ERRATA, UPDATES, AND CORRECTIONS

IT HAS COME TO OUR ATTENTION, THAT THERE MAY BE A SLIGHT MISUNDERSTANDING WITH RESPECT THE MANOR IN WHICH THE EDITOR WORKS. ON PAGE 9 OF THE MANUAL YOU SEE THAT THE EDITOR CAN EITHER 'CREATE' NEW FILES, OR EDIT OLD ONES. THE 'TINY' PASCAL SYSTEM IS OPERATED ENTIRELY FROM NORTH STAR'S 'DOS', EVEN THE EDITOR, SO IF YOU WANT TO EDIT A PREVIOUS FILE FROM DISC, YOU MUST FIRST LOAD THAT FILE AT 7200H, AND THEN 'GO EDIT', THE EDITOR WILL PROMPT YOU WHETHER YOU WANT TO CREATE A NEW FILE OR EDIT THE ONE YOU JUST PUT IN RAM. TO SAVE A FILE, YOU MUST SAVE IT MANUALLY, WITH THE 'SF' COMMAND, THE EDITOR TELLS YOU HOW MANY BLOCKS LONG IT IS, HENCE, IT IS JUST A MATTER OF CREATING THE FILE ON DISC, THEN SAVING THE FILE IN RAM STARTING AT 7200H TO THE DISC FILE. (SORRY FOR THE CONFUSION, HOWEVER, IT BECOMES VERY EASY ONCE YOU DO IT A COUPLE OF TIMES).

ALSO, IN THE LIST OF COMMANDS FOR THE EDITOR, THERE IS ONE MORE VERY IMPORTANT ONE: <cr>. A BLANK LINE WITH ONLY A CARRIAGE RETURN TAKES THE USER OUT OF INPUT MODE AND PUTS THE USER BACK IN COMMAND MODE WHERE THE USER MAY LIST THE FILE, MODIFY, EXIT, ETC.

AT THIS POINT, IT SHOULD BE SAID, THAT ALL! OPERATIONS TAKE PLACE THROUGH THE 'DOS'. AND ALL! ARE MANUAL, MEANING THAT YOU MUST CREATE FILES ON DISC, BOTH TO SAVE SOURCE, BUT ALSO, TO SAVE THE COMPILED PROGRAMS AS WELL, NONE(!!) OF THIS IS AUTOMATIC. (HOWEVER, THOSE AMBITIOUS ONES OUT THERE COULD NO DOUBT WRITE A SMALL OPERATING SYSTEM, IF ANY ONE DOES, LET US KNOW)

DUAL DENSITY USERS

IF YOU ARE MOVING UP TO DUAL DENSITY, THEN YOU WILL NEED A SPECIAL DUAL DENSITY VERSION, TO OBTAIN THIS, SIMPLY RETURN YOUR ORIGINAL 'TINY' PASCAL DISC, AND A CHECK FOR $10.00 FOR POSTAGE AND HANDLING TO SUPERSOFT.

GOOD LUCK, AND LET US KNOW OF ANY INTERESTING IMPROVEMENTS.

STATEMENT OF WARRANTY

SUPERSOFT DISCLAIMS ALL WARRANTIES WITH REGARD TO THE SOFTWARE CONTAINED ON DISC OR LISTED IN MANUAL, INCLUDING ALL WARRANTIES OF MERCHANTABILITY AND FITNESS ; AND ANY STATED EXPRESS WARRANTIES ARE IN LIEU OF ALL OBLIGATIONS OR LIABILITY ON THE PART OF SUPERSOFT FOR DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OF PERORMANCE OF THE SOFTWARE LICENSED.

TRANSFERABILITY

SUPERSOFT SOFTWARE AND MANUALS ARE SOLD ON AN INDIVIDUAL BASIS AND NO RIGHTS FOR DUPLICATION ARE GRANTED.

TITLE AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN WITH SUPERSOFT.

IT IS AND HAS ALWAYS BEEN SUPERSOFT'S BELIEF AND INTENTION TO PROVIDE EXCELLENCE IN BOTH DESIGN AND SERVICE. IT IS TO THESE ENDS WHICH WE DEDICATE OURSELVES.

OVER VIEW

Roughly speaking a COMPILER is a program that translates the statements of a high level language into an equivalent program of machine readable form. 'Tiny' PASCAL is a two stage compiler. It first translates the high level program into an intermediate file called: P-code. The P-code is then translated into 8080 machine readable form. The major advantage over an interpreter is speed, your 'Tiny' PASCAL programs will execute around 25 times faster than a normal BASIC interpreter!

This Program package also contains an editor, and a run time library. Also given, is the source, in 'tiny' PASCAL to these programs. This makes some very exciting things possible: you, the user has the power to modify, enhance, or change, the compiler to fit his\her specific need. You can add features, etc. We, at SUPERSOFT believe that this is perhaps the most exciting aspect of 'tiny' PASCAL.

