARENTHESIZED EXPR.
        CALL    EXPR
        TST     F2,')'          ;MATCHING PARENTHESIS.
        RTN
F2:     ERR                     ;ERROR.

RELOP:  TST     RO,'='
        LIT     0               ;=
        RTN
RO:     TST     R4,'<'
        TST     R1,'='
        LIT     2               ;<=
        RTN
R1:     TST     R3,'>'
        LIT     3               ;<>
        RTN
R3:     LIT     1               ;<
        RTN
```

```
R4:     TST     S17,'>'
        TST     R5'='
        LIT     5               ;>=
        RTN
R5:     TST R6,'<'
        LIT 3                   ;<>
R6:     LIT 4                   ;>
        RTN
```

corrected

TINY BASIC IL

```
;
;  INTERPRETIVE LANGUAGE SUBROUTINES
;
EXPR:   TST     E0      ;TEST FOR UNARY '-'
        DB      '-' OR 2000
        ICALL   TERM    ;PUT TERM ON AESTK
        NEG             ;NEGATE VALUE ON AESTK
        HOP     E1      ;GO GET A TERM
;
E0:     TST     E01     ;TEST FOR UNARY '+'
        DB      '+' OR 2000
E01:    ICALL   TERM    ;PUT TERM ON AESTK
E1:     TST     E2      ;TEST FOR ADDITION
        DB      '+' OR 2000
        CALL    TERM    ;GET SECOND TERM
        ADD             ;PUT SUM OF TERMS ON AESTK
        HOP     E1      ;LOOP AROUND FOR MORE
;
E2:     TST     E3      ;TEST FOR SUBTRACTION
        DB      '-' OR 2000
        CALL    TERM    ;GET SECOND TERM
        SUB             ;PUT DIFFERENCE OF TERMS ON AESTK
        HOP     E1      ;LOOP AROUND FOR MORE
;
E3:     RTN             ;THIS CAN BE RECURSIVE
;
;
;
TERM:   ICALL   FACT    ;GET ONE FACTOR
T0:     TST     T1      ;TEST FOR MULTIPLICATION
        DB      '*' OR 2000
        ICALL   FACT    ;GET A FACTOR
        MPY             ;PUT THE PRODUCT ON AESTK
        HOP     T0      ;LOOP AROUND FOR MORE
;
T1      TST             ;TEST FOR DIVISION
        DB      '/' OR 2000
        ICALL   FACT    ;GET THE QUOTIENT
        DIV             ;PUT QUOTIENT ON AESTK
        HOP     T0      ;LOOP FOR MORE
;
T2      RTN             ;RETURN TO CALLER
;
;
;
FACT:   TSTV    F0      ;TEST FOR VARIABLE
        IND             ;GET INDES OF THE VARIABLE
        RTN
F0:     TSTN    F1      ;TEST FOR NUMBER
        RTN
F1:     TST     F1      ;ERROR IF ITS NOT A '('
        DB      '(' OR 2000
        ICALL   EXPR    ;THIS IS A RECURSIVE PROCESS
;
E1:    TST     FE1     ;EVERY '(' HAS TO HAVE A ')'
        DB      ')' OR 2000
        RTN
;
;
;
RELOP:  TST     R0      ;CHECK FOR '='
        DB      '=' OR 2000
        LIT     0
        RTN
;
R0:     TST     R4      ;CHECK FOR '<'
        DB      '<' OR 2000
        TST     R1
        DB      '=' OR 2000
        LIT     2
        RTN
;
R1:     TST     R3      ;CHECK FOR '>'
        DB      '>' OR 2000
        LIT     3
        RTN
;
R3:     LIT     1
        RTN
;
R4:     TST     R4
        DB      '>' OR 2000
        TST     R5
        DB      '=' OR 2000
        LIT     5
        RTN
;
R5:     TST     R6
        DB      '<'
        LIT     3
        RTN
;
R6:     LIT     4
        RTN
;
```

