BYTE NYBBLES

The Design of an M6800 LISP Interpreter

S Tucker Taft
Harvard University Science Center
1 Oxford St
Cambridge MA 02138

Anyone exposed to small computer systems has used a language interpreter of some sort, and certainly may have thought about implementing their own interpreter. Unhappily, implementing an interpreter for a complete version of most computer languages is a difficult and time-consuming job, unsuitable for a part-time personal computer enthusiast. The language LISP provides a unique opportunity in this respect. The foundation for a very complete interpreter can be programmed by a single person in several months of part-time effort. As a bonus, the resulting interpreter provides the user with a high level language in which to express algorithms.

The Language

From the user's point of view, the primary data structure in LISP is the *list*. Every element of a list is either an *atom* or another list. An atom is a primitive named object, the name being an arbitrary string of characters:

ABC is an atom.
135 is an atom.
(ABC 135) is a list of two elements, both atoms.
((ABC 135) XYZ) is a list of two elements, the first of which is a list, the second is an atom.
(() ()) is a list of two elements, both being lists of zero elements. A list of zero elements, the *null* list, is identified with the atom NIL.

The feature of the language LISP which makes it at the same time a uniquely interesting language, and relatively easy to implement, is that all program elements are represented using these same kinds of objects: atoms and list. Constants, variables, expressions, conditionals, even function definitions are all represented using only atoms and lists.

A value is associated with each atom, allowing atoms to represent program variables and constants. A symbolic atom, like XYZ, would represent a variable. A numeric atom, like 237, would represent a constant.

Operations on variables and constants, like addition, or a function call, are represented by list expressions:

(ADD 2 5) would represent the expression 2 + 5.
(SIN (MUL 2 Y)) would represent the expression sin(2y).

Conditionals, loops, and function definitions are also represented by list expressions, as illustrated by this recursive function implementing Euclid's greatest common divisor algorithm:

```
(DEF GCD (LAMBDA (X Y)
   (COND
      ((GREATER X Y) (GCD (SUB X Y) Y))
      ((GREATER Y X) (GCD X (SUB Y X)))
      (T X)
   )
))
```

This would be equivalent to the Pascal program:

```
function gcd(x,y:integer):integer
   begin
      if x>y then gcd := gcd(x-y, y)
   else
      if y>x then gcd := gcd(x, y-x)
   else
      gcd := x
   end.
```

An important difference to note in the above comparison is that no explicit assignment to a function return value is made in LISP, whereas in Pascal one must explicitly say *gcd* := ... to specify the return value. In Pascal, and most other *procedural* languages, a distinction is made between program statements and expressions. In such languages some program statement must be executed to specify the return value, usually either a

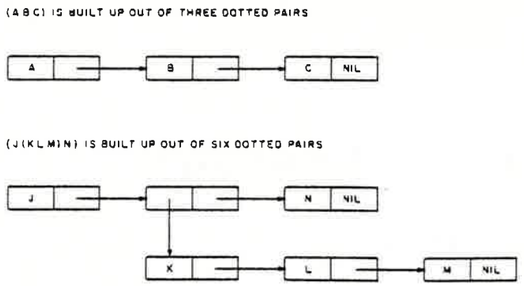

Figure 1: In most LISP systems, lists are built up out of dotted pairs which are two address cells. The left cell points to the first element of a list, and the right cell points to the rest of the list. The letters in the figure stand for atoms. NIL is a special atom used to signify the end of a chain of dotted pairs.

return statement or an assignment to the function name. In LISP, and other *applicative* languages, no such distinction is made. A function is simply a single expression, whose value is the return value of the subprogram.

This is made possible by built-in functions like COND used above. COND takes a list of two element lists as argument. It goes down the list of pairs, evaluating the first element of each pair. If the result is true (the atom T), the result of the entire COND is the value of the second element of the pair. If the value of the first element of the pair is false (the atom NIL), COND proceeds to the next pair. If COND reaches the end of the list, the result of the entire COND is simply NIL. In the above example this would never happen because the first element of the last pair is the atom T (whose value is always guaranteed to be itself, the atom T). This is the normal technique in LISP for using the COND function.

The expression:

(DEF GCD (LAMBDA (X Y)...

defines the atom GCD to be a function (or *lambda expression*) taking two arguments, to be called X and Y in the body of the definition. Notice that no explicit specification of the type of X or Y is provided. In LISP any arbitrary value, atom, or list may be the value associated with an atom. In this sense LISP is a typeless language. In fact the type of a *value* (ie: whether it is an atom or a list) is always determinable at execution time. Functions must check the types of the values of atoms if only certain types are legal arguments. In the above example the calls on GREATER and SUB would fail if the values associated with X and Y were not numeric atoms.

CARs and CDRs

Thus far we have only shown how to re-express algorithms written in a more conventional language, in the language LISP. The real power of LISP comes from its ability to directly manipulate lists, a data type not normally accessible in other languages. Three primitives, CAR, CDR (pronounced could-er), and CONS are provided for list manipulation. The function CAR takes a list as argument, and returns the first element of the list, which may either be an atom or another list. The function CDR takes a list as argument, and returns the tail of the list, that is, all but the first element of the orginal list, as a new list. The function CONS takes two arguments, a new first element, and the tail of a list, and reconstructs a list, now one element longer. For example:

Assume the atom X is associated with the value:
(A B C)
Assume the atom Y is associated with the value:
(THE CAT IN THE HAT)
(CAR X) would be the atom A.
(CDR Y) would be the list (CAT IN THE HAT).
(CONS (CAR X) (CDR Y)) would be the list:
(A CAT IN THE HAT)
(CAR (CDR X)) would be the atom B.

In general the CAR of the CDR of a list is its second element, and a function called CADR is frequently defined as a kind of shorthand for CAR of the CDR.

You might wonder what would result if you gave two atoms as arguments to CONS, rather than an atom and a list. In most LISP systems this is in fact legal. The result reveals the underlying representation used for lists in LISP. In virtually all LISP systems, lists are built up out of *dotted pairs*, two-address cells, the left cell pointing to the first element of a list, and the right cell pointing to the rest of the list. This can be diagrammed schematically as in figure 1.

Because dotted pairs are used this way to build up lists, it is natural to call the left cell of a dotted pair the CAR and the right cell the CDR. (In fact the genealogy of the words CAR and CDR runs the other way. Dotted pairs were used in the initial implementation of LISP, and CAR and CDR referred to the address field and the decrement field of a word on the IBM 704.) Now you can perhaps guess that when you pass two atoms as arguments to CONS, you simply get a dotted pair with an atom in both the CAR and CDR. For example:

A	B

would be printed as:

(A . B)

The notation (A . B) is used whenever the CDR of the last dotted pair forming a linked list is a non-NIL atom. In general (D E F . NIL) would be equivalent to (D E F), whereas (D E F . G) could not be expressed without the *dot* notation.

Given the three primitives CAR, CDR, and CONS, and understanding the underlying representation of lists using dotted pairs, it is possible to write powerful list-manipulating programs in LISP. For example, suppose it is desirable to edit a large data structure, and change all occurrences of the symbol APPLE to ORANGE. In LISP we could easily write a routine called REPLACE which, given the data structure (ie: list structure), the original symbol (the atom APPLE), and the replacement symbol (the atom ORANGE), would go through the structure and do the replacement, using itself recursively to do the replacement in all sublists of the list structure:

```
(DEF REPLACE (LAMBDA (STRUC OLD NEW)
   (COND
      ((EQ STRUC OLD) NEW)
      ((ATOM STRUC) STRUC)
      (T (CONS
            (REPLACE (CAR STRUC) OLD NEW)
            (REPLACE (CDR STRUC) OLD NEW)
      ))
   )
))
```

Notice how the first two lines of the COND allow for the possibility that the input data structure is simply an atom (which may or may not be equal to the atom to be replaced). In addition, notice that the entire body of this function definition is a single COND, just as it was in the GCD example given above. This is frequently true in LISP programs. Finally, notice how the function simply *passes the buck* to recursive calls on itself if the STRUC argument is not an atom, CONSing together the results of the two inner calls. The reader is encouraged to go through an example of the execution of this function when the argument OLD is the atom APPLE, the argument NEW is the atom ORANGE, and the argument STRUC is the list structure:

(AN (APPLE A DAY) KEEPS (THE (APPLE MAN) BUSY))
The result should be:
(AN (ORANGE A DAY) KEEPS (THE (ORANGE MAN) BUSY))
If STRUC were:
(PEAR BANANA . APPLE)
the result should be:
(PEAR BANANA . ORANGE)

Other kinds of list-manipulating programs which are relatively easy to write in LISP, but very difficult in more conventional languages, include formula manipulation programs which might take in the list representation for a function (eg: (SIN (MUL 2 X))), and return the list representation for its derivative according to the rules of the calculus (eg: (MUL 2 (COS (MUL 2 X)))).

The author's system is being used for the development of a compiler/interpreter system which generates the list representation for a program written in a programming language, and then either interprets it directly, or generates the list of machine language statements to implement the program on a particular microcomputer. LISP makes such an undertaking quite straightforward (although not trivial, unfortunately!).

LISP Interpreter

Because programs are data objects (list structures) in LISP, the same routines used to read and print data objects may be used to read and print programs. Furthermore user functions, like a general list editor, can be used also to edit programs. This uniformity vastly simplifies the task of writing an interpreter for LISP. Only three basic modules need be produced: READ, EVAL, and PRINT . READ accepts a LISP list expression from the terminal, in full parenthesized notation, and builds the internal representation of the list, sometimes called a *form*. EVAL takes a form as its single argument, and evaluates the form according to the LISP convention that the first element of such a list specifies the function, with the rest of the list as arguments.

The result of EVAL is another form. (The term *form* is sometimes reserved for LISP expressions which are legal input to EVAL. The term *S-expression* covers all types of lists, whether or not the first element is a legal function name. Within this paper, form will be used to refer to the internal representation of any type of LISP expression.)

PRINT takes a form as its argument, and types it on the terminal in fully parenthesized form. The top level loop of the LISP interpreter simply prompts the user for input (-> is the LISP prompt), READs in the users input, EVALs the resulting form, and PRINTs the result of EVAL. In a conventional high level language syntax, this would be:

```
while true do begin
   patom("->");
   form := read( );
   form := eval(form);
   print(form)
end.
```

or in M6800 assembly language:

```
BIGLUP LDX   PRMPAT    get prompt atom
       JSR   PATOM     print the atom
       JSR   READ      read the form typed in
*  result now in M6800 x-register
       JSR   EVAL      eval the form
*  result of EVAL back in x-register
       JSR   PRINT     print the form
       BRA   BIGLUP      and loop around
```

PATOM is a subroutine, also called by PRINT, when a form is known to be an atom. In an assembly language implementation, it would be very convenient to pass forms in the M6800 index (X) register. This register is 16 bits long, so it requires that forms be only 16 bits. Some representation must be chosen for all LISP objects so that a single 16 bit number may uniquely specify any arbitrary object. Dotted pairs are used to represent lists. Dotted pairs hold two forms, a CAR and a CDR, so they must be 32 bit objects. A natural choice is to allocate 4 consecutive M6800 memory bytes for dotted pairs, and specify dotted pairs by the address of their first byte. This means that any two different dotted pairs will be easily differentiated by the forms that specify them.

This still leaves the problem of deciding on an internal representation for atoms, including symbolic atoms, numeric atoms, and NIL. In the author's LISP system only two items of information are needed for each symbolic atom, the string of characters which are the print name of the atom, and the value currently associated with the atom (which is an arbitrary form). Again a 4 byte representation is chosen, with the first two bytes used as a memory address pointing to the first character of the print name, and the third and fourth bytes used to hold the value (a form) of the atom. Now the address of

this 4 byte object can specify the atom uniquely from all other atoms and from all other dotted pairs.