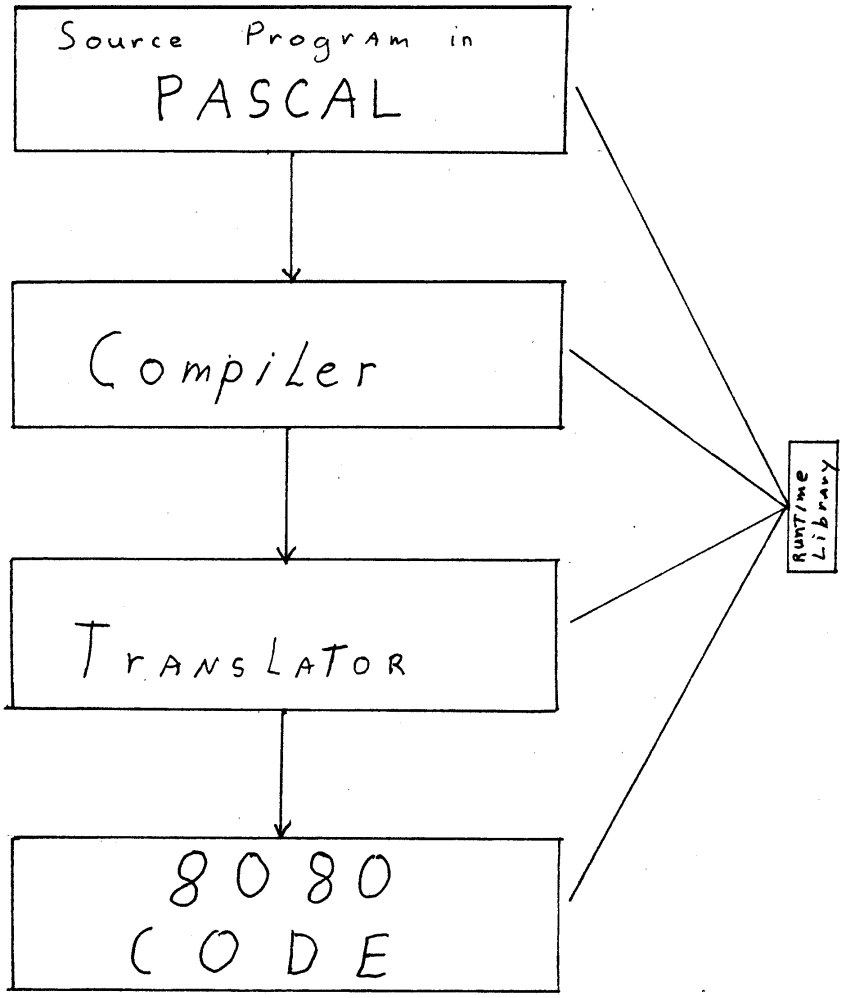
'TinY' PASCAL is a subset of standard PASCAL. Syntax is essentially identical to its larger brother. Syntax diagrams have been included for those who are just now learning the language. It must be emphasised that this manual is not an instructional text on PASCAL programming, but rather an explaination of the limits and special features of 'tiny' PASCAL. However, we will review some essential points in the next section.

For those who need a more thorough introduction, We recommend the the followins (partial) list of books:

Programming in PASCAL; Grogono
Addison-Wesley, 1978

PASCAL: User Manual and Report; Jensen and Wirth
Springer-Verlag, 1974

A Primer on PASCAL; Conway, Gries, and Zimmerman
Winthrop Publishers, 1976


Source Program in
PASCAL
Compiler
TRANSLATOR
8080
CODE
Runtime
Library

SOME 'TINY' PASCAL ESSENTIALS

1). ':=' is used for assignment and '=' is used for equality. They are not inter-changeable!

2). ';' is used to separate statements, not to end statments. Thus the last ';' in a compound statement:

```
BEGIN <STATEMENT>
    <STATMENT>;
    IF <EXP> THEN <EXP> ELSE <EXP>;
    <STATMENT>;
END
```

is not necessary. (It is, however, allowed since a PASCAL statement can be a null.) Note also the absence of ';' before an ELSE or an END in the 'CASE' statement.

3). Expressions may be either arithmatic or logical (BOOLEAN). Thus, the followins are perfectly legal:

```
A := B > C;
.
.
IF A+B THEN.......
```

note also that the boolean operator 'or' has the same precedence as the arithematic operator '+' and '-'. 'AND' the same as '*' and 'DIV' etc. It is important to remember that 'OR' and 'AND' have precedence over '=' ,'<', etc, thus the need for brackets at times as shown below:

```
IF (A < 10) AND (A > 5) THEN......
```

THE STATEMENT: IF A < 10 AND (A > 5) THEN.....