```
;
;
;  STATEMENT EXECUTOR WRITTEN IN IL (INTERPRETIVE LANGUAGE)
;  THIS IS WRITTEN IN MACROS FOR THE INTEL INTELEC 8/MOD 80
;  SYSTEM USING INTEL'S ASSEMBLER.
;
;  CONTROL SECTION
;
START:  INIT                    ;INITIALIZE
ERRENT: NLINE                   ;WRITE A CR-LF
CO:     GETLN                   ;WRITE PROMPT AND GET A LINE
        TSTL    XEC             ;IF NO LINE NUMBER GO EXECUTE IT
        INSRT                   ;INSERT OR DELETE THE LINE
        IJMP    CO              ;LOOP FOR ANOTHER LINE
XEC:    XINIT                   ;INITIALIZE FOR EXECUTION
;
;
;  STATEMENT EXECUTOR
;
STMT:   TST     S1      ;CHECK FOR 'LET'
        DB      'LE','T' OR 2000
SE1:    TSTV    SE1     ;ERROR IF NO VARIABLE!
SE2:    TST     SE2     ;ERROR IF NO "="
        DB      '=' OR 2000
        ICALL   EXPR    ;PUT EXPRESSION ON AESTK
        DONE            ;CHECK FOR CR LINE TERMINATOR
        STORE           ;PUT VALUE OF EXPRESSION IT ITS CELL
        NXT             ;CONTINUE NEXT LINE
;
;
S1:     TST     S3      ;CHECK FOR 'GO'
        DB      'G','O' OR 2000
        TST     S2      ;CHECK FOR 'GOTO'
        DB      'T','O' OR 2000
        ICALL   EXPR    ;GET THE LABEL
        DONE            ;CHECK FOR CR LINE TERMINATOR
        XFER            ;DO A 'GOTO' TO THE LABEL
;
;
S2:     TST     S2      ;CHECK FOR 'GOSUB', FAILURE IS AN ERROR!
        DB      'SU','B' OR 2000
        ICALL   XPER    ;PUT EXPRESSION ON AESTK
        DONE            ;CHECK FOR CR LINE TERMINATOR
        SAV             ;SAVE NEXT LINE NUMBER IN BASIC TEXT
        XFER            ;DO A 'GOSUB' TO THE LABEL
;
;
S3:     TST     S8      ;CHECK FOR 'PRINT'
        DB      'PRIN','T' OR 2000
S4:     TST     S7      ;CHECK FOR '"' TO BEGIN A STRING
        DB      '"' OR 2000
        PRS             ;PRINT THE DATA ENCLOSED IN QUOTES
S5:     TST     S6      ;',' MEANS MORE TO COME
        DB      ',' OR 2000
        SPC             ;SPACE TO NEXT ZONE
        HOP     S4      ;GO BACK FOR MORE
S6:     DONE            ;CHECK FOR CR LINE TERMINATOR
        NXT             ;CONTINUE NEXT LINE
;
;
S8:     TST     S9      ;CHECK FOR 'IF'
        DB      'I','F' OR 2000
        ICALL   EXPR    ;GET THE FIRST EXPRESSION
        ICALL   RELOP   ;GET THE RELATIONAL OPPERATOR
        ICALL   EXPR    ;GET THE SECOND EXPRESSION
S8A:    TST     S8A     ;CHECK FOR 'THEN'
        DB      'THE','N' OR 2000
        CMPR            ;IF NOT TRUE CONTINUE NEXT LINE
        IJMP    STMT    ;IF TRUE PROCESS THE REST OF THIS LINE
;
;
S9:     TST     S12     ;CHECK FOR 'INPUT'
        DB      'INPU','T' OR 2000
S10:    ICALL   VAR     ;GET THE VARIABLE'S INDEX
        INNUM           ;GET THE NUMBER FROM THE TELETYPE
        STORE           ;PUT THE VALUE OF THE VARIABLE IN ITS CELL
        TST     S11     ;',' MEANS MORE DATA
        DB      ',' OR 2000
S11:    DONE            ;CHECK FOR CR LINE TERMINATOR
        NXT             ;CONTINUE NEXT LINE
;
;
S12:    TST     S13     ;CHECK FOR 'RETURN'
        DB      'RETUR','N' OR 2000
        DONE            ;CHECK FOR CR  LINE TERMINATOR
        RSTR            ;RETURN TO CALLER
;
;
S13:    TST     S14     ;CHECK FOR 'END'
        DB      'EN','D' OR 2000
        FIN             ;GO BACK TO CONTROL MODE
;
;
S14:    TST     S15     ;CHECK FOR 'LIST'
        DB      'LIS','T' OR 2000
        DONE            ;CHECK FOR CR LINE TERMINATOR
        LST             ;TYPE OUT THE BASIC PROGRAM
        NXT             ;CONTINUE NEXT LINE
;
;
S15:    TST     S16     ;CHECK FOR 'RUN'
        DB      'RU','N' OR 2000
        DONE            ;CHECK FOR CR LINE TERMINATOR
        NXT             ;CONTINUE NEXT LINE
;
;
S16:    TST     S16     ;CHECK FOR 'CLEAR', FAILURE IS AN ERROR!
        DB      'CLEA','R' OR 2000
        IJMP    START   ;REINITIALIZE EVERYTHING!
;
;
***
```

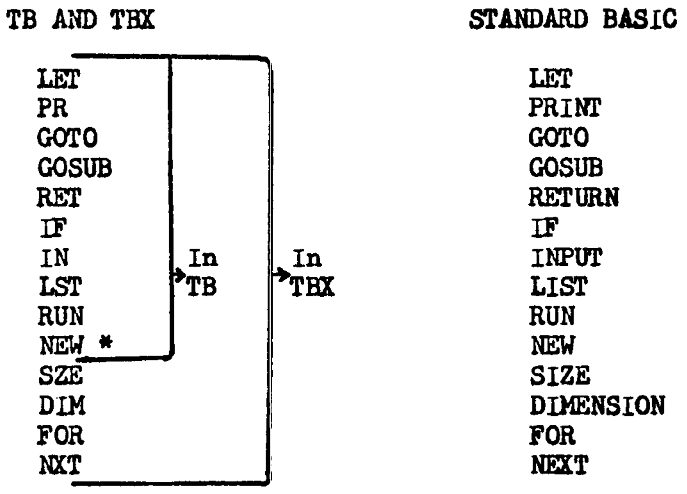
TINY BASIC, EXTENDED VERSION

by Dick Whipple (305 Clernson Dr., Tyler TX 75701)
& John Arnold (Route 4, Box 52-A, Tyler TX 75701)

INTRODUCTION

The version of TINY BASIC (TB) presented here is based on the design noted published in September 1975 *PCC* (Vol. 4, No. 2). The differences where they exist are noted below. In this issue we shall endeavor to present sufficient information to bring the system up on an Itel 8080-based computer such as the Altair 8800. Included is an octal listing of our ASCII version of TINY BASIC EXTENDED (TBX). In subsequent issues, structural details will be presented along with a source listing. A Suding-type cassette is now available from the authors (information to follow). We would greatly appreciate comments and suggestions from readers. Unlike some software people out there, we hope you *will* fiddle with TINY BASIC EXTENDED and make it *less Tiny*!

ABBREVIATED COMMAND SET



TB AND TBX	STANDARD BASIC
LET	LET
PR	PRINT
GOTO	GOTO
GOSUB	GOSUB
RET	RETURN
IF	IF
IN	INPUT
LST	LIST
RUN	RUN
NEW *	NEW
SZE	SIZE
DIM	DIMENSION
FOR	FOR
NXT	NEXT

*CLEAR in original TB

TBX -- HOW IT DIFFERS FROM TB

1. TBX system prompt is a colon ":".

2. Statement label values 1 to 65535.

3. Error correction during line entry:

   a) Rubout (ASCII 177_8) to delete a character. Prints a " ← ".

   b) Control L (Form Feed ASCII 014_8) to delete full line.

4. IN Statement: Termination of numeric input is accomplished by SPACE keystroke. All other terminations use CR (Carriage Return).

5. PR Statement: A comma is used for zone spacing while a semicolon produces a single space. A comma or semicolon at the end of a line surpresses CR and LF (Line Feed). To skip a line, use PR by itself.

6. DIM Statement: One or two dimensional arrays permitted. Array arguments can be expressions.

   Example:

```
10   LET V = 10
20   DIM A(10,10),B(2+V)
 . . .
```

   Array variables can be used in the same manner as ordinary variables.

7. FOR and NXT Statements: Step equal to 1 only. Iterative limits can be expressions. Nesting permitted. Care must be exercised when exiting a loop prior to completion of indexing. See Example.

   Example:

```
10    LET X = 10
20    FOR I = 1 to X
30    LET Y = 2 * A + B
40    IF Y = Z  I = X$NXT I$GOTO 60 *
50    NXT I
60    LET Y = 3
 . . .
```

   * For explanation of "$" see no. 9.

8. Available Functions:
   a) RN: Random number generator. Range 0 ≤ RN ≤ 10,000. No argument permitted.
   b) TB(E): Tab function. In a PR statement, TB(E) prints a number of SPACE's equal to the value of expression "E".

9. The dollar sign can be used to write multiple statement lines.

   Example:

```
10    IN B
20    LET A=2*(B+1)$PRA$END
```

   When using an IF statement, a "false" condition transfers execution to the next numbered line. Thus in line 40 of the example of no. 7, the chained statements will not be executed unless a "true" condition is encountered.

10. LST Command: Can take anyone of three forms:

    a) LST CR— lists all statements in program
    b) LST a CR— lists only statement labelled a
    c) LST a,b CR— lists all statements between labels a and b inclusive.

11. SZE Command: Prints two decimal numbers equal to:

    a) Number of memory bytes used by current program.
    b) Number of memory bytes remaining.

    Note: Array storage included only after first execution of program.

12. Recording Programs on Cassette: Core dumps to cassette should begin at 033350 (split octal) and continue through address stored at

    033354 (low byte of address)
    033355 (high byte of address)

    Of course these cassette programs should be loaded back at 033350.

IMPLEMENTING TBX

Memory Allocation:

I. Misc. Storage (I/O Routines) 000000 to 000377*

II. TBX 020000 to 033377

III. TBX Programs 034000 to upper limit of memory.

* In our system we maintain a Monitor/Editor in the first 1K byte of memory. 3/4 K is protected and 1/4 K can be used for system RAM. Such a configuration is useful but not necessary.

External Program Requirements:

1. **System Entry Routine —**