Unfortunately this does not provide a simple way of distinguishing atoms from dotted pairs, when just given the form. Several solutions to this problem are possible. One is to restrict dotted pairs to a certain part of memory, then the address would determine whether the form specified an atom or a dotted pair. A second method is to add an additional byte to both dotted pairs and atoms which simply contains a type specifier, say 1 for dotted pairs and 2 for atoms. This method makes future expansion of types simple, but is somewhat wasteful in terms of space. The third method, the one chosen for the author's system, is to align all dotted pairs and atoms on 4 byte boundaries, that is, with addresses which are a multiple of four. This means that the low order two bits of the address are expected to always be zero, and hence may be used to encode type information. In the author's system, dotted pairs are specified by forms with both bits zero, and symbolic atoms by *01* in the lower two bits. One of the bits is still unused, but will become very handy when *garbage collection* methods are discussed below.

With numeric atoms, their name determines their value, and hence only their name (or their value) need be specified by a form. On the author's M6800 system only hexadecimal memory addresses 0000 thru 7FFF were accessible for storage of dotted pairs and atoms, meaning that the high order bit of forms specifying either of these was always zero. A representation for numeric atoms was chosen to be a form with the high order bit set, 14 bits of numeric value, and one bit left for *garbage collection*.

A special representation for the NIL atom is used both because the value of NIL is, like numeric atoms, required always to be the atom itself, and because it is used universally to represent the end of a list. The form chosen to specify NIL is simply the value zero. In fact any form with the high order byte zero is treated like NIL to simplify the test for NIL in certain cases. This means that the 256 byte page starting at zero is not usable for storing atoms or dotted pairs, but this restriction causes no problem at all, since both are allocated starting at the highest address available, and the allocator runs into program long before it reaches page zero.

When writing a LISP interpreter, the implementor must decide relatively early on how forms will specify all types of LISP objects. Unfortunately, it may not be until well into the implementation that the implementor discovers that certain choices were inefficient or inconvenient.

One important requirement affecting this decision not yet mentioned is the need to implement the LISP EQ function. This function takes two arbitrary forms, and returns the atom T or the atom NIL depending on whether the forms specify the same dotted pair, or whether the forms specify the same atom. Whenever an atom is input by READ, it must return the form specifying that atom to the caller. Whenever the same symbolic atom name is typed, READ must return the same form, ie: a pointer to the same 4 byte cell. This is accomplished by retaining a linked list of all defined symbolic atoms (called the OBLIST).

Before allocating a new 4 byte cell for an atom, READ scans the OBLIST for an atom of the given print name. If found, READ returns a form specifying that pre-exisiting atom. (Otherwise it must copy the name into some area used for storing names, allocate a 4 byte cell, initialize the left cell to point to the name, and the right cell to NIL, and return a form specifying the new atom.) This method guarantees that two forms specify the same symbolic atom if and only if they have the same address.

In some implementations of numeric atoms, this same rule cannot be guaranteed. In such systems, numeric atoms are simply allocated an appropriately large cell to store their numeric value (and hence allowing numeric atoms greater than 14 bits), a new cell being allocated every time a new number is generated (which happens at every ADD, MUL, etc). In these systems it would be impractical to scan a list like the OBLIST every time any arithmetic calculation is done, and so the LISP function EQ may not rely on the rule that unequal forms indicate unequal atoms. In such systems, EQ must look at the contents of the cell specified by a numeric atom form, and make the comparison that way. In systems like the author's, EQ simply compares the forms themselves, no matter what type of atom the form may specify.

The choices made in representing the various types of LISP objects can be summarized in the high level language (Pascal-like) data structure specification in listing 1.

```
type lisptype =
        (dtprtype, symatmtype, numatmtype, nilatmtype);
    dtpr =
        record
            car: form;
            cdr: form
        end;
    symatm =
        record
            name: ↑array [0..n] of char;
            value: form
        end;
    form =
        packed record
            gcbit: boolean;
            case objtype: lisptype of
                dtprtype:       (dtprform:      ↑dtpr);
                symatmtype:     (symatmform:    ↑symatm);
                numatmtype:     (numatmform:    -5000..4999);
                nilatmtype:     ( )
        end.
```

Listing 1: A Pascal data structure specification that could be used to represent various types of LISP objects.

READ Function

READ is the basic input routine for the LISP interpreter. READ accepts a fully parenthesized expression from the terminal, and builds up the internal representation, allocating new dotted pairs and atoms as necessary. If the expression is a list, READ returns a form specifying the first dotted pair of the constructed list. If the expression typed in is simply an atom, READ returns a form specifying the atom.

The logic of the READ routine is straightforward because the syntax of LISP expressions is so simple. READ calls a function RATOM to return the next input atom. RATOM actually does the work of allocating new 4 byte cells for symbolic atoms (when necessary) as explained above. RATOM returns a form specifying the

Listing 3: M6800 implementation of linked-list stack manipulation routines.

```
* PUSHX saves the value of the X-reg on a linked-list stack.
PUSHX   BSR   GETCEL allocate a new dotted pair and fill in the CAR
              LDAA STKPTR fill in the CDR from STKPTR
              STAA CDR,X   (CAR and CDR are symbols defined as
              LDAA STKPTR+1  0 and 2 respectively)
              STAA CDR+1,X
              STX  STKPTR update the STKPTR
              LDX  CAR,X  restore the X-reg
              RTS           and return.

* allocate a new dotted pair, save X-reg in the CAR,
*  and set the CDR to NIL
GETCEL  STX   XTMP   save the X-reg temporarily
        LDX   FREPTR pick up a free dotted pair off the FREE list.
        BEQ   NOFREE no more left, so it goes...
        LDAA  CDR,X  update the FREE list pointer
        STAA  FREPTR
        LDAA  CDR+1,X
        STAA  FREPTR+1
        CLR   CDR,X  set the CDR to NIL
        CLR   CDR+1,X
        LDAA  XTMP   fill in the CAR from the X-reg
        STAA  CAR,X
        LDAA  XTMP+1
        STAA  CAR+1,X
        RTS           and return.

* restore the X-reg from the linked-list stack
POPX    LDX   STKPTR point to the top of the stack
        LDAA  CDR,X  update the STKPTR
        STAA  STKPTR
        LDAA  CDR+1,X
        STAA  STKPTR+1
        ORA   FREEL  and free up the dotted pair used.

* free up a dotted pair, and load X-reg from the CAR
FREEL   LDAA  FREPTR link the cell onto the FREE list
        STAA  CDR,X
        LDAA  FREPTR+1
        STAA  CDR+1,X
        LDX   CAR,X   load X-reg from CAR
        RTS           and return.

* return value at top of stack in X-reg
* (but leave it on stack)
TOPX    LDX   STKPTR return CAR of cell on top of stack.
        LDX   CAR,X
        RTS
```

Listing 4: LSTINI, LSTADD, and LSTEND build up a linked list of dotted pairs using two pointers on a stack, one to the first dotted pair, one to the dotted pair at the current end of the linked list.

Listing 2: LISP (a) and M6800 assembly (b) code for the READ function.

(a)

```
(DEF READ (LAMBDA ()
    (READR (RATOM))
))

(DEF READR (LAMBDA (A)
    (COND
        ((EQ A LPAREN) (READL (READ)))
        (T A)
    )
))

(DEF READL (LAMBDA (F)
    (COND
        ((EQ F RPAREN) NIL)
        (T (CONS F (READL (READ))))
    )
))
```

(b)

```
READ    JSR RATOM  pick up the next input atom
        CPX LPARAT is it equal to the "(" atom?
        BNE RDRET  no, simply return the atom
        JSR LSTINI yes, initialize a linked list
RDLUP   BSR READ   call READ recursively to pick up next form
        CPX RPARAT is it equal to the ")" atom?
        BEQ RDLDUN yes, finalize the list and return
        JSR LSTADD no, allocate a new dotted pair,
*                    fill in the CAR, and add it to the list,
        BRA RDLUP    and loop around.
RDLDUN  JSR LSTEND finalize the list, get a pointer to
*                  the first dotted pair of the list
RDRET   RTS        and return.


* set up the stack for LSTADD.
* first stack a NIL, then a pointer to the NIL.
LSTINI  LDX  ZERO   stack a NIL form
        JSR  PUSHX
        LDX  STKPTR stack a pointer to the NIL
        DEX           as though it were the CDR of a cell.
        DEX
        JMP  PUSHX  push it and return.

* add form in X-reg to list pointed to by value on top of stack
*  clobbers X-reg, A-reg, and B-reg.
LSTADD  JSR  GETCEL get a dotted pair and fill in the CAR
        STX  CELPTR save address of new cell
        JSR  TOPX   retrieve pointer from top of stack
        LDAA CELPTR link new cell onto list
        STAA CDR,X
        LDAB CELPTR+1
        STAB CDR+1,X
        LDX  STKPTR update list pointer
        STAA CAR,X  (which is on top of stack)
        STAB CAR+1,X
        RTS           and return.

* pop off list end pointer
*  return pointer to first dotted pair of list in X-reg
LSTEND  JSR  POPX   pop off and ignore list end pointer
        JMP  POPX   pop off and return list begin pointer.
```

```
                STX   XTMP     save X-reg temporarily
                LDX   0,X      point at other name
                CMPA  0,X      first char equal?
                BNE   CMPRET   nope, return with Z-bit clear
                TSTA           all done now?
                BEQ   CMPRET   yep, return with Z-bit set
                CMPB  1,X      second char equal?
                BNE   CMPRET   nope, return with Z-bit clear
                TSTB           all done now?
                BEQ   CMPRET   yep, return with Z-bit set
                INX            match so far, advance pointers
                INX
                STX   XTMP2
                LDX   XTMP
                INX
                INX
                BRA   CMPLP    and loop around.
CMPRET          RTS            Z-bit now set properly.

* we have an old atom, reset spcptr and return
OLDATM  LDX   LSTPTR   get form specifying pre-existing atom
        LDX   CAR,X
        CPX   NILATM   is this the atom with the print name "NIL"
        BNE   RETNUM   yep, return a zero form instead
        LDX   ZERO     save form temporarily
RETNUM  STX   FORM     reset SPCPTR
        STX   SPCPTR
RATRET  LDX   FORM     and return with FORM in X-reg
        RTS

* we have a new atom, allocate a new cell and return new form
NEWATM  LDX   HAMPTR   set up new atom cell with pointer to name
        JSR   GETCEL     in CAR of dotted pair
        INX              set low order bit of form to indicate atom
        STX   FORM       save in FORM for later
        JSR   GETCEL   link new atom onto OBLIST
        LDAA  OBLIST
        STAA  CDR,X
        LDAA  OBLIST+1
        STAA  CDR+1,X
        BRA   RATRET   and return with FORM in X-reg

* here is a stripped down version of CKNUMB.  It accepts
*  only one-digit numeric atoms.  Otherwise it
*  returns with C-bit set.
CKNUM3  LDX   HAMPTR   look at name
        LDAA  0,X      get first char
        SUBA  #'0      subtract out ASCII code for zero digit
        BIT   #$80     not a digit if negative
        CMPA  #9       bigger than 9?
        BGT   BADNUM   not a digit
        TST   1,X      more than one digit?
        BNE   BADNUM   not allowed (for now)
        LDAB  #$80     A-OK, build up numeric form in FORM
        STAB  FORM
                       shift value around so that bit 1 is left oper
                         for garbage collector
        RORA
        ROLB
        ASLA
        RORB
        ROLA           bits 1,2,3 --> 2,3,4; bit 0 --> bit 0
        STAA  FORM+1   store low order byte of form
        LDX   FORM       and return it in X-reg
        CLC              with C-bit clear.
        RTS
BADNUM  SEC            bad number, return with C-bit set.
        RTS
```