WOULD BE PARSED AS: IF A < (10 AND (A > 5)) THEN....
THUS PRODUCING THE UNDESIRABLE RESULT.

4). There are some context sensitive rules and meanings that cannot be inferred from the sytax diagrams, and may be particular to this implementation.

1). Identifier names must start with a letter and may be followed with letters or digits, but only the first 8 characters are significant.

2). Identifiers must be declared before used. Identifiers can be declared twice, but only the last one is used. Formal parameters of a procedure need not (and should not) be declared again inside the procedure.

3). Parameters are passed to procedures or functions by value, i.e. a copy of the value of the paramter by the program before the call.

4). The scope rules for identifiers are the same ones used by any block structure language. The scope of a variable is the procedure that contains it. An inner procedure can reference a variable in an outer procedure.

5). The only data types 'Tiny' PASCAL supports are integers and one dimisional integer arrays. The integers are sixteen bit signed. the arrays start at 0.

6). The meanins of certain operations is:

```
A DIV B     --> TRUNCATED INTEGER DIVISION : 27 DIV 5 = 5

A MOD B     --> A - (A DIV B)*B : 27 MOD 5 = 2

A SHL B     --> LEFT SHIFT A BY B :  27 SHL 2 = 54

A SHR B     --> RIGHT SHIFT A BY B : 27 SHR 2 = 13
```

7). The built in array MEM can be used to read to (if it appears in the left side of an assignment) or from (if it appears in an expression) to \ from a specified memory location, such as:

```
A := MEM [24467] +3;

MEM [T] := 0;
```

8). Hex constants are prefixed by '%'. (e.g. %2A00)

9). Strings are enclosed by single quote ('), not double. When a string appears in an expression or as a CASE lable, it has the value equal to the ASCII value of the first character of the string. When a string appears in the WRITE statement, the entire string would be outputed. Such as:

```
X := 'ABCD'          X WOULD = 'A' = 65
```

10). The READ and WRITE statemenst are character oriented, not line oreinted. More than one character can be placed in the same statement. Decimal numbers or Hex numbers can be read in from the keyboard by a '#' (decimal) or '%' (Hex) after the variable in the READ statement. Similarly, a decimal integer can be printed on the output device by following the expression with the appropriate '#' or '%' for Hex.

```
READ (A,B,C, I#,J%)
```

This would READ three (3) characters , a decimal number, and a Hex number.

```
A := 65
WRITE ('HELLO? ',A,' ',A#,' ',A%)
```

would print:

```
HELLO A 65 41
```

11). Since the READ is character oriented, it is necessary to terminate an integer input by a non-integer character. (such as a <cr> or <sp>). To input a hex number, four (4) digits must be typed.

12). To write on a new line, it is also necessary to output explicitly the ASCII code for <cr> and <lf> to the output device. That is , you must manually insert carriage return \ line feed. Such as:

```
WRITE ('THIS IS A TEST',13,10)
```

(HERE <CR> = 13, <LF> = 10)

13). An expression in the IF, WHILE, and REPEAT statements are said to fullfil the condition if the least significant bit is 1. This is equivalent to test that the expression is odd. Thus after:

```
IF X THEN A := 1 ELSE A := 100
```

A would have the value of 1 if x is odd, and 100 if x is even.

14). The relational operators (e.g. '=','=>'... etc) always produce a value of 0 or 1. Thus after :

```
A := X = 5;
```

A would have the value 1 if x equals 5 and 0 otherwise.

15). Comments are deliniated by '(*' and '*)' in addition to '{}'

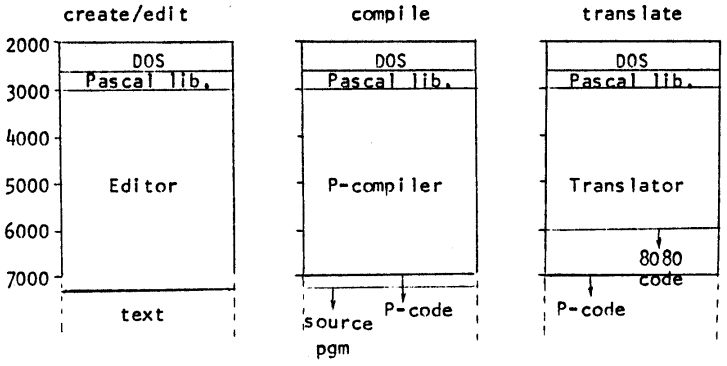
16). The source program can use tab (control I) characters, which are treated as groups of three spaces.

OPERATING INSTRUCTIONS

The editor and compiler can handle fairly large PASCAL progams, as large as the compiler itself. Minimum memory requirement is 24k (2000-7FFF). Only 36k Is need to compile the compiler. ALL OPERATIONS, INCLUDING THE COMPILER, TRANSLATOR, AND EDITOR USE NORTH STAR 'DOS' COMMANDS AND THE 'DOS' OPERATING SYSTEM.