```
         ADRS              INST

         000000            061}
         000001            377}      LXI SP
         000002            000}
         000003            303}
         000004            254}      JMP TBX Entry Point
         000005            021}
```

The stack pointer (SP) must not be in protected memory. If you desire to relocate the SP change the following locations accordingly:

a) 000001 (SP low) and 000002 (SP high)
b) 026301 (SP low) and 026302 (SP high)

2. **System Recovery Routine —**

```
         ADRS              INST

         000070            303
         000071            000
         000072            000
```

3. Input Subroutine: Your input subroutine must begin at 000030. It should carry out the following functions:

a) Move an ASCII character from the input device to register A. The ASCII character should be right justified in A with Parity bit equal to zero. Example: "B" keystroke should set A to 102_8.

b) Test for ESC keystroke (ASCII 177_8) and jump if true to 000000. Suggested instructions

```
        . . .
        376}    CPI 'ESC'
        177}
        312}
        000}    JPZ System Entry Routine
        000}
        . . .
```

c) Output an echo check of the inputed character.

d) No registers should be modified except A.

4. Output Subroutine: Your output subroutine should begin at 000050. It should move the ASCII character in register A to the output device. Parity bit is zero. No registers including A should be modified.

5. CR-LF Subroutine: At 000020 you must have a subroutine that will output a CR followed by a LF. Only register A may be modified.

LOADING TBX:

The octal listing of TBX is reproduced later in the text. Addressing is split octal and gives the address of the first byte of each line. An octal loader of some kind is almost a necessity. Loading by front panel switches would be a considerable chore. A Suding-type cassette is available for $5, postpaid, from the authors. Send check or money order to: TBX Tape c/o John Arnold, Route 4, Box 52-A, Tyler TX 75701. If you are interested in a Baudot version of TBX, please inquire at the same address.

Use of a cassette tape to store TBX is virtually a necessity. Every effort has been made to protect TBX against self-destruction byt nothing is 100% sure!

The highest address available in your system for program storage must be loaded as follows:

026115	XXX$_8$	low part
026116	XXX$_8$	high part

Example: Suppose you have one 4K board; 026115 377
026116 037

EXECUTING TBX:

Simply examine 000000 and place the computer in the RUN mode. A colon indicates the system is operative.

ERROR MESSAGES

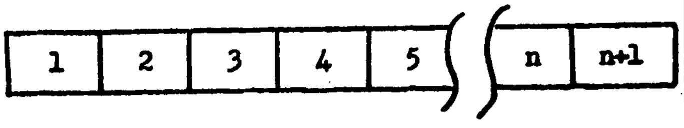
The form of error messages is: ERR $\alpha\,\beta$ where α is error number, and β is statement number where error was detected. Label 00000 indicates error occurred in direct execution.

ERROR NUMBER

1	Input line too long--exceeds 72 characters.
2	Numeric overflow on input.
3	Illegal character detected during execution.
4	No ending quotation mark in PR literal.
5	Arithmetic expression too complex.
6	Illegal arithmetic expression.
7	Label does not exist.
8	Division by zero not permitted.
9	Subroutine nesting too deep.
10	RET executed with no prior GOSUB
11	Illegal variable.
12	unrecognizable statement or command.
13	Error in use of parentheses.
14	Memory depletion.

EXAMPLE PROGRAM OF TBX

One example program written in TBX follows. It might assist you in debugging. A TBX line is structured as follows:



Byte No.

1 & 2	Binary value of label; most significant part in 1.
3	Length of text plus 2 in octal.
4 thru n	Text of line.
n + 1	CR (015_8).

After the last line you should find two 377s. At the end of the example run is an octal dump of the program area of memory.

EXAMPLE PROGRAM IN TBX

```
:NEW

:10 IN A
:20 PR"  TEST A IS ";A
:30 PR
:40 GOTO 10
:LST

00010 IN A
00020 PR"  TEST A IS ";A
00030 PR
00040 GOTO 10
:LST 20

00020 PR"  TEST A IS ";A
:LST 20,30

00020 PR"  TEST A IS ";A
00030 PR
:RUN

? 12    TEST A IS  12

? 356   TEST A IS  356

?
:
:DPO:034000 007,
034000  000 012 007 040 111 116 040 101
034010  015 000 024 025 040 120 122 042
034020  040 040 124 105 123 124 040 101
034030  040 111 123 040 042 073 101 015
034040  000 036 005 040 120 122 015 000
034050  050 012 040 107 117 124 117 040
034060  061 060 015 377 377 ~~107~~ ~~022~~ ~~000~~
:
```

TINY BASIC EXTENDED

OCTAL LISTING

