```
05020                   DSCTMP, OR ELSE THEY WILL BE COPIED BEFORE
05040                   THEY ARE PRINTED.

05080           FNDFOR
05100                   MOST SMALL ROUTINES ARE FAIRLY SIMPLE
05120                   AND ARE DOCUMENTED IN PLACE. FNDFOR IS
05140                   USED FOR FINDING "FOR" ENTRIES ON
05160                   THE STACK. WHENEVER A "FOR" IS EXECUTED AN
05180                   18 BYTE ENTRY IS PUSHED ONTO THE STACK.
05200                   BEFORE THIS IS DONE, HOWEVER, A CHECK
05220                   MUST BE MADE TO SEE IF THERE
05240                   ARE ANY "FOR" ENTRIES ALREADY ON THE STACK
05260                   FOR THE SAME LOOP VARIABLE. IF SO, THAT "FOR" ENTRY
05280                   AND ALL OTHER "FOR" ENTRIES THAT WERE MADE AFTER IT
05300                   ARE ELIMINATED FROM THE STACK. THIS IS SO A
05320                   PROGRAM THAT JUMPS OUT OF THE MIDDLE
05340                   OF A "FOR" LOOP AND THEN RESTARTS THE LOOP AGAIN
05360                   AND AGAIN WON'T USE UP 18 BYTES OF STACK
05380                   SPACE EVERY TIME. THE "NEXT" CODE ALSO
05400                   CALLS FNDFOR TO SEARCH FOR A "FOR" ENTRY WITH
05420                   THE LOOP VARIABLE IN
05440                   THE "NEXT". AT WHATEVER POINT A MATCH IS FOUND
05460                   THE STACK IS RESET. IF NO MATCH IS FOUND A
05480                   "NEXT WITHOUT FOR"  ERROR OCCURS. GOSUB EXECUTION
05500                   ALSO PUTS A 6 BYTE ENTRY ON STACK.
05520                   WHEN A RETURN IS EXECUTED FNDFOR IS
05540                   CALLED WITH A VARIABLE POINTER THAT CAN'T
05560                   BE MATCHED. WHEN "FNDFOR" HAS RUN
05580                   THROUGH ALL THE "FOR" ENTRIES ON THE STACK
05600                   IT RETURNS AND THE RETURN CODE MAKES
05620                   SURE THE ENTRY THAT WAS STOPPED
05640                   ON IS A GOSUB ENTRY. THIS ASSURES THAT
05660                   IF YOU GOSUB TO A SECTION OF CODE
05680                   IN WHICH A FOR LOOP IS ENTERED BUT NEVER
05700                   EXITED THE RETURN WILL STILL BE
05720                   ABLE TO FIND THE MOST RECENT
05740                   GOSUB ENTRY. THE "RETURN" CODE ELIMINATES THE
05760                   "GOSUB" ENTRY AND ALL "FOR" ENTRIES MADE AFTER
05780                   THE GOSUB ENTRY.

05820           NON-RUNTIME STUFF
05840                   THE CODE TO INPUT A LINE,CRUNCH IT,GIVE ERRORS,
05860                   FIND A SPECIFIC LINE IN THE PROGRAM,
05880                   PERFORM A "NEW","CLEAR", AND "LIST" ARE
05900                   ALL IN THIS AREA. GIVEN THE EXPLANATION OF
05920                   PROGRAM STORAGE GIVEN BELOW THESE ARE
05940                   ALL STRAIGHTFORWARD.

05980           NEWSTT
06000                   WHENEVER A STATEMENT FINISHES EXECUTION IT
06020                   DOES A "RET" WHICH TAKES
06040                   EXECUTION BACK TO NEWSTT. STATEMENTS THAT
06060                   CREATE OR LOOK AT SEMI-PERMANENT STACK ENTRIES
```

```
06080                   MUST GET RID OF THE RETURN ADDRESS OF NEWSTT AND
06100                   JMP TO NEWSTT WHEN DONE. NEWSTT ALWAYS
06120                   CHRGETS THE FIRST CHARACTER AFTER THE STATEMENT
06140                   NAME BEFORE DISPATCHING. WHEN RETURNING
06160                   BACK TO NEWSTT THE ONLY THING THAT
06180                   MUST BE SET UP IS THE TEXT POINTER IN
06200                   [H,L]. NEWSTT WILL CHECK TO MAKE SURE
06220                   [H,L] IS POINTING TO A STATEMENT TERMINATOR.