System Start-up

After North Star DOS has been loaded, use the DOS command: 'GO LIB' to load the PASCAL run time library. This load from 2A00 to 2FFF. It contains all runtime routines, PASCAL Keywords, and P-code mnemonics. The diagram below show the memory maps during different steps of the program development.



The character input routine checks for a control-c, hence all developmental programs will terminate on a control-C.

The overflow message flag is OFF at initial loading. Thus all arithmetic overflows, such as add , subtract and negate, will not cause overflow messages to be printed. This flag can be turned on by placing a 1 at location 2A67H. Off, a 0.

To call the control-C routines from a PASCAL program, one must CALL(%2E66); IF MEM [%2A68] THEN.... (*CONDITION TRUE IF CTR-C TRUE*).

The P-Compiler (3000-6FFF)

To run the compiler type: 'LF <FILENAME> 7200', THEN 'GO COMP'. (If the compiler is executed directly after the editor , the source program file is still in memory and therefore, the LF command is not needed).

The user has to input the starting memory addresses of the source program and P-code. The source program is listed during compilation. The compiler will either:

1. Terminate normally if no error is found; the user should then proceed to translate the P-code. OR 2, abort execution with an error code at the FIRST occurrence of any compile error. (Refer to page 119 of Jensen and Wirth book: PASCAL User Manual and Report). The user should then go back and fix the problem.

Note: the P-code generated is usually of about the same size as the source program. The diagram on the page above shows how memory space can be saved by overlaying the source program with the P-code.

The Translator (3000-5FFF)

To run the Translator type: 'GO TRAN'. The user has to input the starting memory addresses of P-code and the translated 8080 machine code. nd the last memory address to be used by the runtime stack. The runtime stack starts at immediately following the 8080 code. The output 8080 code can be specified to fill at a different address, such as: 8080 code to execute at 3000 but placed at 6000 becuase code cannot be filled into the translator area 3000-5FFF during translation. After translation, the 8080 code can be saved on disc, the user is told the location

of the code, and the number of blocks, hence an appropriate file can be created using DOS and then the SF command to save the program.

Note: the the translated 8080 code is usually 95%-135% of the original P-code.

The Editor (3000-5FFF)

To execute the editor type, from DOS:

to create a file: 'GO EDIT'

to edit an existing disc file:

```
: LF <filename> 7200
: GO EDIT
```

This is a line-oriented text editor which does not use line numbers. The Maximum file size is only limited by available memory. Maximum number of lines is 800. Maximum number of characters per line is 80. All edtor commands are single character, some are immediately followed by a number xxx or a character string <string>. Invalid commands will be warned with the message 'ILLEGAL' and the bell runs (if terminal has one).

In the following commands, xxx means an integer number between 1 and 999; '^' usually refers to the top of line and '*' refers to the last, or bottom line. The line pointer always point sat the line most recently displayed or modified. After a Delete command , the line pointer is moved up 1 line.

If you want to edit an existing file you must first load that file into RAM from 'DOS' using the 'LF' command, load the file at 7200H. Then type 'GO EDIT', the editor prompts you whether to create a NEW file or EDIT an existing file, if you have loaded a file from disc, then you will want to type 'EDIT' otherwise type 'NEW'.

To save the source program you must use the 'SF' command form 'DOS'. Upon termination of editing, the editor will give the number of blocks the file is long, and you know the starting address is 7200H.

Below is a list of the editor commands, however one command is in need of fuller explaination: the <cr> command. Once you are in the INSERT mode you may enter text at will, to exit the INSERT mode and go back to the command mode you simply hit a carriage return for the

first character on an empty line. Next follows a list of the commands.

Editor Commands

<cr>		TERMINATE INERST MODE
LIST	L	LIST THE ENTIRE FILE
PRINT	P	PRINT THE CURRENT LINE
	Pxxx	PRINT THE SPECIFIED NUMBER OF LINES
	P^	PRINT THE TOP LINE
	P*	PRINT LAST LINE
REPLACE	R<string>	REPLACE CURRENT LINE BY <STRING>
APPEND	A<string>	APPEND <string> TO THE END OF CURRENT LINE
INSERT	I	ENTER INSER MODE (EXIT WITH <CR>
	I^	INSERT ABOVE TOP LINE
	I*	INSERT BELOW LAST LINE
DELETE	D	DELETE CURRENT LINE
	Dxxx	DELELT xxx LINES STARTING FROM CURRENT
	D^	DELELTE ALL LINES FROM TOP TO CURRENT
	D*	DELELTE ALL LINE FROM CURRENT TO LAST
MODIFY	M	ENTER INTRA-LIE EDITING MODE. SAME AS NORTH STAR BASIC.
STATUS	X	DISPLAY CURRENT FILE STATUS
UP	U	MOVE LINE POINTER UP 1 LINE
	Uxxx	MOVE LINE POINTER UP xxx LINES
NEXT	N	MOVE LINE POINTER TO NEXT LINE
	Nxxx	MOVE LINE POINTER xxx LINES
EXIT	E	RETURN TO DOS

Both the editor and the P-compiler recognize control-I as tab. The editor is written in 'tiny' PASCAL, not 8080 assembler. In general it is fairly efficient in speed. However, if the file is large, say more that 200 lines and 6k Bytes, it will take a few seconds to replace/append/insert/delete/or modify a line near the top of the file.