Listing 5: RATOM accepts characters one at a time from the terminal and builds them into atoms.

```
* return form in X-reg for next atom typed in
RATOM   LDX   SPCPTR   save pointer to first open spot in name area
        STX   HAMPTR
        LDAA  PEEKC    PEEKC always contains the next character typed.
SKIPLUP CMPA  #$20     is PEEKC a separator (space or any control character)?
        BGT   GNAM     no, go collect name of atom
        BSR   READC    yes, get next char
        BRA   SKIPLUP    and loop.

* get next char of input, and store in global PEEKC
READC   JSR   GETC     use get char routine in ACIA monitor
        STAA  PEEKC    save it in PEEKC
                         and return with it also in A-reg.

* collect name of atom, build up form, etc.
GNAM    BSR   COPYC    save char in name area
        BSR   SPCLCH   is it a special char ("(" or ")")?
        BNE   RCLUP    no, collect multi-character name
        DCR   RCLUP    yes, update PEEKC
        BRA   SCANOL     and go scan OBLIST
RCLUP   BSR   READC    get next character of name
        CMPA  #$20     separator?
        BLE   SCANOL   yes, all done with name
        BSR   SPCLCH   special char?
        BEQ   SCANOL   yes, end atom now
        DCR   RCLUP    no, copy this char to name area
        BRA             and loop.

* copy char in A-reg to name area, and advance name space pointer
COPYC   LDX   SPCPTR   point to empty slot in name space
        STAA  0,X      store the char
        INX            increment the pointer
        STX   SPCPTR
                         and return.
        RTS

* check if char is a special char (i.e. "(" or ")")
SPCLCH  CMPA  #'(      1. paren?
        BEQ   SPCLY
        CMPA  #')      or r. paren?
SPCLY   RTS            return with Z-bit set appropriately.

* null-terminate name, check if numeric, scan OBLIST if not.
SCANOL  CLRA
        DCR   COPYC    check if numeric...
        JSR   CKNUMB   yep, go return numeric form
        BCC   GETNUM   nope, scan the OBLIST for the symbolic atom
        LDX   LSTPTR   using LSTPTR to point to the elements of the list
SCANLP  STX   HAMPTR     end of list, must be new atom
        BEQ   NEWATM   get next atom
        LDX   CAR,X    clear low-order bit of form
        DEX            get pointer to the atom's name
        LDX   CAR,X    compare with that at HAMPTR
        CPX   HAMPTR   found it! return pre-existing atom
        BNE   NXTSTR   not equal, go on to next atom
        LDX   CDR,X
        BRA   SCANLP

* compare strings pointed to by X-reg and HAMPTR
* Z-bit set if equal
* clobbers X-reg, A-reg, T-reg
COMPAR  STX   XTMP2    save X-reg temporarily
        LDY   HAMPTR   point at new atom name
CMPLP   LDAA  0,X      get next two chars
        LDAB  1,X
```


(a)

```
(DEF PRINT (LAMBDA (F)
    (PROGN
        (PRINR F)
        (PATOM NEWLINE)
        F
    )
))

(DEF PRINR (LAMBDA (F)
    (COND
        ((ATOM F) (PATOM F))
        (T (PROGN
            (PATOM LPAREN)
            (PRINR (CAR F))
            (PRINL (CDR F))
            (PATOM RPAREN)
        ))
    )
))

(DEF PRINL (LAMBDA (L)
    (COND
        ((DTPR L) (PROGN
            (PATOM SPACE)
            (PRINR (CAR L))
            (PRINL (CDR L))
        ))
    )
))
```

(b)

```
* type out a form, fully parenthesized, and then go to a new line.
PRINT   JSR     PUSHX   save X-reg on stack
        BSR     PRINR   simply pass the buck to recursive PRINR
        LDX     CRLFAT  type out CR/LF
        BSR     PATOM     using PATOM
        JMP     POPX    restore X-reg and return.

* type out a form, with no CR/LF
*  clobbers X-reg
PRINR   JSR     ISATOM  is the form an atom?
        BCC     PATOM   yes, pass the buck to PATOM
        JSR     PUSHX   nope, stack the X-reg
        LDX     LPARAT  type out a "("
        BSR     PATOM
PRINL   JSR     TOPX    restore the X-reg
        LDX     CAR,X   type out the CAR
        BSR     PRINR     (recursively!)
        JSR     POPX    restore pointer to the dotted pair
        LDX     CDR,X   advance to next dotted pair in linked list
        JSR     ISDTPR    is there a next dotted pair?
        BCS     PRPAR   nope, go type a ")"
        JSR     PUSHX   yep, save the new X-reg again
        LDX     SPACAT  type out a space
        BSR     PATOM
        BRA     PRINL     and loop around.
PRPAR   LDX     RPARAT  type out a ")"
        BRA     PATOM     and return (through PATOM).
```

Listing 6: LISP and M6800 assembly code of the PRINT routine.

atom typed. If this atom is anything but the atom "(" READ simply returns the atom as its result. If the atom returned by RATOM is "(", READ calls itself recursively until it gets the atom ")", meanwhile stringing the forms returned together as the CARs on a linked list of dotted pairs. This could be written as in listing 2.

In the LISP functions we are assuming that the atoms LPAREN and RPAREN were initialized to point to the atoms with print names "(" and ")" respectively. Notice that in the LISP version, READ accomplishes the loop of the machine code version with recursion in READL. The routines LSTINI, LSTADD, and LSTEND used in the assembly language version build up a linked list of dotted pairs, using two pointers on a stack, one to the first dotted pair, one to the dotted pair at the current end of the linked list. The pointers are on a stack so that READ may call itself recursively. The stack is actually a linked list itself. The linked-list stack is manipulated with the routines in listing 3. With these routines it is straightforward to implement LSTINI, LSTADD, and LSTEND for use in READ. These routines are shown in listing 4.

The primitive function RATOM turns out to be the real workhorse of READ. It is stuck with the job of accepting characters one at a time from the terminal, and building them up into an atom. RATOM must distinguish symbolic atoms from numeric atoms, and build up the corresponding forms. Atoms are in general separated by spaces, tabs, or carriage returns. However a few special characters always form single-character atoms when encountered (eg: "(" and ")") without any separator characters necessary.

In the author's LISP system RATOM is relatively sophisticated, allowing for atoms with spaces in their names if they are quoted ("..."). Also the single quote character ("'") is given special significance, as are "[" and "]". However a simpler RATOM is quite enough for an initial implementation. To make this exposition simpler, only single digit numeric atoms will be allowed. Certainly in an eventual implementation, multidigit numeric atoms, optionally preceded by a minus sign would be accepted.

In this RATOM, the characters are copied into an area set aside to hold the names of atoms as they are input. A null character (ASCII code zero) is used to terminate the name, when a separator or special character is encountered. If the name is entirely numeric, then the atom is a numeric atom, and the form is simply the value of the number, with the high order bit set, and one other bit left zero for use in the garbage collector. Otherwise the atom is a symbolic atom, and a scan is made of the OBLIST for a pre-existing atom with the same name. If one is found, the characters just typed in are thrown away and a form specifying the pre-existing atom is returned. If the atom is a new one, a 4 byte cell is allocated (using GETCEL defined in listing 4) and a pointer to the new atom is added to the OBLIST. A form specifying the new atom is returned. The M6800 assembly language code for this is in listing 5.

PRINT Function

PRINT is the second major recursive function comprising the LISP interpreter. It takes a single form as argument, and types the value as a fully parenthesized LISP expression. PRINT simply calls the more primitive function PATOM when it is given an atom to type. Otherwise, PRINT types a left parenthesis, calls itself recursively to type out the elements of the list, and then types a right parenthesis. In any case, PRINT always types out a carriage-return/line-feed at the end. This can be coded as in listing 6.

In the LISP routines, the special function PROGN is used. PROGN simply evaluates all of its arguments in sequence, and then returns the value of the last one as the value of the entire PROGN. The two functions ATOM and DTPR are used to test the type of a LISP object. ATOM returns T if the argument evaluates to an atom — symbolic, numeric, or NIL. Otherwise ATOM returns NIL. DTPR is the exact opposite. It returns T if the argument evaluates to a dotted pair, and returns NIL otherwise. Such functions which return either T or NIL are called "predicates" in LISP in analogy with predicates as used in symbolic logic. Such functions in other languages are called *Boolean* functions.

```
*  save it in ALP (recursive call on EVAL
*  will clobber FORM)
       LDX   CAR,X     evaluate CAR to get LAMBDA or SUBR expression
       PSR   EVAL        (recursively!)
       JSR   ISATOM    is result an atom?
       BCC   EVLERR    yes, illegal expression!
* X-reg should now be pointing to either
* (LAMBDA (...) ...)
* or
* (SUBR . funny-atom)
       STX   FLP       save pointer to function def
       LDX   CAR,X     now check the CAR for LAMBDA or SUBR
       CPX   SUBRAT
       BEQ   EVLSUB    yes, go evaluate args, etc.
       CPX   LAMBAT    LAMBDA?
       BEQ   EVLLAM    yes, go apply the LAMBDA
* illegal expression to EVAL, go report error to user
EVLERR LDX   #ERRMSG   error break routine expects error message
       JSR   ERRBRK      pointed to by X-reg.
* normal or error return, restore ALP, FLP, and NLP
* return with result of evaluation in X-reg and FORM
EVLXIT STX   FORM      save return value
       JSR   POPX      restore NLP, FLP, ALP
       STX   FLP
       JSR   POPX
       STX   NLP
       JSR   POPX
       STX   ALP
       LDX   FORM      and return with result in X-reg
       RTS               and FORM.

* evaluate the arg list, call the machine code subroutine.
EVLSUB LDA   EVALS     EVAL list of args (pointer to it in FLP)
       LDX   FLP         point at CDR of function def
       LDX   CDR,X
       BLE   EVLERR    error if numeric or NIL atom
       JSR   ISATOM
       BCC   EVLERR    error if not atom at all
       DEX             point to print name of funny atom
       LDX   CAR,Y
       LDAA  0,X       check for presence of special string
       CMPA  #$21        (hex $21,$00 used on this system)
       BNE   EVLERR
       LDAA  1,X
       CMPA  #$00
       BNE   EVLERR
       JSR   2,X       SUBR looks OK, call subroutine
       STX   FORM      save returned result
       JSR   POPALS    free up list of EVALed args
       LDX   FORM      restore X-reg
       BRA   EVLXIT      and return from EVAL.

* evaluate call on (LAMBDA (J K) ...)
EVLLAM JSR   EVALS     evaluate list of args
       JSR   EVLNSV    save old values of formals on stack
       JSR   EVLRSO    store new values for formals
       LDX   FLP       point to body of LAMBDA
       LDX   CAR,X
       JSR   EVAL      evaluate body (recursively!)
       STX   FLP       save result form temporarily
       JSR   EVLRST    restore old values of formals
       LDX   FLP       restore X-reg
       BRA   EVLXIT      and return from EVAL.
```

Listing 7: A simplified version of PATOM which assumes single digit atoms.

```
* given atomic form in X-reg, type out name on terminal
*  preserves X-reg
PATOM  STX   FORM      save X-reg for later
       BEQ   PNIIAT    go print 'NIL' if form is zero
       BLT   PNUMAT    go print numeric atom (high bit set)
       DEX             symbolic atom, clear low bit of form
       LDX   CAR,X     get pointer to print name of atom
       BRA   PNAME       and go type it out.
*
NILNAM FCC   'NIL'     string to print for NIL form
       FCB   0           (the null terminator)
*
PNIIAT LDX   #NILNAM   point to a null-terminated string "NIL"
PNAME  LDAA  0,X       type out the chars one at a time
       BEQ   PATDUN      until null char,
       JSR   OUTCH       using the ROM monitor put char routine.
       INX             advance to next character of name
       BRA   PNAME       and loop around.
*
PATDUN LDX   FORM      restore X-reg
       RTS               and return.

* simplified version of numeric atom type-out
*  form already stored in FORM. Only look at low order byte for now.
PNUMAT LDAA  FORM+1    get low order byte, and shift it a bit
       LSRA              bits 2,3,4 --> 1,2,3
       ADCA  #0          bit 0 --> bit 0
       ADDA  #'0       add in ASCII code for zero
       CMPA  #'9       more than a single digit?
       BLE   PNM2
       LDAA  #'*       yes, simply type out "*" for now.
PNM2   JSR   OUTCH     send digit to monitor put char routine
       BRA   PATDUN      and return from PATOM.
```

Listing 8: A simplified version of EVAL.

```
* evaluate passed in X-reg, return result in X-reg.
EVAL   STX   FORM      save form temporarily
       BLE   NUMNIL    numeric atom or NIL, simply return
       LDAA  FORM+1    is this a symbolic atom?
       RORA            (rotate bit 0 into carry bit)
       BCC   EVLDTP    no (low bit clear), must be dotted pair
       DEX             yes (low bit set), load current value of atom
       LDX   CDR,X
       STX   FORM      store it back in FORM
NUMNIL RTS               and return with value in X-reg and FORM.

* evaluate a function call (function specified by CAR of dotted pair)
EVLDTP LDX   ALP       save value of ALP, FLP, NLP on stack
       JSR   PUSHY
       LDX   FLP
       JSR   PUSHX
       LDX   NLP
       JSR   PUSHX     (now ALP, FLP, and NLP are "local" variables)
       LDX   FORM      point back to original form
```


Nowhere in the routines for PRINT, nor for that matter in the routines given earlier for READ, is the allowance made for the input or output of list structures which require the use of "dot" notation. A structure like (A B C . D) could not be input, and the above PRINT routines would type it out as (A B C), simply assuming that the atom which ended the linked list was NIL. It turns out that the changes necessary to implement dot notation are quite straightforward. For example, to add it to the LISP version of PRINT, only the routine PRINL need be rewritten, as follows:

```
(DEF PRINL (LAMBDA (L)
  (COND
    ((DTPR L) (PROGN
      (PATOM SPACE)
      (PRINR (CAR L))
      (PRINL (CDR L))
    ))
    ((EQ L NIL) NIL)
    (T (PROGN
      (PATOM SPACE)
      (PATOM DOT)
      (PATOM SPACE)
      (PATOM L)
    ))
  )
))
```

A corresponding change could be made to the assembly language routines.