06240                   IF A STATEMENT SHOULDN'T BE PERFORMED UNLESS
06260                   IT IS PROPERLY FORMATTED (I.E. "NEW") IT CAN
06280                   SIMPLY DO A "RNZ" AFTER READING ALL OF
06300                   ITS ARGUMENTS. SINCE THE ZERO FLAG
06320                   BEING OFF INDICATES THERE IS NOT
06340                   A STATEMENT TERMINATOR NEWSTT WILL
06360                   DO THE JMP TO THE "SYNTAX ERROR"
06380                   ROUTINE. IF A STATEMENT SHOULD BE STARTED
06400                   OVER IT CAN DO LHLD TEMP,RET SINCE THE [H,L]
06420                   AT NEWSTT IS ALWAYS STORED IN TEMP. OF COURSE
06440                   CARE MUST BE TAKEN THAT NO ROUTINE
06460                   THAT SMASHES TEMP HAS BEEN CALLED.
06480                   THE "C CODE STORES TEMP IN OLDTXT AND CURLIN (THE
06500                   CURRENT LINE NUMBER) IN OLDLIN SINCE THE "C CHECK
06520                   IS MADE BEFORE THE STATEMENT POINTED TO IS
06540                   EXECUTED. "STOP" AND "END" STORE THE TEXT POINTER
06560                   IN [H,L] WHICH POINTS AT THEIR TERMINATING
06580                   CHARACTER IN OLDTXT.

06620           STATEMENT CODE
06640                   THE INDIVIDUAL STATEMENT CODE COMES
06660                   NEXT. THE APPROACH USED IN EXECUTING EACH
06680                   STATEMENT IS DOCUMENTED IN THE STATEMENT CODE
06700                   ITSELF.

06740           FRMEVL, THE FORMULA EVALUATOR
06760                   GIVEN AN [H,L] POINTING TO THE STARTING
06780                   CHARACTER OF A FORMULA FRMEVL
06800                   EVALUATES THE FORMULA AND LEAVES
06820                   THE VALUE IN THE FLOATING ACCUMULATOR (FAC).
06840                   [H,L] IS RETURNED POINTING TO THE FIRST CHARACTER
06860                   THAT COULD NOT BE INTERPRETED AS PART OF THE
06880                   FORMULA. THE ALGORITHM USES THE STACK
06900                   TO STORE TEMPORARY RESULTS:

06940                           0. PUT A DUMMY PRECEDENCE OF ZERO ON
06960                                   THE STACK.
06980                           1. READ LEXEME (CONSTANT,FUNCTION,
07000                                   VARIABLE,FORMULA IN PARENS)
07020                                   AND TAKE THE LAST PRECEDENCE VALUE
07040                                   OFF THE STACK.
07060                           2. SEE IF THE NEXT CHARACTER IS AN OPERATOR
07080                                   IF NOT,RETURN. THIS MAY CAUSE
07100                                   OPERATOR APPLICATION OR AN ACTUAL
07120                                   RETURN FROM FRMEVL.
```

```
                3. IF IT IS, SEE WHAT PRECEDENCE IT HAS
                        AND COMPARE IT TO THE PRECEDENCE
                        OF THE LAST OPERATOR ON THE STACK
                4. IF = OR LESS REMEMBER THE TEXT
                        POINTER AT THE START OF THIS OPERATOR
                        AND DO A RETURN TO CAUSE
                        APPLICATION OF THE LAST OPERATOR.
                        EVENTUALLY RETURN TO STEP 2
                        BY RETURNING TO RETAOP.