We know that you will enjoy using 'Tiny' PASCAL, and recommend that you 'play' with it a while just to get the 'hang' of it, and to become familiar with all of its features.

ERROR CODES

```
  1: ERROR IN SIMPLE TYPE
  2: IDENTIFIER EXPECTED
  3: 'PROGRAM' EXPECTED
  4: ')' EXPECTED
  5: ':' EXPECTED
  6: ILLEGAL SYMBOL
  7: ERROR IN PARAMETER LIST
  8: 'OF' EXPECTED
  9: '(' EXPECTED
 10: ERROR IN TYPE
 11: '[' EXPECTED
 12: ']' EXPECTED
 13: 'END' EXPECTED
 14: ';' EXPECTED
 15: INTEGER EXPECTED
 16: '=' EXPECTED
 17: 'BEGIN' EXPECTED
 18: ERROR IN DECLARATION PART
 19: ERROR IN FIELD-LIST
 20: ',' EXPECTED
 21: '*' EXPECTED

 50: ERROR IN CONSTANT
 51: ':=' EXPECTED
 52: 'THEN' EXPECTED
 53: 'UNTIL' EXPECTED
 54: 'DO' EXPECTED
 55: 'TO'/'DOWNTO' EXPECTED
 56: 'IF' EXPECTED
 57: 'FILE' EXPECTED
 58: ERROR IN FACTOR
 59: ERROR IN VARIABLE

101: IDENTIFIER DECLARED TWICE
102: LOW BOUND EXCEEDS HIGH BOUND
103: IDENTIFIER IS NOT OF APPR. CLASS
104: IDENTIFIER NOT DECLARED
105: SIGN NOT ALLOWED
106: NUMBER EXPECTED
107: INCOMPATIBLE SUBRANGE TYPES
```

108: FILE NOT ALLOWED HERE
109: TYPE MUST NOT BE REAL
110: TAGFIELD TYPE MUST BE SCALAR
111: INCOMPATIBLE WITH TAGFIELD TYPE
112: INDEX TYPE MUST NOT BE REAL
113: INDEX TYPE MUST BE SCALAR
114: BASE TYPE MUST NOT BE REAL
115: BASE TYPE MUST BE SCALAR
116: ERROR IN TYPE OF STANDARD PROCEDURE PARAMETER
117: UNSATISFIED FORWARD REFERENCE
118: FORWARD REFERNCE TYPE IDENTIFIER IN VARIABLE DELCLARATION
119: FORWARD DECLARED; REPITITION NOT ALLOWED
120: FUNCTION RESULT TYPE MUST BE SCALAR
121: FILE VALUE PARAMETER NOT ALLOWED
122: FORWARD DECLARED FUNCTION, REPETITION NOT ALLOWED
123: MISSING RESULT TYPE IN FUNCTION DECLARATION
124: F-FORMAT FOR REAL ONLY
125: ERROR IN TYPE OF STANDARD FUNCTION PARAMETER
126: NUMBER OF PARAMETERS DOES NOT AGREE WITH DECLARATION
127: ILLEGAL PARAMETER SUBSTITUTION
128: RESULT TYPE OF PARAMETER FUNCTION DOES NOT AGREE WITH DECLARATION
129: TYPE CONFLICT OF OPERANDS
130: EXPRESSION IS NOT OF SET TYPE
131: TESTS ON EQUALITY ALLOWED ONLY
132: STRICT INCLUSION NOT ALLOWED
133: FILE COMPARISION NOT ALLOWED
134: ILLEGAL TYPE OF OPERAND
135: TYPE OF OPERAND MUST BE BOOLEAN
136: SET ELEMENT TYPE MUST BE SCALAR
137: SET ELEMENT TYPES NOT COMPATIBLE
138: TYPE OF VARIABLE IS NOT ARRAY
139: INDEX TYPE IS NOT COMPATIBLE WITH DECLARATION
140: TYPE OF VARIABLE IS NOT RECORD
141: TYPE OF VARIABLE MUST BE FILE OR POINTER
142: ILLEGAL PARAMETER SUBSTITUTION
143: ILLEGAL TYPE OF LOOP CONTROL VARIABLE
144: ILLEGAL TYPE OF EXPRESSION
145: TYPE CONFLICT
146: ASSIGNMENT OF FILES NOT ALLOWED
147: LABEL TYPE INCOMPATIBLE WITH SELECTING EXPRESSION
148: SUBRANGE BOUNDS MUST BE SCALAR
149: INDEX TYPE MUST MOT BE INTEGER
150: ASSIGNMENT TO STANDRD FUNCTION IS NOT ALLOWED
151: ASSIGNMENT TO FORMAL FUNCTION IS NOT ALLOWED
152: NO SUCH FIELD IN THIS RECORD
153: TYPE ERROR IN READ
154: ACTUAL PARAMETER MUST BE A VARIABLE
155: CONTROL VARIABLE MUST NEITHER BE FORMAL NOR NON LOCAL
156: MULTIDEFINED CASE LABEL
157: TOO MANY CASES IN CASE STATEMENT
158: MISSING CORRESPONDING VARIANT DECLARATION
159: REAL OR STRING TAGFIELDS NOT ALLOWED
160: PREVIOUS DECLARATION WAS NOT FORWARD
161: AGAIN FORWARD DECLARED