As with the primitive function RATOM, the function PATOM turns out to be more difficult to implement than the recursive PRINT. PATOM must distinguish between symbolic atoms, numeric atoms, and NIL, and act accordingly. With symbolic atoms, PATOM simply types the null-terminated name of the atom. With numeric atoms, PATOM must convert back from the internal representation of the numeric value, to the string of ASCII characters which represent the number. With NIL, PATOM simply types 'NIL'. Listing 7 is a simplified version of PATOM with numeric atoms of only a single digit.

EVAL Function

The EVAL function is the *heart* of the LISP interpreter. EVAL accepts one form as an argument, and evaluates it according to the LISP convention: the value of NIL is NIL, the value of a numeric atom is itself, the value of a symbolic atom is the form associated with the atom, and the value of a list is determined by applying the function specified by the CAR of the list to the list of arguments which make up the CDR of the list.

In most LISP systems at least two distinct kinds of functions exist, SUBRs and LAMBDAs. SUBRs are the built-in functions of the LISP system, written in machine code (like CAR, CDR, PATOM). LAMBDAs are the user-defined functions, defined like (DEF GCD (LAMBDA (X Y) ...)). The effect of such a DEF is simply to define the list (LAMBDA (X Y) ...) as the value associated with the atom GCD.

The type of object used to specify a SUBR function varies among LISP systems. Frequently a new type of object is defined, called CODE, distinct from atoms and dotted pairs. A second alternative is to treat SUBRs like a funny kind of atom. The author's LISP system treats the bytes which make up the machine code of the SUBR like the print name of an atom. The SUBR is then specified by a dotted pair, with the CAR being the atom "SUBR" to identify the type of function, and the CDR being this atom with the funny print name. In fact the print name is prefixed with a special string which is unlikely to occur in a normal atom's print name, and hence PATOM could detect that the print name was not typeable, and simply type, say, "!" instead. In addition EVAL can check for the presence of this special string at the beginning of the print name to avoid treating a normal atom's print name as machine code. This method for specifying SUBRs avoids introducing an additional type, but the added complication in PATOM and EVAL may rule out the method in some implementations.

When EVAL is given a list to evaluate, it first evaluates the CAR of the list (recursively). The evaluation of the CAR should be either a LAMBDA expression, or a SUBR expression. If the evaluation of the CAR is an atom, or a list not headed by LAMBDA or SUBR, then EVAL stops, and indicates an error to the user.

If the CAR of the list gives a LAMBDA expression, the arguments to the function call are evaluated one at a time and saved on a list. The value associated with the "formal" arguments of the LAMBDA expression (eg: X and Y to the GCD routine given earlier) are saved on the stack. These formal arguments are then set one at a time to have the value of the corresponding actual arguments to the function (which were evaluated already). Finally, the "body" of the LAMBDA expression is evaluated, with the formal arguments now holding their new values. The result of evaluating the body is the result of the original function call. As a last step, EVAL restores the original values of the formal arguments.

Following the details of evaluation of such a function call is very difficult at first. The sequence of these steps is critical: evaluate actual arguments, save old values of formal arguments, set new values of formal arguments, evaluate body of LAMBDA, restore old values of formals. With any other sequence there is a chance that changes to the formal arguments of this function might interfere undesirably with the values of atoms in the calling routine's environment. These formal arguments are supposed to be strictly "local," that is, the choice of a name for a formal argument should be a strictly local decision, having no impact on variables with the same name in calling routines. Observing these rules allows LISP functions to be freely recursive. As the above examples of routines demonstrate, this recursion is in fact heavily used in LISP programming.

The steps in applying a SUBR function are simpler, because there are no formal arguments to worry about. EVAL simply evaluates the arguments to the SUBR, and passes them as a list to the machine code subroutine. EVAL expects the result of the SUBR to be left in register X when the subroutine returns.

This much of EVAL can be implemented on the M6800 as in listing 8.

The routines EVLALS, POPFRE, EVLNSV, EVLRSO, and EVLRST have not been included in listing 8 for brevity's sake. They are all relatively straightforward

routines, making heavy use of GETCEL, PUSHX, POPX, and FRECEL to build up and then release the lists of saved values.

Two additional types of LISP functions, normally recognized by an EVAL function, are called NLAMBDAs and NSUBRs (or FSUBRs, or FEXPRs if you prefer). These types of functions take their argument lists un-EVALed. NSUBRs are simply passed the CDR of the original function call list, instead of a list of evaluated arguments. Similarly, NLAMBDAs are provided with only a single argument, the list of unevaluated arguments. Without NSUBRs it is necessary for EVAL to recognize functions like COND as special cases, so that their argument list is not immediately evaluated. NSUBRs are specified in the same way as SUBRs, with the atom "NSUBR" replacing "SUBR" in the CAR of the dotted pair. PRINT will type out NSUBRs as "(NSUBR .!)"

NLAMBDAs are very useful for creating elaborate user-defined functions which take argument lists that are as or more complicated than COND. NLAMBDAs are necessary anytime the number of arguments is variable, or some of the arguments are wanted unevaluated.

To incorporate NLAMBDAs and NSUBRs in the above EVAL routines, two additional checks must be added immediately prior to EVLERR:

```
        .
        .
        .
BEQ EVLLAM      ...
CPX NSUBAT      NSUBR?
BEQ EVLNSU      yes, go call machine code
                subroutine
CPX NLAMAT      NLAMBDA?
BEQ EVLNLA      yes, pass list of args as single
                argument
* illegal exp...
EVLERR
        .
        .
        .
```

and the additional routines EVLNSU and EVLNLA must be included. Both of these routines are simpler than the corresponding routines EVLSUB and EVLLAM.

To make EVAL useful, some number of built-in SUBRs and NSUBRs must be written. The number of such built-in primitives can be kept quite small in LISP if they are chosen carefully. Most routines can be implemented as user functions if a few primitives exist. The primitives will certainly include PATOM, RATOM, EVAL, CAR, CDR, CONS, COND, SET, ADD, SUB, EQ, GREATER, ATOM, and NUMBER. All but SET and NUMBER have been used in the LISP function listings. SET is the primitive LISP assignment function. SET takes an atom and a value, and sets the value associated with the given atom to be the given value. NUMBER is a predicate function like ATOM, and simply returns T when its argument is a numeric atom. Listing 9 is an example of one of these primitives, the SUBR EQ.

Notice that the SUBRs and NSUBRs will start with the preface string (hex 21, 00 is used in this system). The argument list is always pointed to by ALP. Also notice that the SUBR may not assume that the proper number of arguments were supplied. The general rule is to treat unspecified arguments as though they were NIL. In EQ above, this gives some rather strange behavior, where simply (EQ) will always return T. It still remains for the implementor to initialize the atom EQ to point to a dotted pair, (SUBR . funny-atom), with the print name of the funny atom set to point to the code at EQSBR as shown in listing 9. The final section of this article goes over some of the problems involved with this kind of initialization.

Garbage Collector

A *garbage collector* eventually becomes essential in any LISP system. It is possible to create dotted pairs that are no longer accessible to a LISP program by any path. This happens, for example, if a function like REPLACE is called and then the value returned simply PRINTed but not saved as a LISP atom. This cannot go on for long before all of the free space is used up with dotted pairs. The garbage collector's job is to find all of the dotted pairs.

The various algorithms for locating such jetsom of the LISP function evaluation process are all quite intricate. The basic idea is always to trace systematically down every list structure to its component atoms, marking every dotted pair encountered along the way. If a dotted pair is encountered which is already marked, then that branch of the list structure is assumed to be already fully traced. The garbage collector then makes a sequential scan of all of memory space occupied by dotted pairs, and links together all unmarked dotted pairs onto a special list, the *free* list. During the scan, the marked dotted pairs are simply skipped over, because they are assumed to still be a part of some useful list structure. When a marked dotted pair is skipped over, its mark is also cleared in anticipation of future garbage collections, when it might no longer be so lucky.

The difficulty with this trace and collect algorithm is that each dotted pair points to possibly two more dotted pairs, so during the tracing phase the garbage collector must eventually follow both paths. What this means

```
* two argument SUBR EQ
*  return T if given identical forms, NIL otherwise
EQSBR   FCB     $21     special preface string
        FCB     $00
* ALP points to the list of evaluated arguments
        LDX     ALP     get first arg
        BEQ     TRUE    no args is equivalent to
                        (EQ NIL NIL)
*                        which should return T.
        LDX     CAR,X   save first arg temporarily
        STX     XTMP
        LDX     ALP     pick up second arg
        LDX     CDR,X
        BEQ     EQSNIL  (EQ X) is equivalent to
                        (EQ X NIL)
        LDX     CAR,X
EQSNIL  CPX     XTMP    are the forms identical?
        BEQ     TRUE    yes, return T.
        LDX     ZERO    no, return the NIL form
        RTS
*
TRUE    LDX     TATOM   return T atom
        RTS
```

Listing 9: EVAL may have built in primitives to expand the language. This is an example of the primitive SUBR EQ.

is that a second indication must be made on each dotted pair, indicating that the garbage collector is now busy tracing the CAR of this dotted pair, and will be returning later to trace the CDR of the dotted pair.

During the tracing phase, the garbage collector might very well be thought of as an ant determined to visit every branch of a tree. It goes out to the tip of each branch, but as it returns it must remember whether it has already traversed the other paths going out from each branching point. Even this analogy underrepresents the difficulty of a garbage collector, because the ant can simply turn around when it reaches the tip of a branch, but the garbage collector would normally have no clue as to how to climb back toward the root of a list structure once it gets out on a distant dotted pair.

The solution to the garbage collector's problem is to either reverse all the pointers in the list structure as it forays out to the terminating atoms and then reset the pointers on the way back in, or to keep a list of all dotted pairs which still require that their CDRs be traced. The first solution is like stringing a spool of thread behind you as you venture into an unexplored cave, following the thread back toward the mouth of the cave when you reach a dead end. Of course the same danger exists; that the delicate thread leading you back to the starting point might get tangled or broken.

The second solution is simpler, but suffers from the grave problem that it requires room to store the list of partially visited dotted pairs, and garbage collectors tend to be called upon at times when there is no more room to spare. In fact, the list of partially visited pairs need get no longer than the maximum "depth" of any list structure in the system, so that by setting aside a small portion of memory reserved for the use of the garbage collector's list, the implementor can get by with coding a much simpler tracing algorithm.