                5. IF GREATER PUT THE LAST PRECEDENCE
                        BACK ON, SAVE THE CURRENT
                        TEMPORARY RESULT, OPERATOR ADDRESS
                        AND PRECEDENCE AND RETURN TO STEP 1.

        RELATIONAL OPERATORS ARE ALL HANDLED THROUGH
        A COMMON ROUTINE. SPECIAL
        CARE IS TAKEN TO DETECT TYPE MISMATCHES SUCH AS 3+"F"

EVAL -- THE ROUTINE TO READ A LEXEME
        EVAL CHECKS FOR THE DIFFERENT TYPES OF
        ENTITIES IT IS SUPPOSED TO DETECT.
        LEADING PLUSES ARE IGNORED.
        DIGITS AND "." CAUSE FIN (FLOATING INPUT)
        TO BE CALLED. FUNCTION NAMES CAUSE THE
        FORMULA INSIDE THE PARENTHESES TO BE EVALUATED
        AND THE FUNCTION ROUTINE TO BE CALLED. VARIABLE
        NAMES CAUSE PTRGET TO BE CALLED TO GET A POINTER
        TO THE VALUE, AND THEN THE VALUE IS PUT INTO
        THE FAC. AN OPEN PARENTHESIS CAUSES FRMEVL
        TO BE CALLED (RECURSIVELY), AND THE ")" TO
        BE CHECKED FOR. UNARY OPERATORS (NOT AND
        NEGATION)  PUT THEIR PRECEDENCE ON THE STACK
        AND ENTER FORMULA EVALUATION AT STEP 1, SO
        THAT EVERYTHING UP TO AN OPERATOR GREATER THAN
        THEIR PRECEDENCE OR THE END OF THE FORMULA
        WILL BE EVALUATED. WHEN FRMEVL DOES A RETURN
        BECAUSE IT SEES AN OPERATOR OF HIGHER PRECEDENCE
        IT DOES NOT PASS THE TEXT POINTER IN [H,L], SO
        AFTER THE UNARY OPERATION HAS BEEN PERFORMED
        ON THE FAC THE TEXT POINTER MUST BE FETCHED FROM
        A TEMPORARY LOCATION THAT FRMEVL USES AND
        A RETURN BACK TO FRMEVL DONE.

DIMENSION AND VARIABLE SEARCHING
        SPACE IS ALLOCATED FOR VARIABLES AS THEY ARE
        ENCOUNTERED. THUS "DIM" STATEMENTS MUST BE
        EXECUTED TO HAVE EFFECT. 6 BYTES ARE ALLOCATED
        FOR EACH SIMPLE VARIABLE, WHETHER IT IS A STRING,
        NUMBER OR USER DEFINED FUNCTION. THE FIRST TWO
        BYTES GIVE THE NAME OF THE VARIABLE AND THE LAST FOUR
        GIVE ITS VALUE. [VARTAB] GIVES THE FIRST LOCATION
        WHERE A SIMPLE VARIABLE NAME IS FOUND AND [ARYTAB]
        GIVES THE LOCATION TO STOP SEARCHING FOR SIMPLE
```

```
        VARIABLES. A "FOR" ENTRY HAS A TEXT POINTER
        AND A POINTER TO A VARIABLE VALUE SO NEITHER
        THE PROGRAM OR THE SIMPLE VARIABLES CAN BE
        MOVED WHILE THERE ARE ACTIVE "FOR" ENTRIES ON THE STACK.
        USER DEFINED FUNCTION VALUES ALSO CONTAIN
        POINTERS INTO SIMPLE VARIABLE SPACE SO NO USER-DEFINED
        FUNCTION VALUES CAN BE RETAINED IF SIMPLE VARIABLES
        ARE MOVED. ADDING A SIMPLE VARIABLE
        ADDING SIX TO ARYTAB AND STREND, BLOCK TRANSFERING
        THE ARRAY VARIABLES UP BY SIX AND MAKING SURE THE
        NEW [STREND] IS NOT TO CLOSE TO THE STACK.