```
020000  041 111 020 006 110 337 376 015
020010  312 036 020 376 177 312 040 020
020020  376 014 312 067 020 167 043 005
020030  312 306 026 303 005 020 167 311
020040  053 004 076 077 357 303 005 020
020050  332 000 021 076 057 276 322 000
020060  021 303 371 020 000 000 000 327
020070  076 072 357 076 015 062 007 020
020100  303 000 020 000 000 000 000 000
020110  000 114 123 124 040 066 060 060
020120  054 066 062 060 015 015 042 124
020130  105 123 124 061 042 044 120 122
020140  040 042 105 116 104 042 015 106
020150  117 122 040 122 117 127 040 042
020160  073 111 015 015 111 124 040 116
020170  117 122 105 040 114 111 116 105
020200  123 042 015 015 042 015 057 067
020210  062 010 000 000 000 000 000 000
020220  000 032 376 060 330 376 072 320
020230  346 017 311 000 000 000 000 000
020240  000 000 000 000 000 000 000 000
020250  000 000 000 000 000 000 000 000
020260  000 000 000 000 000 021 111 020
020270  325 032 376 040 023 312 271 020
020300  033 041 000 000 376 100 332 320
020310  020 042 350 033 000 321 311 000
020320  315 331 020 042 350 033 067 321
020330  311 315 221 020 376 012 320 023
020340  104 115 051 051 011 051 332 311
020350  026 117 006 000 011 303 331 020
020360  325 052 350 033 104 115 041 111
020370  020 076 071 043 276 303 050 020
021000  345 026 001 076 015 276 312 016
021010  021 024 043 303 005 021 172 062
021020  356 033 321 052 352 033 176 270
021030  312 052 021 322 064 021 043 043
021040  175 206 157 322 026 021 044 303
021050  026 021 043 176 271 312 170 021
021060  332 037 021 053 053 325 353 052
021070  354 033 345 072 356 033 306 003
021100  205 322 105 021 044 157 315 340
021110  030 104 115 341 176 002 053 013
021120  174 272 302 114 021 175 273 302
021130  114 021 023 052 350 033 353 162
021140  043 163 043 072 356 033 074 167
021150  043 321 032 167 376 015 312 166
021160  021 043 023 303 152 021 321 311
021170  053 345 043 043 043 176 376 015

021200  312 207 021 043 303 175 021 043
021210  353 052 354 033 043 104 115 341
021220  032 167 043 023 172 270 302 220
021230  021 173 271 302 220 021 053 042
021240  354 033 072 356 033 376 001 302
021250  361 020 321 311 041 002 032 176
021260  376 200 322 314 021 376 100 322
021270  300 021 043 156 147 303 257 021
021300  346 077 107 043 116 043 345 140
021310  151 303 257 021 376 300 322 000
021320  022 346 077 107 043 116 043 032
021330  023 376 040 312 327 021 033 325
021340  353 032 376 200 322 363 021 276
021350  043 023 312 341 021 321 140 151
021360  303 257 021 346 177 276 302 355
021370  021 353 301 023 043 303 257 021
022000  346 077 043 116 043 345 041 015
022010  022 345 147 151 351 341 322 257
022020  021 043 043 303 257 021 041 357
022030  033 357 043 065 300 066 017 311
022040  000 000 000 000 000 000 000 000
022050  000 000 000 000 000 000 000 000
022060  000 000 000 000 000 000 000 000
022070  000 000 000 000 000 000 000 000
022100  000 345 325 305 353 016 000 041
022110  020 047 315 147 022 041 350 003
022120  315 147 022 041 144 000 315 147
022130  022 041 012 000 315 147 022 173
022140  315 201 022 301 321 341 311 006
022150  377 004 173 225 137 172 234 127
022160  322 151 022 173 205 137 172 214
022170  127 170 271 310 015 315 201 022
022200  311 000 000 000 000 306 060 315
022210  026 022 311 325 052 306 033 053
022220  104 115 052 304 033 353 033 023
022230  327 170 272 302 243 022 171 273
022240  312 275 022 032 147 023 032 157
022250  315 205 026 023 023 032 376 015
022260  312 227 022 305 345 315 026 022
022270  341 301 303 254 022 321 311 000
022300  341 301 345 311 032 023 376 040
022310  312 304 022 033 376 015 310 303
022320  022 030 032 023 376 042 310 376
022330  015 312 317 026 315 026 022 303
022340  322 022 041 360 033 076 040 357
022350  065 302 345 022 066 017 247 311
022360  041 360 033 066 017 000 076 012
022370  357 227 311 000 311 052 350 033
```