The author's system uses the pointer reversal method, and he will testify to the unlimited number of obscure problems which can appear during the debugging phase of its implementation.

It should be clear now why it was important to leave one bit in each form, and hence two bits per dotted pair, free for the use of the garbage collector. The bit in the CAR form can be used to indicate that the dotted pair has been visited once, and the bit in the CDR can be used to indicate that both paths from the dotted pair have been traced. These bits are only used during garbage collection, but because the garbage collector may be called at any time when GETCEL finds that there are no more 4 byte cells on the free list it may, in fact, run at almost any moment.

Because of this unpredictability, a LISP system with a garbage collector must be coded "defensively," jealously protecting any dotted pair allocated but not yet added to some accessible list structure. The machine code routines given in the listings do not all adhere to this rule. The reason for ignoring the garbage collector in the development thus far was simply to keep the design of the routines simple and relatively intuitive.

If the reader intends to include a garbage collector in an implementation of a LISP interpreter, more care must be taken. For example, two versions of the routine PUSHX would be defined, normal PUSHX and PROPSH (protected push). The PROPSH would be used when the 16 bit value being pushed on the stack pointed to list structure which might not be accessible in any other way, and hence might get collected in the next garbage collection scan. PROPSH avoids this danger by marking the cell used to store the saved value so that the garbage collector will know to trace this form and its descendents.

Initialization

It is ironic, but somehow appropriate, that the section on initialization comes at the end of this article. Frequently it is in fact one of the last things an implementor thinks about. That is probably because initialization is one of the biggest difficulties facing the implementor of any language: assembler, interpreter, or compiler. By initialization is meant the inevitably awkward methods of getting the symbol tables, or the OBLIST in LISP preloaded with the names which are to be built-in to the system. Most of the routines written to enter symbols into symbol tables, or to add new atoms to the OBLIST, are all oriented toward names entered by the *user* of the language processor. The initialization phase of the system becomes quite complicated because of this orientation. The methods finally chosen are, in general, tedious, requiring a lot of special preparation by the writer of the intialization routine.

The best way to avoid these initialization difficulties is to spend a little extra effort in designing a few nice routines for taking information out of tables which are convenient for the *implementor* to set up and modify, and let these routines do the intricate bit-twiddling work necessary to get the objects in shape for the symbol table, or the OBLIST.

In the author's LISP initialization module are routines to build up dotted pairs in the form required for SUBRs and NSUBRs, and routines to allocate 4 byte cells for built-in atoms. The atom initialization routines are given the address of a contiguous table of null-terminated ASCII names, each followed by the address of a memory cell where the form specifying the new atom should be stored. This is where the symbols like TATOM, SUBRAT, LAMBAT, etc came from. They refer to memory locations in the base page of the M6800 (0 thru 255), where the forms specifying the atom T, SUBR, and LAMBDA, etc, are stored. The table to initialize these atoms was simply:

```
ATMTAB  FCC     'T'
        FCB     0
        FDB     TATOM
        FCC     'SUBR'
        FCB     0
        FDB     SUBRAT
        FCC     'LAMBDA'
        FCB     0
        FDB     LAMBAT
        FCC     ...
          .
          .
          .
        FDB     ...
        FCB     0       null-name terminates table
```

Although writing the special initialization routines was initially time-consuming, it was more than compensated for by the ease of adding more built-in atoms as the system grew.

Conclusion

We have traced through the implementation of a LISP interpreter and looked at a specific example for the M6800 processor. For further information on the garbage collecting routines and a complete listing of the interpreter, see listing 10.

Listing 10: The entire LISP interpreter for the M6800 with built in garbage collection.

```
                        NAM     LISP
                *SIMPLE LISP INTERPRETER
                        OPT     NOG
0200            OBEGAD  EQU     $200
0000            CAR     EQU     0
0002            CDR     EQU     2
0004            EOI     EQU     4
                * DIRECT PAGE STORAGE CELLS
0020                    ORG     $20
0020 03 A0      BEGPTR  FDB     BEGADR    THIS SHOULD BE IN EACH MODULE
0022            STR1    RMB     2
0024            STR2    RMB     2
0026            STP1    RMB     2
0028            STP2    RMB     2
0026            XTMP    EQU     STP1
0028            XTMP2   EQU     STP2
002A            FREPTR  RMB     2
002C            SPCPTR  RMB     2
002E            ENDPTR  RMB     2
0030            SYMLST  RMB     2
0032            NAMPTR  RMB     2
0034            FORM    RMB     2
0036            LSTPTR  RMB     2
0038            SYMPTR  RMB     2
003A            CELPTR  RMB     2
003C            SPSAVE  RMB     2
003E            STKPTR  RMB     2
0040            CMPTMP  RMB     2
0042            CMPTM2  RMB     2
0044            ALP     RMB     2
0046            NLP     RMB     2
0048            FLP     RMB     2
004A            NILATM  RMB     2
004C            LPARAT  RMB     2
004E            RPARAT  RMB     2
0050            DOTATM  RMB     2
0052            SQUTAT  RMB     2
0054            QUOATM  RMB     2
0056            LAMATM  RMB     2
0058            NLAMAT  RMB     2
005A            SUBRAT  RMB     2
005C            MSUBAT  RMB     2
005E            TATOM   RMB     2
0060            CONATM  RMB     2
0062            RSETAT  RMB     2
0064            EOIATM  RMB     2
0066            SUBLS2  RMB     2
0068            LBRKAT  RMB     2
006A            RBRKAT  RMB     2
006C            CUREVL  RMB     2
006E            SONPTR  RMB     2
0070            DADPTR  RMB     2
0072            GCTEMP  RMB     2
0074            GCFREE  RMB     2       NUMBER OF FREE CELLS AFTER LAST GCOL
0076            SUBLS3  RMB     2       INITIALIZED IN SUBRS3
                * SINGLE CHAR SLOTS ...
00F0                    ORG     $F0
00F0            PEEKC   RMB     1
00F1            RSIFLG  RMB     1
                * GLOBAL CONSTANTS
00FC                    ORG     $FC
                * END OF MEMORY, MINUS 192 FOR NUMFRM,FRMNUM TABLES
00FC 6F 40      ENDMEM  FDB     $7000-192 MUST BE MULTIPLE OF 256, MINUS 192
                * ZERO WORD
00FE 00 00      ZERO    FDB     0
                * GLOBAL VECTORS
0100                    ORG     $100
0100            EVAL    RMB     3
0103 7E 02 E5           JMP     READ
0106            PRINT   RMB     3
0109                    RMB     3         USED TO BE TYPSUT (R.I.P.)
010C            GETCEL  RMB     3
010F            FRECEL  RMB     3
0112            PUSHX   RMB     3
0115            POPX    RMB     3
0118            TOPX    RMB     3
011B            EVLATM  RMB     3
011E            SETATM  RMB     3
0121            LOOKUP  RMB     3
0124            GNXTL   RMB     3
0127            GFNXTL  RMB     3
012A            LSTINI  RMB     3
012D            LSTADD  RMB     3
0130            LSTAD2  RMB     3
0133            LSTEND  RMB     3
0115            LSTENO  EQU     POPX
0136 7E 02 67           JMP     ERREX
0139            ATMINI  RMB     3
013C            ISDTFR  RMB     3
013F            ISATOM  RMB     3
0142            GETC    RMB     3
0145            NUMINV  RMB     3
0148            NUMFRM  RMB     3
014B            FRMNUM  RMB     3
014E            PUTSYM  RMB     3
0151            PUTC    RMB     3
0154 7E 02 76           JMP     ERRBRK
0157            GETSYM  RMB     3
015A            STKFRG  RMB     3
015D            GCOL    RMB     3
0160            PROPSH  RMB     3
0163 7E 03 7C           JMP     GETCAR
0166            ISVAR   RMB     3
0169            PRINR   RMB     3
016C            PROPOP  RMB     3
                *
0200                    ORG     OBEGAD
                *
0200            START   EQU     *
0200 9F 3C              STS     SPSAVE
0202 DE 20              LDX     BEGPTR
0204 DF 2C              STX     SPCPTR
0206 DE FC              LDX     ENDMEM
0208 DF 2E              STX     ENDPTR
                * INITIALIZE TABLES FOR NUMFRM AND FRMNUM
                * LAST 128 BYTES OF MEM HOLD TABLE TO CONVERT HIGH 7 BITS OF BCD TO 6 BITS OF BINARY
                * PREVIOUS 64 BYTES HOLD TABLE TO CONVERT 6 BITS TO 7 BITS
                * BIT 6 OF 128 BYTE TABLE IS ALWAYS ON
                *
020A 96 FC              LDA A   ENDMEM    SET UP HIGH BYTE OF XTMP
020C 97 26              STA A   XTMP
020E 86 40              LDA A   #$40      INITIALIZE TABLES TO 40'S
0210 C6 C0              LDA B   #192
0212            NUMILP  EQU     *
0212 A7 00              STA A   0,X
0214 08                 INX
0215 5A                 DEC B
0216 26 FA              BNE     NUMILP
                * SET UP 64-BYTE TABLE
0218 DE FC              LDX     ENDMEM
```