        THIS MOVEMENT OF ARRAY VARIABLES MEANS
        THAT NO POINTER TO AN ARRAY WILL STAY VALID WHEN
        NEW SIMPLE VARIABLES CAN BE ENCOUNTERED. THIS IS
        WHY ARRAY VARIABLES ARE NOT ALLOWED "FOR"
        LOOP VARIABLES. SETING UP ANEW ARRAY VARIABLE
        MERELY INVOLVES BUILDING THE DESCRIPTOR,
        UPDATING STREND, AND MAKING SURE THERE IS
        STILL ENOUGH ROOM BETWEEN STREND AND THE
        STACK. WITHOUT MULTIPLE DIMENSIONS THE FORMAT
        OF AN ARRAY VARIABLE IS SIMPLY:
                SECOND CHARACTER
                FIRST CHARACTER
                NUMBER OF BYTES USED BY VALUES
                VALUES
        THE FORMAT WHEN MULTIPLY DIMENSIONED VARIABLES
        ARE ALLOWED IS DESCRIBED IN THE "MULDIM" CODE.
        PTRGET, THE ROUTINE WHICH RETURNS A POINTER
        TO A VARIABLE VALUE, HAS TWO IMPORTANT FLAGS. ONE IS
        "DIMFLG" WHICH INDICATED WHETHER "DIM" CALLED PTRGET
        OR NOT. IF SO, NO PRIOR ENTRY FOR THE VARIABLE IN
        QUESTION SHOULD BE FOUND, AND THE INDEX INDICATES
        HOW MUCH SPACE TO SET ASIDE. SIMPLE VARIABLES CAN
        BE "DIMENSIONED", BUT THE ONLY EFFECT WILL BE TO
        SET ASIDE SPACE FOR THE VARIABLE IF IT HASN'T BEEN
        ENCOUNTERED YET. THE OTHER IMPORTANT FLAG IS SUBFLG
        WHICH INDICATES WHETHER A SUBSCRIPTED VARIABLE SHOULD BE
        ALLOWED IN THE CURRENT CONTEXT. IF SUBFLG IS NON-ZERO
        THE OPEN PARENTHESIS FOR A SUBSCRIPTED VARIABLE
        WILL NOT BE SCANNED BY PTRGET, AND PTRGET WILL RETURN
        WITH A TEXT POINTER POINTING TO THE "(", IF
        THERE WAS ONE.
STRINGS
        IN THE VARIABLE TABLE STRINGS ARE STORED JUST LIKE
        NUMERIC VARIABLES. SIMPLE STRINGS HAVE FOUR VALUE
        BYTES WHICH ARE INITIALIZED TO ALL ZEROS (WHICH
        REPRESENTS THE NULL STRING). THE ONLY DIFFERENCE
        IN HANDLING IS THAT WHEN PTRGET SEES A "$" AFTER THE
        NAME OF A VARIABLE, PTRGET SETS VALTYP TO ONE AND TURNS
        ON THE MSB (MOST-SIGNIFIGANT-BIT) OF THE VALUE OF
        THE FIRST CHARACTER OF THE VARIABLE NAME.
        HAVING THIS BIT ON IN THE NAME OF THE VARIABLE ENSURES
        THAT THE SEARCH ROUTINE WILL NOT MATCH
'A' WITH 'A$' OR 'A$' WITH 'A'. THE MEANING OF
THE FOUR VALUE BYTES ARE:
        LOW
                LENGTH OF THE STRING
                UNUSED
                LOW 8 BITS
                HIGH 8 BITS  OF THE ADDRESS
                        OF THE CHARACTERS IN THE
                        STRING IF LENGTH.NE.0.
                        MEANINGLESS OTHERWISE.