162: PARAMETER SIZE MUST BE CONSTANT
163: MISSING VARIANT IN DECLARATION
164: SUBSTITUION OF STANDARD PROC/FUNC NOT ALLOWED
165: MULTIDEFINED LABEL
166: MULTIDECLARED LABEL
167: UNDELARED LABEL
168: UNDEFINED LABEL
169: ERROR IN BASE SET
170: VALUE PARAMETER EXPECTED
171: STANDARD FILE WAS REDECLARED
172: UNDECLARED EXTERNAL FILE
173: (NOT RELAVENT)
174: PASCAL PROCEDURE OR FUNCTION EXPECTED
175: MISSING INPUT FILE
176: MISSING OUTPUT FILE

201: ERROR IN RREAL CONSTANT: DIGIT EXPECTED
202: STRING CONSTANT MUST NTO EXCEEED SOURCE LINE
203: INTEGER CONSTANT EXCEEDS RANGE
204: (NOT RELAVENT)

250: TOO MANY NESTED SCOPES OF IDENTIFIERS
251: TOO MANY NESTED PROCEDURES AND OR FUNCTIONS
252: TOO MANY FORWARD REFERENCES OF PROCEURE ENTRIES
253: PROCEURE TOO LONG
254: TOO MANY LONG CONSTANTS IN THIS PROCEDURE
255: TOO MANY ERRORS ON THIS SOURCE LINE
256: TOO MANY EXTERNAL REFERENCES
257: TOO MANY EXTERNALS
258: TOO MANY LOCAL FILES
259: EXPRESSION TOO COMPLICATED

300: DIVISION BY ZERO
301: NO CASE PROVIDED FOR THIS VALUE
302: INDEX EXPRESSION OUT OF BOUNDS
303: VALUE TO BE ASSIGNED IS OUT OF BOUNDS
304: ELEMENT EXPRESSION OUT OR RANGE