1979 © BYTE Publications Inc

```
021A 4F             CLR A
021B          NUM64I EQU   *
021B 44             LSR A
021C A7 00          STA A  0,X
021E 08             INX
021F 48             ASL A
0220 8B 02          ADD A  #2
0222 19             DAA
0223 24 F6          BCC    NUM64I
              * SET UP 128-BYTE TABLE
0225 86 80          LDA A  #$80       START ADDR
0227 97 27          STA A  XTMP+1
0229 DE 26          LDX    XTMP
022B 86 40          LDA A  #$40
022D          NM128I EQU   *
022D C6 05          LDA B  #5         INNER COUNTER
022F          N128I2 EQU   *
022F A7 00          STA A  0,X
0231 08             INX
0232 4C             INC A
0233 5A             DEC B
0234 26 F9          BNE    N128I2
0236 08             INX               SKIP 3 BYTES
0237 08             INX
0238 08             INX
0239 81 72          CMP A  #50+$40    ALL DONE?
023B 26 F0          BNE    NM128I
              *             YEP, TABLES NOW SET UP
023D 4F             CLR A
023E 97 F0          STA A  PEEKC
0240 DE FE          LDX    ZERO
0242 DF 2A          STX    FREPTR
0244 DF 30          STX    SYMLST
0246 DF 3E          STX    STKPTR
0248 DF 4A          STX    NILATM
024A DF 6C          STX    CUREVL
024C DF 34          STX    FORM       FORM MUST ALWAYS BE A LEGALLY TRACEABLE FORM
              * INITIALIZE BUILT-IN ATOMS
024E BD 01 39       JSR    ATMINI
              *
0251          BIGLUP EQU   *
0251 7F 00 F1       CLR    RSTFLG     CLEAR RESET FLAG
0254 CE 03 9C       LDX    #PROMSG    '->'
0257 BD 71 18       JSR    PSTRNG
025A BD 02 E5       JSR    READ
025D BD 71 1E       JSR    PCRLF
0260 BD 01 00       JSR    EVAL
0263 8D 0E          BSR    PPRINT
0265 20 EA          BRA    BIGLUP
              *
0267          ERREX  EQU   *
0267 9E 3C          LDS    SPSAVE
0269 BD 71 18       JSR    PSTRNG
026C BD 71 1E       JSR    PCRLF
026F 3F             SWI               SOFTWARE INT (EXIT TO SWTBUG)
0270 7E 02 00       JMP    START      RESTART
              *
0273 7E 01 06 PPRINT JMP   PRINT      SAVE A FEW JSR'S
              *
0276          ERRBRK EQU   *          CONTINUABLE ERROR
0276 7C 02 D4       INC    EFLAG      INDICATE IN ERROR CODE
0279 BD 71 18       JSR    PSTRNG     PRINT ERROR MESSAGE
027C BD 71 1E       JSR    PCRLF
027F DE 34          LDX    FORM       PRINT CURRENT VALUE
0281 27 02          BEQ    ERRBR2
0283 8D EE          BSR    PPRINT
0285          ERRBR2 EQU   *
0285 DE 6C          LDX    CUREVL     PRINT CURRENT FORM BEING EVAL'D
0287 BD 03 7C       JSR    GETCAR
028A 25 02          BCS    ERRBR3
028C 8D E5          BSR    PPRINT
028E          ERRBR3 EQU   *
028E 96 F1          LDA A  RSTFLG     RESET OR RETBRK?
0290 27 0C          BEQ    ERRLUP
0292          ERRSET EQU   *
0292 DE 64          LDX    EOIATM     YEP, RESULT IS NOTHING
0294 8B 30          ADD A  #'0        AT THIS LEVEL?
0296 B1 02 D4       CMP A  EFLAG
0299 25 33          BLO    ERRETN     NOPE, KEEP RESETTING
029B 7F 00 F1       CLR    RSTFLG     YEP, CLEAR RESET FLAG
029E          ERRLUP EQU   *
029E CE 02 D4       LDX    #EFLAG     ERROR PROMPT (E.G. "2:> ")
02A1 A6 00          LDA A  0,X        FIRST LEVEL?
02A3 81 31          CMP A  #'1
02A5 26 01          BNE    ERRLP2
02A7 08             INX               YEP, SKIP PAST DIGIT
02A8          ERRLP2 EQU   *
02A8 BD 71 18       JSR    PSTRNG
02AB BD 02 E5       JSR    READ       READ AND EVAL
02AE DF 34          STX    FORM
02B0 BD 03 7C       JSR    GETCAR     BUT FIRST , IS IT (CONT ...)
02B3 9C 60          CPX    CONATM     'CONT'?
02B5 27 0D          BEQ    ERRCON     YEP, RETURN WITH ARG TO CONT
02B7 DE 34          LDX    FORM       NOPE, EVAL FORM
02B9 BD 01 00       JSR    EVAL
02BC 96 F1          LDA A  RSTFLG     IN A RESET?
02BE 26 D2          BNE    ERRSET     YEP, CHECK LEVEL
02C0 8D B1          BSR    PPRINT     PRINT VALUE
02C2 20 DA          BRA    ERRLUP     AND LOOP AROUND
              *
02C4          ERRCON EQU   *          CONTINUE
02C4 DE 34          LDX    FORM       GET ARG
02C6 EE 02          LDX    CDR,X
02C8 BD 03 7C       JSR    GETCAR
02CB BD 01 00       JSR    EVAL       EVAL ARG TO CONT AND RETURN
02CE          ERRETN EQU   *
02CE 7A 02 D4       DEC    EFLAG      BACK UP EFLAG
02D1 DF 34          STX    FORM       RETURN WITH RESULT IN FORM AND X-REG
02D3 39             RTS
              *
02D4 30       EFLAG  FCB   '0         IF NON-ZERO, NOW IN ERROR CODE
02D5 3A             FCC   ':> '
02D8 04             FCB   4
              *
02D9          ISFORM EQU   *
              * EXPECTS FORM IN X-REG
              * SETS C-BIT IF NOT A LEGAL FORM
02D9 DF 26          STX    XTMP       STORE IT AND SET CC'S
02DB 2F 06          BLE    ISFRT      NIL OR A NUMBER, ~A-OK
02DD 9C 2E          CPX    ENDPTR     LEGAL CELL ADDR?
02DF 2A 02          BPL    ISFRT      YEP
02E1 0D             SEC               NOT LEGAL, SET C-BIT
02E2 39             RTS
02E3          ISFRT  EQU   *
02E3 0C             CLC               LEGAL, CLEAR C-BIT
02E4 39             RTS
              *
02E5          READ   EQU   *
              * READ ASSEMBLES A FORM, USING GETOSY'S
              * TO RETRIEVE THE TOKENS
              * FORM RETURNED IN X-REG AND "FORM"
              * A AND B-REGS CLOBBERED!
02E5 7F 02 F9       CLR    RBRKFG     INITIALIZE RIGHT BRACKET FLAG
02E8          READR  EQU   *
02E8 BD 03 3D       JSR    GETOSY
              * (ASSUME RESULT ALSO SAVED IN FORM)
02EB 9C 4C          CPX    LPARAT     IS IT A LEFT PAREN?
02ED 27 0B          BEQ    READL      YEP, RETURN RESULT FROM READL
```

```
02EF 9C 68          CPX    LBRKAT     LEFT BRACKET ("E")?
02F1 26 05          BNE    READRT
02F3 8D 05          BSR    READL      YEP, READ AND CLEAR RBRKFG
02F5 7F 02 F9       CLR    RBRKFG
02F8          READRT EQU   *
02F8 39             RTS               NOPE, SIMPLY RETURN WITH ATOM
              *
02F9 00       RBRKFG FCB   0          POSITIVE WHEN "]" FOUND
              *
02FA          READL  EQU   *
02FA BD 01 2A       JSR    LSTINI     INITIALIZE RDLST AND RDLPTR
02FD BD 01 5A       JSR    STKFRG     ADD A NEW FRAGMENT TO STACK
0300          RDL2   EQU   *
0300 8D E6          BSR    READR      GET NEXT FORM
0302 9C 4E          CPX    RPARAT     RIGHT PAREN?
0304 27 10          BEQ    RDLRT      YES, FINISH UP READL
0306 9C 50          CPX    PDTATM     "."?
0308 27 15          BEQ    RDLDOT     YES, GO FINISH UP WITH ONE MORE READ
              * FORM IS POINTING TO OBJECT TO BE ADDED TO LIST
030A 9C 64          CPX    EOIATM     PREMATURE END OF INPUT?
030C 27 05          BEQ    RDLEOI     YEP, TREAT LIKE "]"
030E BD 01 30       JSR    LSTAD2     ADD TO LIST
0311 20 ED          BRA    RDL2       GO GET NEXT FORM
              *
0313          RDLEOI EQU   *
0313 7C 02 F9       INC    RBRKFG     EOIATM ENCOUNTERED, CLOSE OFF UNFINISHED LISTS
0316          RDLRT  EQU   *
0316 BD 01 15       JSR    LSTEND     POP OFF RDLPTR
0319          RDLRT2 EQU   *
0319 BD 01 6C       JSR    PROPOF     POP OFF AND RETURN RDLST
031C DF 34          STX    FORM
031E 39             RTS
              *
031F          RDLDOT EQU   *
031F BD C7          BSR    READR      PULL IN NEXT FORM
0321 BD 01 33       JSR    LSTEND     FILL IN CDR OF LAST CELL
0324 BD 03 3D       JSR    GETOSY     NOW, REQUIRE ")" TO END IT ALL
0327 9C 4E          CPX    RPARAT
0329 27 EE          BEQ    RDLRT2     ALL DONE NOW!
              * ERROR IN LIST STRUCTURE...
032B          RLSTER EQU   *
032B BD 01 6C       JSR    PROPOF     POP OFF FORM BEING BUILT
032E DF 34          STX    FORM
0330 CE 03 8C       LDX    #RLERMS
0333 7A 00 F1       DEC    RSTFLG     FORCE IMMEDIATE RETURN FROM ERRBRK
0336 BD 02 76       JSR    ERRBRK
0339 7F 00 F1       CLR    RSTFLG
033C 39             RTS               AND RETURN WITH RESULT IN X-REG AND FORM
              *
033D          GETOSY EQU   *
              *  SAME AS GETSYM, ONLY CHECK FOR "'"
              * AND BUILD UP (QUOTE ...) IF FOUND
              *  ALSO DEAL WITH "]" FEATURE.
033D B6 02 F9       LDA A  RBRKFG     RIGHT BRACKET FLAG SET?
0340 26 36          BNE    GOSRPR     YEP, RETURN A ")"
0342 BD 01 57       JSR    GETSYM
0345 9C 6A          CPX    RBRKAT     RIGHT BRACKET?
0347 27 2C          BEQ    GOSRBK     YEP, SET RBRKFG AND RETURN A ")"
0349 9C 52          CPX    SQUTAT     "'"?
034B 26 27          BNE    GOSRT
034D BD 02 E5       JSR    READ       YES, PICK UP NEXT FORM
0350 DF 34          STX    FORM       SAVE POINTER
0352 BD 01 0C       JSR    GETCEL     PICK UP A CELL
0355 96 34          LDA A  FORM       FILL IN THE CAR
0357 A7 00          STA A  CAR,X
0359 96 35          LDA A  FORM+1
035B A7 01          STA A  CAR+1,X
035D DF 34          STX    FORM       SAVE IT AGAIN
035F BD 01 0C       JSR    GETCEL     ANOTHER CELL...
0362 96 54          LDA A  QUOATM     WITH "QUOTE" ATOM
0364 A7 00          STA A  CAR,X
0366 96 55          LDA A  QUOATM+1
0368 A7 01          STA A  CAR+1,X
036A 96 34          LDA A  FORM
036C A7 02          STA A  CDR,X
036E 96 35          LDA A  FORM+1
0370 A7 03          STA A  CDR+1,X
0372          GOSR0  EQU   *
0372 DF 34          STX    FORM       RETURN WITH FORM IN FORM
0374          GOSRT  EQU   *
0374 39             RTS
              *
0375          GOSRBK EQU   *
0375 7C 02 F9       INC    RBRKFG     SET RIGHT BRACKET FLAG
0378          GOSRPR EQU   *
0378 DE 4E          LDX    RPARAT     RETURN ")"
037A 20 F6          BRA    GOSR0
              *
037C          GETCAR EQU   *
              * RETURN CAR OF X-REG OR NIL IF NONE (SETS C-BIT IF NONE)
037C DF 26          STX    XTMP
037E 2F 08          BLE    GCANIL
0380 76 00 27       ROR    XTMP+1     ATOM?
0383 25 03          BCS    GCANIL
0385 EE 00          LDX    CAR,X      NO, RETURN CAR
0387 39             RTS               RETURN WITH C-BIT CLEAR
0388          GCANIL EQU   *
0388 DE FE          LDX    ZERO       RETURN NIL
038A 0D             SEC               AND C-BIT SET
038B 39             RTS
              *
7118          PSTRNG EQU   $7118
711E          PCRLF  EQU   $711E
              * TABLES, ETC.
038C 52       RLERMS FCC   'READ LIST ERROR'
039B 04             FCB   4
039C 2D       PROMSG FCC   '-> '