        HIGH
THE VALUE OF A STRING VARIABLE (THESE 4 BYTES)
IS CALLED THE STRING DESCRIPTOR TO DISTINGUISH
IT FROM THE ACTUAL STRING DATA. WHENEVER A
STRING CONSTANT IS ENCOUNTERED IN A FORMULA OR AS
PART OF AN INPUT STRING, OR AS PART OF DATA, STRLIT
IS CALLED, CAUSING A DESCRIPTOR TO BE BUILT FOR
THE STRING. IF THE STRING CONSTANT IS IN BUF (WHICH
IT WILL BE IF THE STRING IS BEING "INPUT", OR THE
STRING IS PART OF SOME FORMULA IN A DIRECT STATEMENT)
THE VALUE IS COPIED INTO STRING SPACE SINCE BUF
IS ALWAYS CHANGING. "STRCPY" IS USED TO COPY
STRINGS.

STRING FUNCTIONS AND THE ONE STRING OPERATOR "+"
ALWAYS RETURN THEIR VALUES IN STRING SPACE.
ASSIGNING A STRING A CONSTANT VALUE IN A PROGRAM
THROUGH A "READ" OR ASSIGNMENT STATEMENT
WILL NOT USE ANY STRING SPACE SINCE
THE STRING DESCRIPTOR  WILL POINT INTO THE
PROGRAM ITSELF. IN GENERAL, COPYING IS DONE
WHEN A STRING VALUE IS IN BUF, OR IT IS IN STRING
SPACE AND THERE IS AN ACTIVE POINTER TO IT.
THUS F$=G$ WILL CAUSE COPYING IF G$ HAS ITS
STRING DATA IN STRING SPACE. F$=CHR$(7)
WILL USE ONE BYTE OF STRING SPACE TO STORE THE
NEW ONE CHARACTER STRING CREATED BY "CHR$", BUT
THE ASSIGNMENT ITSELF WILL CAUSE NO COPYING SINCE
THE ONLY POINTER AT THE NEW STRING IS A
TEMPORARY DESCRIPTOR CREATED BY FRMEVL WHICH WILL
GO AWAY AS SOON AS THE ASSIGNMENT IS DONE.
IT IS THE NATURE OF GARBAGE COLLECTION THAT
DISALLOWS HAVING TWO STRING DESCRIPTORS POINT TO THE SAME
AREA IN STRING SPACE. STRING FUNCTIONS AND OPERATORS
MUST PROCEED AS FOLLOWS:
        1) FIGURE OUT THE LENGTH OF THEIR RESULT
        2) CALL GETSPA TO FIND SPACE FOR THEIR
        RESULT. THE ARGUMENTS TO THE FUNCTION
        OR OPERATOR MAY CHANGE SINCE GARBAGE COLLECTION
        MAY BE INVOKED. THE ONLY THING THAT CAN
        BE SAVED DURING THE CALL TO GETSPA IS A POINTER
        TO THE DESCRIPTORS OF THE ARGUMENTS.
        3) CONSTRUCT THE RESULT DESCRIPTOR IN DSCTMP.
```

```
        GETSPA RETURNS THE LOCATION OF THE AVAILABLE
        SPACE.
        4) CREATE THE NEW VALUE BY COPYING PARTS
        OF THE ARGUMENTS OR WHATEVER.
        5) FREE UP THE ARGUMENTS BY CALLING FRETMP.
        6) JUMP TO PUTNEW TO GET THE DESCRIPTOR IN
        DSCTMP TRANSFERRED INTO A NEW STRING TEMPORARY.

THE REASON FOR STRING TEMPORARIES IS THAT GARBAGE
COLLECTION HAS TO KNOW ABOUT ALL ACTIVE STRING DESCRIPTORS
SO IT KNOWS WHAT IS AND ISN'T IN USE. STRING TEMPORARIES ARE
USED TO STORE THE DESCRIPTORS OF STRING EXPRESSIONS.

INSTEAD OF HAVING AN ACTUAL VALUE STORED IN THE
FAC, AND HAVING THE VALUE OF A TEMPORARY RESULT
BEING SAVED ON THE STACK, AS HAPPENS WITH NUMERIC
VARIABLES, STRINGS HAVE THE POINTER TO A STRING DESCRIPTOR
STORED IN THE FAC, AND IT IS THIS POINTER
THAT GETS SAVED ON THE STACK BY FORMULA EVALUATION.