```
023000  227 274 302 021 023 275 302 021
023010  023 041 004 032 301 343 305 247
023020  311 023 032 147 023 032 157 042
023030  350 033 023 023 301 041 022 032
023040  343 305 247 311 305 104 115 052
023050  361 033 160 043 161 043 042 361
023060  033 301 175 376 177 330 303 322
023070  026 305 052 361 033 053 106 053
023100  042 361 033 146 175 376 100 150
023110  301 320 303 325 026 174 057 147
023120  175 057 157 043 311 315 071 023
023130  174 267 362 147 023 315 115 023
023140  076 055 345 315 026 022 341 315
023150  101 022 247 311 345 052 352 033
023160  104 115 341 012 274 312 174 023
023170  320 303 204 023 003 012 275 312
023200  220 023 320 013 003 003 012 201
023210  117 322 163 023 004 303 163 023
023220  013 140 151 311 315 071 023 315
023230  154 023 353 312 022 023 303 330
023240  026 325 076 077 315 026 022 076
023250  040 357 062 007 020 315 000 020
023260  021 111 020 032 376 055 041 000
023270  000 312 312 023 315 331 020 315
023300  044 023 076 015 062 007 020 321
023310  247 311 023 315 331 020 315 115
023320  023 303 277 023 032 376 040 023
023330  312 324 023 033 306 300 320 007
023340  157 046 024 315 044 023 067 023
023350  311 032 376 040 023 312 351 023
023360  033 376 100 322 310 023 376 050
023370  310 041 000 000 303 124 024 000
024000  000 023 055 050 007 056 073 025
024010  000 001 002 000 001 000 001 000
024020  002 000 001 000 013 000 010 000
024030  000 000 000 000 070 000 025 000
024040  000 000 000 000 000 000 002 000
024050  324 046 004 000 002 000 001 000
024060  000 000 003 000 126 053 000 023
024070  016 000 004 000 000 023 000 023
024100  032 023 376 040 312 100 024 033
024110  376 015 310 376 044 310 303 314
024120  026 023 076 001 315 331 020 315
024130  044 023 311 315 071 023 106 043
024140  146 150 315 044 023 247 311 315
024150  071 023 114 105 315 071 023 160
024160  043 161 247 311 035 372 034 125
024170  023 321 076 001 311 023 000 023

024200  315 071 023 104 115 315 071 023
024210  011 315 044 023 247 311 315 071
024220  023 315 115 023 104 115 315 071
024230  023 011 315 044 023 247 311 000
024240  325 006 000 315 071 023 174 267
024250  374 301 024 353 315 071 023 174
024260  267 374 301 024 315 306 024 005
024270  314 115 023 315 044 023 321 247
024300  311 004 315 115 023 311 305 104
024310  115 041 000 000 076 021 062 363
024320  033 170 037 107 171 037 117 322
024330  333 024 031 174 037 147 175 037
024340  157 072 363 033 075 312 356 024
024350  062 363 033 303 321 024 140 151
024360  301 311 325 006 000 315 071 023
024370  174 267 374 301 024 353 315 071
025000  023 174 267 374 301 024 353 227
025010  274 302 020 025 275 312 333 026
025020  315 026 025 303 267 024 305 006
025030  001 174 346 100 302 044 025 051
025040  004 303 031 025 170 062 363 033
025050  104 115 041 000 000 173 221 137
025060  172 230 127 322 117 025 173 201
025070  137 172 210 127 051 072 363 033
025100  075 312 115 025 062 363 033 353
025110  051 353 303 055 025 301 311 051
025120  043 072 363 033 075 312 115 025
025130  303 104 025 315 071 023 315 115
025140  023 315 044 023 247 311 000 000
025150  000 325 315 071 023 353 315 071
025160  023 345 315 071 023 174 346 200
025170  302 262 025 172 346 200 302 227
025200  025 174 272 312 214 025 322 227
025210  025 303 224 025 175 273 312 232
025220  025 322 227 025 076 001 041 076
025230  004 041 076 000 341 021 242 025
025240  325 351 312 260 025 321 032 376
025250  015 312 375 022 023 303 246 025
025260  321 311 172 346 200 302 201 025
025270  303 224 025 376 000 311 376 001
025300  311 376 000 310 376 001 311 376
025310  001 310 376 004 311 376 004 311
025320  376 000 310 376 004 311 056 273
025330  001 056 276 001 056 301 001 056
025340  307 001 056 315 001 056 320 046
025350  025 315 044 023 247 311 305 104
025360  115 052 364 033 160 043 161 043
025370  042 364 033 301 175 376 177 330
```