398: IMPLEMENTATION RESTRICTION
399: VARIABLE DIMENSION ARRAYS NOT IMPLEMENTED

Sample

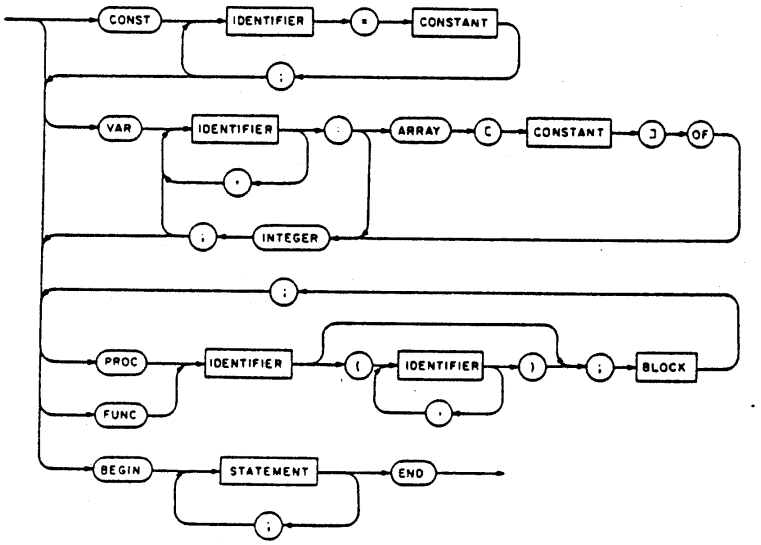
```
*GO LIB
*LF SAMPLE 7000
*GO COMP
SOURCE ADDR:7000
P-CODE ADDR:7400
0  { PASCAL TRIANGLE - V.1 BY H. YUEN 12/13/78 }
0 CONST COL=4;
1 VAR   N,I,J,SPACE:INTEGER;
1       A,B:ARRAY[14] OF INTEGER;
1
1 PROC CRLF; BEGIN WRITE(13,10) END;
7
8 PROC PRINT(NUM,IC);
8   VAR I,J:INTEGER;
9   BEGIN J:=NUM; I:=0;
13     REPEAT I:=I+1; J:=J DIV 10 UNTIL J=0;
24     IF I>IC THEN FOR I:=1 TO IC DO WRITE('*')
38     ELSE BEGIN FOR J:=1 TO IC-I DO WRITE(' ');
56             WRITE(NUM#) END
63   END;
63
64 BEGIN (* MAIN PROGRAM *)
65   WRITE('TO GENERATE A PASCAL TRIANGLE'); CRLF;
96   REPEAT WRITE('INPUT N (<14)'); READ(N#); CRLF
114   UNTIL (N>0) AND (N<=13);
121   SPACE:=N*COL DIV 2; A[0]:=0; A[1]:=1;
134   FOR J:=1 TO N DO
137    BEGIN SPACE:=SPACE-COL DIV 2;
145     FOR I:=0 TO SPACE DO WRITE(' ');
157     FOR I:=1 TO J DO B[I]:=A[I-1]+A[I]; A[J+1]:=0;
187     FOR I:=1 TO J DO BEGIN PRINT(B[I],COL); A[I]:=B[I] END;
204     CRLF END; CRLF
215 END.
216 217 P-CODES (7400-7764) 4 BLOCKS
*GO TRAN
** P-CODE TO 8080 CONVERSION **
P-CODE AT: 7400
8080 CODE STARTS: 7800   FILLED AT: 7800
STACK ENDS AT: 7FFF
19 LABELS   9 FWD REF
217 P-CODE   851 BYTES 8080 CODE (7800-7B52) 4 BLOCKS
8080:P-CODE = 98%
*JP 7800
TO GENERATE A PASCAL TRIANGLE
INPUT N (<14)?9
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5  10  10   5   1
        1   6  15  20  15   6   1
      1   7  21  35  35  21   7   1
    1   8  28  56  70  56  28   8   1