STRING FUNCTIONS CANNOT FREE THEIR ARGUMENTS UP RIGHT
AWAY SINCE GETSPA MAY FORCE
GARBAGE COLLECTION AND THE ARGUMENT STRINGS
MAY BE OVER-WRITTEN SINCE GARBAGE COLLECTION
WILL NOT BE ABLE TO FIND AN ACTIVE POINTER TO
THEM. FUNCTION AND OPERATOR RESULTS ARE BUILT IN
DSCTMP SINCE STRING TEMPORARIES ARE ALLOCATED
(PUTNEW) AND DEALLOATED (FRETMP) IN A FIFO ORDERING
(I.E. A STACK) SO THE NEW TEMPORARY CANNOT
BE SET UP UNTIL THE OLD ONE(S) ARE FREED. TRYING
TO BUILD A RESULT IN A TEMPORARY AFTER
FREEING UP THE ARGUMENT TEMPORARIES COULD RESULT
IN ONE OF THE ARGUMENT TEMPORARIES BEING OVERWRITTEN
TOO SOON BY THE NEW RESULT.

STRING SPACE IS ALLOCATED AT THE VERY TOP
OF MEMORY. MEMSIZ POINTS BEYOND THE LAST LOCATION OF
STRING SPACE. STRING ARE STORED IN HIGH LOCATIONS
FIRST. WHENEVER STRING SPACE IS ALLOCATED (GETSPA)
FRETOP, WHICH IS INITIALIZED TO [MEMSIZ], IS UPDATED
TO GIVE THE HIGHEST LOCATION IN STRING SPACE
THAT IS NOT IN USE. THE RESULT IS THAT
FRETOP GETS SMALLER AND SMALLER, UNTIL SOME
ALLOCATION WOULD MAKE [FRETOP] LESS THAN OR EQUAL TO
[STKTOP]. THIS MEANS STRING SPACE HAS RUN INTO THE
STACK AND THAT GARBAGE COLLECTION MUST BE CALLED.

GARBAGE COLLECTION:
        0. MINPTR=[STKTOP] [FRETOP]=[MEMSIZ]
        1. REMMIN=0
        2. FOR EACH STRING DESCRIPTOR
        (TEMPORARIES, SIMPLE STRINGS, STRING ARRAYS)
        IF THE STRING IS NOT NULL AND ITS POINTER IS
        .GT.MINPTR AND .LT.FRETOP,
11380                                   MINPTR=THIS STRING DESCRIPTORS POINTER
11400                                   REMMIN=POINTER AT THIS STRING DESCRIPTOR
11420                                   END
11440                                   3, IF REMMIN,NE,0 (WE FOUND AN UNCOLLECTED STRING)
11460                                   BLOCK TRANSFER THE STRING DATA POINTED
11480                                   TO IN THE STRING DESCRIPTOR POINTED TO BY REMMIN
11500                                   SO THAT THE LAST BYTE OF STRING DATA IS AT
11520                                   [FRETOP], UPDATE FRETOP SO THAT IT
11540                                   POINTS TO THE LOCATION JUST BELOW THE ONE
11560                                   THE STRING DATA WAS MOVED INTO, UPDATE
11580                                   THE POINTER IN THE DESCRIPTOR SO IT POINTS
11600                                   TO THE NEW LOCATION OF THE STRING DATA,
11620                                   GO TO STEP 1,

11660                           AFTER CALLING GARBAGE COLLECTION GETSPA AGAIN CHECKS
11680                           TO SEE IF [A] CHARACTERS ARE AVAILABLE BETWEEN
11700                           [STKTOP] AND [FRETOP] , IF NOT AN "OUT OF STRING"
11720                           ERROR IS INVOKED,

11760                   MATH PACKAGE
11780                           THE MATH PACKAGE CONTAINS FLOATING INPUT (FIN),
11800                           FLOATING OUTPUT (FOUT) FLOATING COMPARE (FCOMP)
11820                           ... AND ALL THE NUMERIC OPERATORS AND FUNCTIONS,
11840                           THE FORMATS,CONVENTIONS AND ENTRY POINTS ARE ALL
11860                           DESCRIBED IN THE MATH PACKAGE ITSELF,

11900                   INIT -- THE INITIALIZATION ROUTINE
11920                           INITIALIZATION FIRST LOOKS AT THE SWITCH REGISTER
11940                           TO SEE WHAT TYPE OF I/O SHOULD BE DONE,
11960                           ANY NON-STANDARD I/O CAUSES LOCATIONS IN BASIC