```
026000  303 336 026 305 052 364 033 053
026010  106 053 042 364 033 146 175 376
026020  164 150 301 320 303 341 026 142
026030  153 315 356 025 247 311 315 003
026040  026 353 247 311 076 040 315 026
026050  022 247 311 000 000 000 041 077
026060  026 001 350 033 176 002 175 376
026070  033 310 003 043 303 064 026 000
026100  000 000 034 001 034 000 040 017
026110  100 030 000 164 024 377 057 000
026120  000 056 241 051 321 377 057 377
026130  377 041 100 030 042 361 033 041
026140  164 024 042 364 033 315 020 027
026150  052 352 033 126 043 136 353 000
026160  042 350 033 023 023 247 311 076
026170  015 357 303 360 022 327 076 017
026200  062 360 033 247 311 345 325 305
026210  353 016 377 303 107 022 000 000
026220  327 000 000 000 076 105 357 076
026230  122 357 357 076 040 357 046 000
026240  000 000 000 315 101 022 052 350
026250  033 076 040 357 315 205 026 016
026260  010 041 357 033 021 106 026 032
026270  167 015 302 267 026 041 002 032
026300  061 377 000 303 257 021 056 001
026310  001 056 002 001 056 003 001 056
026320  004 001 056 005 001 056 006 001
026330  056 007 001 056 010 001 056 011
026340  001 056 012 001 056 013 001 056
026350  014 001 056 015 001 056 016 001
026360  056 017 001 056 020 303 216 026
026370  000 000 000 000 000 000 000 000
027000  000 000 000 000 000 000 000 000
027010  000 000 000 000 000 000 000 000
027020  076 012 357 052 115 026 042 366
027030  033 311 325 315 071 023 353 315
027040  071 023 104 115 315 044 023 353
027050  315 044 023 321 305 315 240 024
027060  315 071 023 303 072 027 315 071
027070  023 345 051 104 115 052 366 033
027100  175 221 117 174 230 107 013 052
027110  354 033 274 302 120 027 171 275
027120  332 360 026 140 151 301 160 053
027130  161 104 115 042 366 033 315 071
027140  023 161 043 160 247 311 315 071
027150  023 053 051 104 115 315 071 023
027160  011 315 044 023 247 311 315 071
027170  023 053 315 044 023 052 370 033

027200  315 044 023 315 240 024 315 200
027210  024 303 146 027 032 023 376 040
027220  312 214 027 033 306 300 320 007
027230  117 023 032 376 050 312 243 027
027240  033 247 311 151 046 024 116 043
027250  146 151 116 043 106 043 315 044
027260  023 140 151 042 370 033 067 311
027270  300 325 342 000 000 000 000 000
027300  000 000 000 000 000 325 023 032
027310  376 015 302 064 030 353 315 044
027320  023 321 247 311 325 315 071 023
027330  345 116 043 106 315 071 023 353
027340  315 071 023 003 172 270 302 361
027350  027 173 271 322 361 027 303 006
027360  030 345 315 044 023 341 353 315
027370  044 023 341 315 044 023 140 151
030000  315 044 023 341 247 311 341 315
030010  044 023 140 151 315 044 023 321
030020  247 311 376 044 302 314 026 303
030030  033 023 032 376 040 023 312 032
030040  030 033 306 300 320 325 023 032
030050  306 300 321 320 376 015 310 376
030060  040 310 067 311 376 044 312 315
030070  027 303 306 027 347 323 241 324
030100  000 120 000 036 000 006 000 007
030110  000 010 000 012 000 000 000 030
030120  000 000 000 030 326 036 322 375
030130  230 141 001 014 211 326 167 323
030140  011 230 155 022 006 005 220 322
030150  304 322 213 322 375 230 166 012
030160  007 214 322 304 027 320 230 073
030170  014 001 223 322 304 027 300 000
030200  204 232 146 015 041 375 033 006
030210  010 176 007 007 007 256 027 027
030220  055 055 055 176 027 167 054 176
030230  027 167 054 176 027 167 054 176
030240  027 167 005 302 211 030 052 374
030250  033 174 346 077 147 376 047 312
030260  272 030 322 204 030 315 044 023
030270  077 311 175 376 020 303 262 030
030300  315 071 023 105 076 040 315 026
030310  022 005 302 304 030 063 063 063
030320  063 063 063 301 341 043 043 345
030330  305 073 073 073 073 073 073 311
030340  072 367 033 274 312 360 030 332
030350  360 026 042 354 033 311 000 000
030360  072 366 033 326 000 275 322 352
030370  030 303 360 026 000 000 000 000