*
```

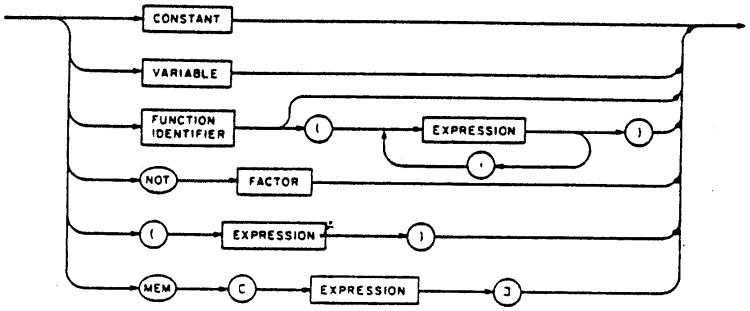
SYNTAX DIAGRAMS

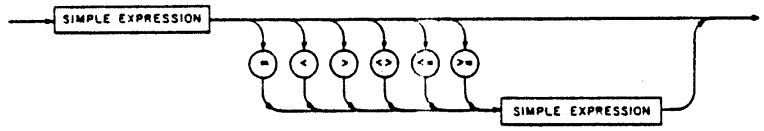
PROGRAM

BLOCK

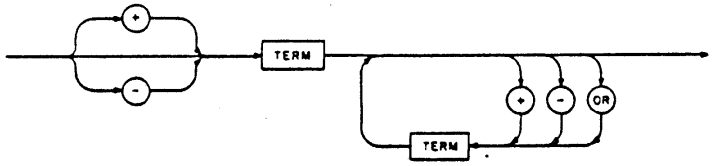


FACTOR

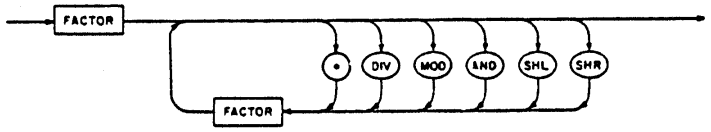




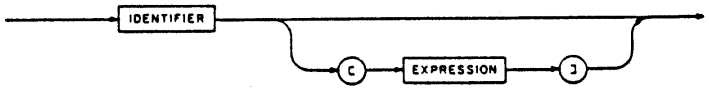
SIMPLE EXPRESSION



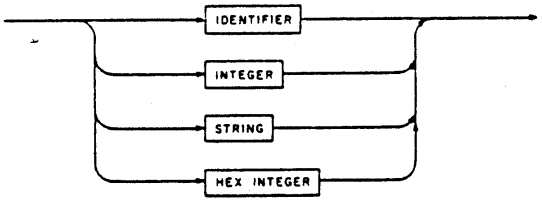
TERM



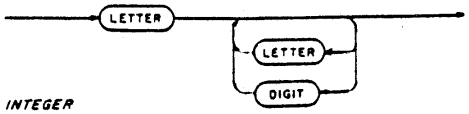
VARIABLE



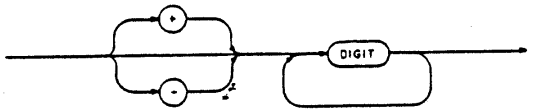
CONSTANT



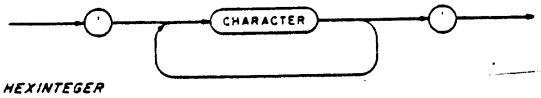
IDENTIFIER



INTEGER

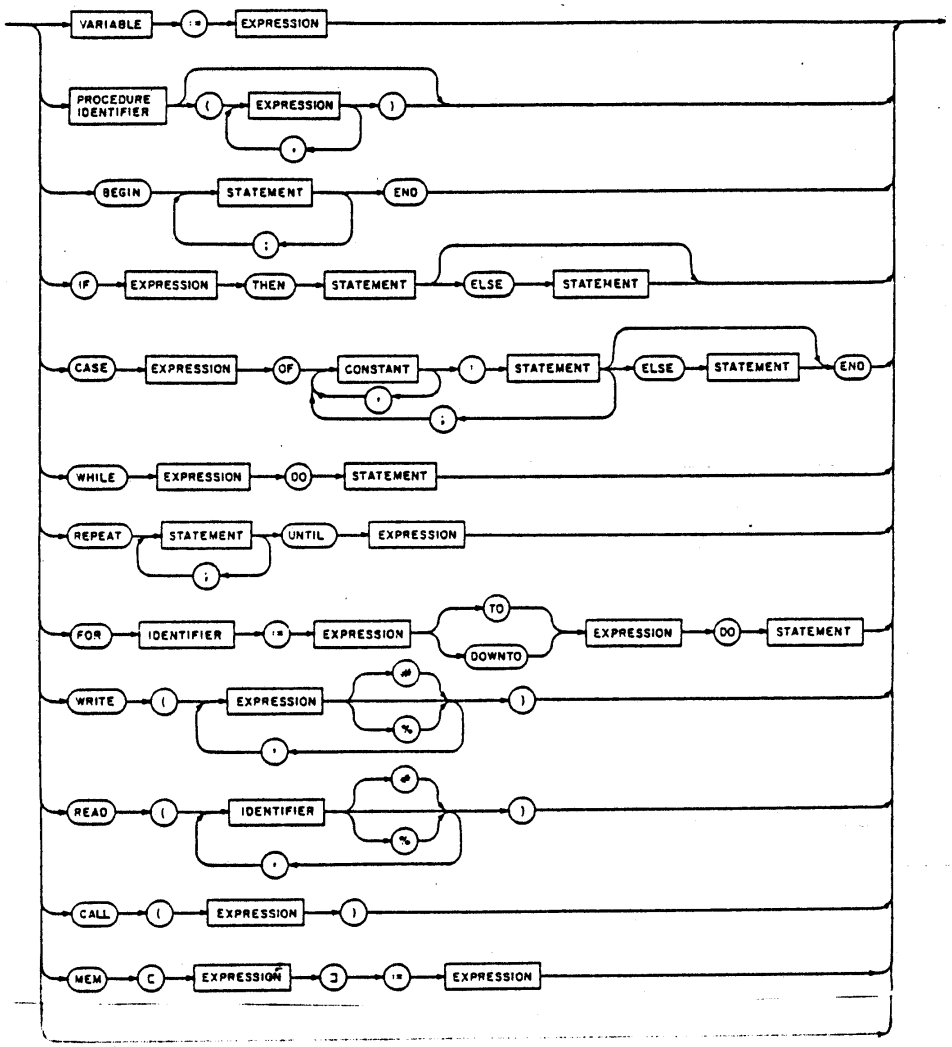


STRING



HEXINTEGER

STATEMENT


VARIABLE
:=
EXPRESSION
PROCEDURE IDENTIFIER
(
EXPRESSION
,
)
BEGIN
STATEMENT
;
END
IF
EXPRESSION
THEN
STATEMENT
ELSE
STATEMENT
CASE
EXPRESSION
OF
CONSTANT
,
:
STATEMENT
;
ELSE
STATEMENT
END
WHILE
EXPRESSION
DO
STATEMENT
REPEAT
STATEMENT
;
UNTIL
EXPRESSION
FOR
IDENTIFIER
:=
EXPRESSION
TO
DOWNTO
EXPRESSION
DO
STATEMENT
WRITE
(
EXPRESSION
,
#
%
)
READ
(
IDENTIFIER
,
#
%
)
CALL
(
EXPRESSION
)
MEM
[
EXPRESSION
]
:=
EXPRESSION