039F 04             FCB   4
03A0          REGADR EQU   *
              END    START
```

```
NO ERROR(S) DETECTED

 SYMBOL TABLE:

ALP    0044   ATMINI 0139   BEGADR 03A0   BEGPTR 0020   BIGLUP 0251
CAR    0000   CDR    0002   CELPTR 003A   CMPTM2 0042   CMPTMP 0040

CONATM 0060   CUREVL 006C   DADPTR 0070   DOTATM 0050   EFLAG  02D4
ENDMEM 00FC   ENDPTR 002E   EOI    0004   EQIATM 0064   ERRBR2 0285
ERRBR3 028E   ERRBRK 0276   ERRCON 02C4   ERRETM 02CE   ERREX  0267
ERRLP2 02A8   ERRLUP 029E   ERRSET 0292   EVAL   0100   EVLATM 011B
FLP    0048   FORM   0034   FRECEL 010F   FREPTR 002A   FRMNUM 014B
GCANIL 0388   GCFREE 0074   GCOL   015D   GCTEMP 0072   GETC   0142
GETCAR 037C   GETCEL 010C   GETDSY 033D   GETSYM 0157   GFNXTL 0127
GNXTL  0124   GOSRO  0372   GOSRBK 0375   GOSRPR 0378   GOSRT  0374
ISATOM 013F   ISDTPR 013C   ISFORM 02D9   ISFRT  02E3   ISVAR  0166
LAMATM 0056   LBRKAT 0068   LOOKUP 0121   LPARAT 004C   LSTAD2 0130
LSTADD 0126   LSTEND 0115   LSTEND 0133   LSTINI 012A   LSTPTR 0036
M128I2 022F   NAMPTR 0032   NILATM 004A   NLAMAT 0058   NLP    0046
NM128I 022D   NSUBAT 005C   NUM64I 021B   NUMFRM 0148   NUMILP 0212
NUMINV 0145   OBEGAD 0200   PCRLF  711E   PEEKC  00F0   POPX   0115
PPRINT 0273   PRINK  0169   PRINT  0106   PROMSG 039C   PROPOP 016C
PROPSH 0160   PSTRNG 7118   PUSHX  0112   PUTC   0151   PUTSYM 014E
QUOATM 0054   RBRKAT 006A   RBRKFG 02F9   RDL2   0300   RDLDOT 031F
RDLEDI 0313   RDLRT. 0316   RDLRT2 0319   READ   02E5   READL  02FA
READR  02E8   READRT 02F8   RLERMS 038C   RLSTER 0326   RPARAT 004E
RSETAT 0062   RSTFLG 00F1   SETATM 011E   SONPTR 006E   SPCPTR 002C
SPSAVE 003C   SQUTAT 0052   START  0200   STKFRG 015A   STKPTR 003E
STP1   0026   STP2   0028   STR1   0022   STR2   0024   SUBLS2 0066
SUBLS3 0076   SUBRAT 005A   SYMLST 0030   SYMPTR 0038   TATOM  005E
TOPX   0118   XTMP   0026   XTMP2  0028   ZERO   00FE
```

```
                        OPT    NOG
0300            OBEGAD  EQU    $0300     LISP ENDS WITH 38C
0000            CAR     EQU    0
0002            CDR     EQU    2
0004            EOI     EQU    4
                * DIRECT PAGE STORAGE CELLS
0020                    ORG    $20
0020 05 7D      BEGPTR  FDB    BEGADR    THIS SHOULD BE IN EACH MODULE
0022            STR1    RMB    2
0024            STR2    RMB    2
0026            STP1    RMB    2
0028            STP2    RMB    2
0026            XTMP    EQU    STP1
0028            XTMP2   EQU    STP2
002A            FREPTR  RMB    2
002C            SPCPTR  RMB    2
002E            ENDPTR  RMB    2
0030            SYMLST  RMB    2
0032            NAMPTR  RMB    2
0034            FORM    RMB    2
0036            LSTPTR  RMB    2
0038            SYMPTR  RMB    2
003A            CELPTR  RMB    2
003C            SPSAVE  RMB    2
003E            STKPTR  RMB    2
0040            CMPTMP  RMB    2
0042            CMPTM2  RMB    2
0044            ALP     RMB    2
0046            NLP     RMB    2
0048            FLP     RMB    2
004A            NILATM  RMB    2
004C            LPARAT  RMB    2
004E            RPARAT  RMB    2
0050            DOTATM  RMB    2
0052            SQUTAT  RMB    2
0054            QUOATM  RMB    2
0056            LAMATM  RMB    2
0058            NLAMAT  RMB    2
005A            SUBRAT  RMB    2
005C            NSUBAT  RMB    2
005E            TATOM   RMB    2
0060            CONATM  RMB    2
0062            RSETAT  RMB    2
0064            EQIATM  RMB    2
0066            SUBLS2  RMB    2
0068            LBRKAT  RMB    2
006A            RBRKAT  RMB    2
006C            CUREVL  RMB    2
006E            SONPTR  RMB    2
0070            DADPTR  RMB    2
0072            GCTEMP  RMB    2
0074            GCFREE  RMB    2         NUMBER OF FREE CELLS AFTER LAST GCOL
0076            SUBLS3  RMB    2         INITIALIZED IN SUBRS3
                * SINGLE CHAR SLOTS ...
00F0                    ORG    $F0
00F0            PEEKC   RMB    1
00F1            RSTFLG  RMB    1
                * GLOBAL CONSTANTS
00FC                    ORG    $FC
00FC            ENDMEM  RMB    2         INITIALIZED IN LISP
                * ZRO WORD
00FE 00 00      ZERO    FDB    0
                * GLOBAL VECTORS
0100                    ORG    $100
0100            EVAL    RMB    3
0103            READ    RMB    3
0106            PRINT   RMB    3
0109            TYPSUT  RMB    3
010C            GETCEL  RMB    3
```

```
010F            FRECEL  RMB    3
0112            PUSHX   RMB    3
0115            POPX    RMB    3
0118            TOPX    RMB    3
011B            EVLATM  RMB    3
011E            SETATM  RMB    3
0121 7E 03 FC           JMP    LOOKUP
0124            GNXTL   RMB    3
0127            GFNXTL  RMB    3
012A            LSTINI  RMB    3
012D            LSTADD  RMB    3
0130            LSTAD2  RMB    3
0133            LSTEND  RMB    3
0115            LSTENO  EQU    POPX
0136            ERREX   RMB    3
0139            ATMINI  RMB    3
013C            ISDTPR  RMB    3
013F            ISATOM  RMB    3
0142            GETC    RMB    3
0145            NUMINV  RMB    3
0148            NUMFRM  RMB    3
014B            FRMNUM  RMB    3
014E            PUTSYM  RMB    3
0151            PUTC    RMB    3
0154            ERRBRK  RMB    3
0157 7E 04 D4           JMP    GETSYM
015A            STKFRG  RMB    3
015D            GCOL    RMB    3
0160            PROPSH  RMB    3
0163            GETCAR  RMB    3
0166            ISVAR   RMB    3
0169            PRINR   RMB    3
                *
03D0                    ORG    OBEGAD
03D0            STREQ   EQU    *
                * STREQ EXPECTS 1ST ARG IN STR1, 2ND IN X-REG
03D0 DF 24              STX    STR2
03D2 DF 28              STX    STP2
03D4 DE 22              LDX    STR1
03D6 DF 26      STE2    STX    STP1
03D8 A6 00              LDA A  0,X
03DA DE 28              LDX    STP2
03DC A1 00              CMP A  0,X
03DE 26 18              BNE    UNEQ
03E0 08                 INX
03E1 4D                 TST A
03E2 27 10              BEQ    ISEQ
03E4 A6 00              LDA A  0,X
03E6 08                 INX
03E7 DF 28              STX    STP2
03E9 DE 26              LDX    STP1
03EB 08                 INX
03EC A1 00              CMP A  0,X
03EE 26 08              BNE    UNEQ
03F0 08                 INX
03F1 4D                 TST A
03F2 26 E2              BNE    STE2
                * NOTE: THE ABOVE LOOP IS "UNRAVELLED" FOR SPEED.
03F4 DE 24      ISEQ    LDX    STR2
03F6 4D                 TST A
                * RETURN WITH COND. CODE SET
03F7 39                 RTS
03F8 86 01      UNEQ    LDA A  #1
03FA 20 FB              BRA    ISEQ
                ***
03FC            LOOKUP  EQU    *
                * LOOKUP EXPECTS PTR TO NAME IN X-REG.
                * RETURNS PTR TO NEW ATOM CELL IN X-REG.
                * LOOKS ON SYMLST FOR EXISTING ATOM, ADDS CELL TO SYMLST
                *
03FC DF 32              STX    NAMPTR
03FE DF 22              STX    STR1
0400 E6 00              LDA B  0,X        ** 1ST CHAR OF NAME SAVED IN B-REG
0402 17                 TBA               COPY TO A FOR NUM TEST
0403 81 2D              CMP A  #'-        COULD THIS BE A NUMBER?
0405 26 09              BNE    LKUNM2
0407 6D 01              TST    1,X        "-" ALL ALONE?
0409 27 0D              BEQ    LKU1       YEP, TREAT LIKE A NON-NUMERIC ATOM
040B            LKUNUM  EQU    *
040B 08                 INX               POINT TO NEXT DIG
040C A6 00              LDA A  0,X        END OF SYM?
040E 27 68              BEQ    LKUNM3     YEP, WE HAVE A NUMBER!
0410            LKUNM2  EQU    *
0410 81 30              CMP A  #'0        <
0412 2D 04              BLT    LKU1       (NOT A NUM)
0414 81 39              CMP A  #'9
0416 2F F3              BLE    LKUNUM     KEEP CHECKING
0418            LKU1    EQU    *
0418 DE 30              LDX    SYMLST     POINT TO FIRST ATOM
041A 27 18              BEQ    EOL        (UNLESS LIST EMPTY)
041C            LKU2    EQU    *
041C DF 36              STX    LSTPTR
041E EE 00              LDX    CAR,X
0420 DF 34              STX    FORM       ** SAVE ADDR
0422 09                 DEX
0423 EE 00              LDX    CAR,X      ** PICK UP NAME FROM ATOM CELL
0425 E1 00              CMP B  0,X        ** QUICK CHECK  FOR 1ST CHAR MATCH
0427 26 05              BNE    LKU3
0429 BD 03 D0           JSR    STREQ
042C 27 2D              BEQ    LKU4
042E            LKU3    EQU    *
                * ADVANCE TO NEXT ATOM
042E DE 36              LDX    LSTPTR
0430 EE 02              LDX    CDR,X
0432 26 E8              BNE    LKU2
                * NOT FOUND, ALLOCATE TWO NEW CELLS
0434            EOL     EQU    *
0434 BD 01 0C           JSR    GETCEL
0437 96 32              LDA A  NAMPTR
0439 A7 00              STA A  CAR,X
043B 96 33              LDA A  NAMPTR+1
043D A7 01              STA A  CAR+1,X
043F 08                 INX               SET UP 'FORM'
0440 DF 34              STX    FORM
0442 BD 01 0C           JSR    GETCEL
0445 96 34              LDA A  FORM
0447 A7 00              STA A  CAR,X
0449 96 35              LDA A  FORM+1
044B A7 01              STA A  CAR+1,X
044D 96 30              LDA A  SYMLST
044F A7 02              STA A  CDR,X
0451 96 31              LDA A  SYMLST+1
0453 A7 03              STA A  CDR+1,X
0455 DF 30              STX    SYMLST     PUT NEW ATOM ON FRONT OF SYMLST
0457 20 14              BRA    FOUND
                *
0459            FNDNUM  EQU    *
0459 DF 34              STX    FORM       SAVE RETURN FORM
045B            LKU4    EQU    *
                * FOUND IT, RESET SPCPTR
045B 96 32              LDA A  NAMPTR
045D 91 20              CMP A  BEGPTR     (UNLESS NAME IS BUILT-IN)
045F 25 0C              BLO    FOUND
0461 22 06              BHI    LKU5
0463 96 33              LDA A  NAMPTR+1
0465 91 21              CMP A  BEGPTR+1
0467 25 04              BLO    FOUND
0469            LKU5    EQU    *
```

```
0469 DE 32          LDX   NAMPTR
046B DF 2C          STX   SPCPTR
                    * RETURN POINTER TO CELL
046D                FOUND EQU   *
046D DE 34          LDX   FORM
046F 9C 4A          CPX   #ILATM    NIL?
0471 26 04          BNE   FND2
0473 DE FE          LDX   ZERO      YEP, RETURN WITH ZERO