031000  052 354 033 053 104 115 052 376
031010  033 011 345 052 366 033 104 115
031020  052 352 033 011 301 315 060 031
031030  315 101 022 076 040 357 052 366
031040  033 104 115 052 354 033 053 315
031050  060 031 315 101 022 327 247 311
031060  171 225 157 170 234 147 311 052
031070  352 033 042 304 033 052 354 033
031100  042 306 033 247 311 315 165 031
031110  042 304 033 043 043 076 015 043
031120  276 302 117 031 043 043 042 306
031130  033 247 311 000 315 165 031 043
031140  043 076 015 043 276 302 143 031
031150  043 043 042 306 033 315 165 031
031160  042 304 033 247 311 315 071 023
031170  315 154 023 310 303 330 026 000
031200  000 000 000 000 000 000 000 000
031210  000 000 000 000 000 000 000 000
031220  000 000 000 000 000 000 000 000
031230  000 000 000 000 000 000 000 000
031240  000 000 000 000 000 000 000 000
031250  000 000 000 000 000 000 000 000
031260  000 000 000 000 000 000 000 000
031270  000 000 000 000 000 000 000 000
031300  231 310 122 316 330 204 322 300
031310  232 330 124 302 132 343 330 300
031320  322 300 231 331 215 326 175 322
031330  375 232 210 244 326 175 323 034
031340  231 351 215 331 067 322 213 322
031350  375 132 343 231 366 254 132 343
031360  331 134 322 213 032 216 331 105
031370  322 213 032 216 047 041 066 010
032000  326 053 326 167 320 070 322 360
032010  320 265 032 022 320 360 032 004
032020  326 131 232 041 114 105 324 133
032030  310 132 340 324 147 322 304 322
032040  375 232 074 107 317 232 057 124
032050  317 132 343 322 304 323 224 232
032060  275 123 125 302 132 343 324 100
032070  326 027 323 224 232 112 111 306
032100  132 343 133 114 132 343 325 151
032110  032 022 233 326 106 117 322 323
032120  324 326 363 132 340 324 147 226
032130  363 124 317 327 305 132 343 322
032140  304 322 375 075 046 062 004 032
032150  232 226 120 322 231 322 242 322
032160  322 232 173 254 322 342 232 332
032170  215 322 375 232 202 273 326 044

032200  032 166 326 175 322 304 322 375
032210  132 343 323 125 032 161 322 304
032220  322 375 000 000 000 000 232 251
032230  111 316 133 310 323 241 324 147
032240  232 245 254 032 232 322 304 322
032250  375 232 264 122 105 324 326 036
032260  322 304 322 375 233 200 105 116
032270  304 326 167 323 011 232 306 114
032300  123 324 031 340 322 375 232 317
032310  122 125 316 322 304 032 020 233
032320  101 116 105 327 322 304 032 000
032330  326 347 232 154 244 323 034 000
032340  232 343 275 232 354 255 133 003
032350  325 133 032 361 232 357 253 133
032360  003 232 372 253 133 003 324 200
032370  032 361 233 055 255 133 003 324
033000  216 032 361 133 027 233 016 252
033010  133 027 324 240 033 005 233 055
033020  257 133 027 324 362 033 005 330
033030  032 033 035 031 300 327 214 033
033040  047 133 254 324 133 322 300 323
033050  324 033 057 324 133 322 300 323
033060  351 033 065 322 300 233 077 250
033070  132 343 233 077 251 322 300 326
033100  352 232 330 123 132 305 331 000
033110  032 216 000 000 233 123 275 325
033120  326 322 300 233 150 274 233 135
033130  275 325 334 322 300 233 144 276
033140  325 337 322 300 325 331 322 300
033150  232 330 276 233 162 275 325 345
033160  322 300 233 171 274 325 337 322
033170  300 325 342 322 300 000 000 000
033200  232 275 104 111 315 323 324 326
033210  352 226 355 250 132 343 233 241
033220  254 132 343 226 355 251 327 032
033230  233 235 254 033 205 322 304 322
033240  375 226 355 251 327 066 033 230
033250  000 000 000 000 226 355 250 132
033260  343 233 275 254 132 343 226 355
033270  251 327 166 322 300 226 355 251
033300  327 146 322 300 044 034 054 034
033310  327 214 033 320 133 254 322 300
033320  323 324 326 344 322 300 232 150
033330  116 130 324 323 324 326 352 327
033340  324 324 147 322 304 322 375 072
033350  000 000 000 034 054 034 004 040
033360  017 100 030 000 164 024 377 057
033370  000 000 056 241 051 321 377 057
```