11980                           TO BE CHANGED, THEN THE AMOUNT OF MEMORY,
12000                           TERMINAL WIDTH,AND WHICH FUNCTIONS TO BE RETAINED
12020                           ARE ASCERTAINED FROM THE USER, A ZERO IS PUT DOWN
12040                           AT THE FIRST LOCATION NOT USED BY THE MATH-PACKAGE
12060                           AND TXTTAB IS SET UP TO POINT AT THE NEXT LOCATION,
12080                           THIS DETERMINES WHERE PROGRAM STORAGE WILL START, THE
12100                           HIGHEST MEMORY LOCATION MINUS THE AMOUNT OF DEFAULTED
12120                           STRING SPACE (50) GIVES THE FIRST LOCATION USED BY THE
12140                           STACK, SPECIAL CHECKS ARE MADE TO MAKE SURE
12160                           ALL QUESTIONS IN INIT ARE ANSWERED REASONABLY, SINCE
12180                           ONCE INIT FINISHES THE LOCATIONS IT USES ARE
12200                           USED FOR PROGRAM STORAGE, THE LAST THING INIT DOES IS
12220                           CHANGE LOCATION ZERO TO BE A JUMP TO READY INSTEAD
12240                           OF INIT, ONCE THIS IS DONE THERE IS NO WAY TO RESTART
12260                           INIT,

12300                   STORAGE
12320                                           A ZERO,
12340                           [TXTTAB]        POINTER TO NEXT LINE'S POINTER
12360                                           LINE # OF THIS LINE (2 BYTES)
12380                                           CHARACTERS ON THIS LINE
12400                                           ZERO
12420                                           POINTER AT NEXT LINE'S POINTER
```

```
12440                                                       (POINTED TO BY THE ABOVE POINTER)
12460                                           ... REPEATS ...
12480                           LAST LINE:      POINTER AT ZERO POINTER
12500                                           LINE # OF THIS LINE
12520                                           CHARACTERS ON THIS LINE
12540                                           ZERO
12560                                           DOUBLE ZERO (POINTED TO BY THE ABOVE POINTER)
12580                           [VARTAB]        SIMPLE VARIABLES, 6 BYTES PER VALUE,
12600                                           2 BYTES GIVE THE NAME, 4 BYTES THE VALUE
12620                                           ... REPEATS ...
12640                           [ARYTAB]        ARRAY VARIABLES, 2 BYTES NAME, 2 BYTE
12660                                           LENGTH, VALUE (EXTRA IF MULDIM ON)
12680                                           ... REPEATS  ...
12700                           [STREND]        FREE SPACE
12720                                           ... REPEATS ...
12740                                           MOST RECENT STACK ENTRY
12760                                           ... REPEATS ...
12780                           [STKTOP]        FIRST STACK ENTRY
12800                                           FREE STRING SPACE
12820                                           ... REPEATS ...
12840                           [FRETOP]        STRING SPACE IN USE
12860                                           ... REPEATS ...
12880                           [MEMSIZ]        HIGHEST MACHINE LOCATION
12900                                           UNUSED EXCEPT BY THE VAL FUNCTION,
12920   HIGH LOCATIONS

12960   *
12980   PAGE
```