0475                FND1  EQU   *
0475 DF 34          STX   FORM
0477                FND2  EQU   *
0477 39             RTS
                    *
0478                LKUNM3 EQU  *
                    * BUILD UP A NUMBER
                    *  IS 16-BIT BCD NUMBER (+/-4999)
                    *  HIGH BIT SET AS INDICATOR
0478 DE 32          LDX   NAMPTR
047A 4F             CLR A           A AND B WILL BE NUMBER
047B C0 2D          SUB B #'-       SAVE STATUS OF '-' FLAG
047D D7 26          STA B XTMP
047F 27 11          BEQ   LKUNM5    START WITH INX IF '-' FOUND
0481 5F             CLR B
0482                LKUNM4 EQU  *
                    * SHIFT NUMBER LEFT 4
0482 81 04          CMP A #4        OVERFLOW?
0484 22 23          BHI   NUMOVR    YEP-->ERROR MESSAGE
0486 58             ASL B
0487 49             ROL A
0488 58             ASL B
0489 49             ROL A
048A 58             ASL B
048B 49             ROL A
048C 58             ASL B
048D 49             ROL A
048E EB 00          ADD B 0,X       ADD IN NEW DIGIT
0490 C0 30          SUB B #'0
0492                LKUNM5 EQU  *
0492 08             INX
0493 6D 00          TST   0,X
0495 26 EB          BNE   LKUNM4    AND LOOP AROUND UNTIL DONE
                    *
0497                LKUNM6 EQU  *
0497 97 28          STA A XTMP2     STORE THE RESULT
0499 D7 29          STA B XTMP2+1
049B DE 28          LDX   XTMP2
049D 96 26          LDA A XTMP      NEGATIVE?
049F 26 03          BNE   LKUNM7
                    * YEP, DO A BCD INVERT
04A1 BD 01 45       JSR   NUMINV
04A4                LKUNM7 EQU  *
04A4 BD 01 48       JSR   NUMFRM    CONVERT TO FORM
04A7 20 B0          BRA   FNDNUM    STORE IN FORM AND RETURN
                    *
04A9                NUMOVR EQU  *
04A9 DE 32          LDX   NAMPTR    RESET SPCPTR  (ASSUMING NO BUILTIN NUMBERS!!)
04AB DF 2C          STX   SPCPTR
04AD CE 05 59       LDX   #NNOVMS   'NUMERIC OVERFLOW'
04B0 BD 01 54       JSR   ERRBRK    BREAK FOR NEW NUMBER
04B3 20 C0          BRA   FND1      AND CLEANUP
                    ***
04B5                IXLIST EQU  *
                    * PTR TO LIST OF CHARS IN X-REG
                    * CHAR IN A-REG, RETURNS INDEX IN B-REG
                    * B < 0 IF NOT FOUND (CC'S SET ON EXIT)
04B5 DF 26          STX   XTMP      SAVE X-REG FOR INDEX CALC
04B7 BD 10          BSR   INLIST    ON LIST?
04B9 24 04          BCC   IXL2
04BB C6 FF          LDA B #-1       NOPE
04BD 20 06          BRA   IXLRT
04BF DF 28          IXL2  STX   XTMP2     CALC INDEX
04C1 D6 29          LDA B XTMP2+1
04C3 D0 27          SUB B XTMP+1
04C5                IXLRT EQU   *
04C5 DE 26          LDX   XTMP
04C7 5D             TST B
04C8 39             RTS
                    ***
04C9                INLIST EQU  *
                    * PTR TO CHAR LIST IN X-REG, CHAR IN A-REG
                    * RETURNS WITH CARRY SET IF NOT FOUND, CLEAR OTHERWISE
                    * X-REG CLOBBERED (LEFT POINTING AT MATCH)
04C9 A1 00          CMP A 0,X       MATCH CHAR IN LIST?
04CB 27 06          BEQ   INLRT
04CD 08             INX             NOPE, TRY THE NEXT
04CE 6D 00          TST   0,X       END OF LIST?
04D0 26 F7          BNE   INLIST
04D2 0D             SEC             YUP, SET CARRY
04D3                INLRT EQU   *
04D3 39             RTS             RETURN WITH CARRY SET OR CLEAR
                    ***
04D4                GETSYM EQU  *
                    * READ NEXT SYMBOL, ADD TO SYMLST, RETURN PTR IN X-REG
                    * SPCPTR,FREPTR,SYMLST,ENDPTR MUST BE INITIALIZED
                    * USES GETC SUBROUTINE TO GET CHARS
                    * PEEKC SHOULD BE ZERO INITIALLY
                    * TYPE OF LEXEME RETURNED IN B-REG...
                    * 0 = EOF
                    * 1 = QUOTED STRING
                    * 2 = BREAK CHAR (LIKE ',' OR '(' )
                    * 3 = ALPHANUMERIC
                    ** 4 = NON-ALPHANUMERIC ** REMOVED THIS CATEGORY 11/24/78 **
                    * <SP> AND ALL CTL CHARS (E.G. <HT> AND <FF>)
                    *  ACT AS SEPARATORS, EXCEPT <EOI> IS BREAK
04D4 5F             CLR B
04D5 96 F1          LDA A RSTFLG    BEING RESET?
04D7 26 38          BNE   RETEI2    YEP, RETURN WITH EOIATM
04D9 DE 2C          LDX   SPCPTR    GET START AD FOR CHARS OF SYM
04DB DF 38          STX   SYMPTR
04DD 96 F0          LDA A PEEKC     DID WE PEEK?
04DF 26 03          BNE   GS2       YUP, SKIP THE GETC
04E1                GSNXTC EQU  *
04E1 BD 01 42       JSR   GETC
04E4                GS2   EQU   *
04E4 CE 05 51       LDX   #BRKLST   IS IT A BREAK CHAR?
04E7 BD E0          BSR   INLIST
04E9 24 09          BCC   GSBRK
04EB 81 20          CMP A #' '      NOPE, IS IT A <SP> OR CTL CHAR?
04ED 22 27          BHI   GSNSEP
04EF 5D             TST B           YEP, HAVE WE GOT A SYMBOL?
04F0 26 12          BNE   ENDSYM
04F2 20 ED          BRA   GSNXTC    NOPE, GO GET NEXT CHAR
04F4                GSBRK EQU   *
04F4 5D             TST B           HAVE WE GOT A SYMBOL?
04F5 26 0D          BNE   ENDSYM
04F7 81 04          CMP A #EOI      <EOI> TYPED?
04F9 27 13          BEQ   RETEOI    YES, RETURN WITH B=0, NO PEEKC, FORM=EOIATM
04FB BD 42          BSR   COPYC     NOPE, START A NEW ONE
04FD 81 22          CMP A #'"       QUOTED STRING?
04FF 27 24          BEQ   GSQSTR    YES, GO COPY REST OF STRING
0501 C6 02          LDA B #2        NOPE, SET BREAK-CHAR CODE
0503                ENDNPK EQU  *
0503 4F             CLR A           END SYM WITH NO PEEKC
0504                ENDSYM EQU  *
0504 97 F0          STA A PEEKC     SAVE PEEKC
0506 4F             CLR A           FINISH UP SYMBOL
0507 BD 36          BSR   COPYC
0509 DE 38          LDX   SYMPTR    LOOKUP ON SYMLST
050B 7E 03 FC       JMP   LOOKUP    AND RETURN DIRECTLY FROM LOOKUP
                    *
050E                RETEOI EQU  *
050E 7F 00 F0       CLR   PEEKC
0511                RETEI2 EQU  *
0511 DE 64          LDX   EOIATM    RETURN WITH EOI ATOM
0513 DF 34          STX   FORM
0515 39             RTS
                    *
0516                GSNSEP EQU  *
                    ** NOT A SEPARATOR OR BREAK, CHECK IF OTHER SPECIAL
                    * LDX #SPCLST
                    * BSR INLIST
                    * BCC GSSPCL
                    * NOT SPECIAL, ASSUME "ALPHANUMERIC"
0516 5D             TST B           FIRST CHAR?
0517 26 06          BNE   GSANCK
0519 C6 03          LDA B #3        YES, SET ALPHA-NUMERIC CODE
051B                GSANST EQU  *
051B BD 22          BSR   COPYC
051D 20 C2          BRA   GSNXTC    LOOP FOR NEXT CHAR
051F                GSANCK EQU  *
051F C1 03          CMP B #3        NOW ALPHA-NUMERIC?
0521 27 F8          BEQ   GSANST    YEP, STORE CHAR AND GET NEXT
0523 20 DF          BRA   ENDSYM    NOPE, GO FINISH UP PREV SYM
                    *GSSPCL EQU *
                    * TSTB   FIRST SPECIAL CHAR?
                    * BNE GSSPCK
                    * LDAB #4  YES, SET SPECIAL CHAR SYM CODE
                    * BRA GSANST   AND GO STORE CHAR
                    *GSSPCK EQU *
                    * CMPB #4  NOW STORING SPECIAL CHARS?
                    * BEQ GSANST
                    * BRA ENDSYM  NOPE, GO FINISH UP PREVIOUS SYMBOL
0525                GSQSTR EQU  *         QUOTED STRING...
0525 C6 01          LDA B #1        QUOTED STRING TYPE SYMBOL
0527                GSQ2  EQU   *
0527 BD 01 42       JSR   GETC      GET NEXT CHAR
052A BD 13          BSR   COPYC     AND SAVE IT
052C 81 04          CMP A #EOI      END OF INPUT?
052E 27 04          BEQ   ENDSYM    YEP, AND SAVE EOI IN PEEKC
0530 81 22          CMP A #'"       CLOSE QUOTE?
0532 26 F3          BNE   GSQ2
0534 BD 01 42       JSR   GETC      YEP, CHECK FOR DOUBLE ""
0537 81 22          CMP A #'"
0539 26 C9          BNE   ENDSYM
053B BD 02          BSR   COPYC     YEP, SAVE IT
053D 20 E8          BRA   GSQ2      AND KEEP GOING
053F                COPYC EQU   *
                    *  SAVE CHAR IN A-REG AT LOC PT'D BY SPCPTR
                    *  CHECK AGAINST ENDPTR FOR SAFETY
053F DE 2C          LDX   SPCPTR
0541 9C 2E          CPX   ENDPTR
0543 2A 06          BPL   CPCBAD
0545 A7 00          STA A 0,X
0547 08             INX
0548 DF 2C          STX   SPCPTR
054A 39             RTS
054B                CPCBAD EQU  *
054B CE 05 6A       LDX   #NOGCMG   'NO MORE FREE SPACE'
054E 7E 01 36       JMP   ERREX
                    *
                    *
7118                PSTRNG EQU  $7118
711E                PCRLF  EQU  $711E
                    * TABLES, ETC.
0551 22             BRKLST FCC  '"()[]'
                    *   (USED TO INCLUDE "," AND "()" IN ABOVE LIST)
0556 27                    FCB  ''
0557 04                    FCB  EOI
0558 00                    FCB  0
                    *SPCLST FCC '!#$%&=++;:^'<>/?|\'
                    * FCB 0
                    *
0559 4E             NNOVMS FCC  'NUMERIC OVERFLOW'
0569 04                    FCB  4
056A 4E             NOGCMG FCC  'NO MORE FREE SPACE'
057C 04                    FCB  4
057D                REGADR EQU  *
                           END
```

```
NO ERROR(S) DETECTED

 SYMBOL TABLE:

ALP    0044   ATMINI 0139   BEGADR 057D   BEGPTR 0020   BRKLST 0551
CAR    0000   CDR    0002   CELPTR 003A   CMPTM2 0042   CMPTMP 0040
CONATM 0060   COPYC  053F   CPCBAD 054B   CUREVL 006C   DADPTR 0070

DOTATM 0050   ENDMEM 00FC   ENDNPK 0503   ENDPTR 002E   ENDSYM 0504
EOI    0004   EOIATM 0064   EOL    0434   ERRBRK 0154   ERREX  0136
EVAL   0100   EVLATM 011B   FLP    0048   FND1   0475   FND2   0477
FNDNUM 0459   FORM   0034   FOUND  046D   FRECEL 010F   FREPTR 002A
FRMNUM 014B   GCFREE 0074   GCOL   015D   GCTEMP 0072   GETC   0142
GETCAR 0163   GETCEL 010C   GETSYM 04D4   GFNXTL 0127   GNXTL  0124
GS2    04E4   GSANCK 051F   GSANST 051B   GSBRK  04F4   GSNSEP 0516
GSNXTC 04E1   GSQ2   0527   GSQSTR 0525   INLIST 04C9   INLRT  04D3
ISATOM 013F   ISDTPR 013C   ISEQ   03F4   ISVAR  0166   IXL2   04BF
IXLIST 04B5   IXLRT  04C5   LAMATM 0056   LBRKAT 0068   LKU1   0418
LKU2   041C   LKU3   042E   LKU4   045B   LKU5   0469   LKUNM2 0410
LKUNM3 0478   LKUNM4 0482   LKUNM5 0492   LKUNM6 0497   LKUNM7 04A4
LKUNUM 040B   LOOKUP 03FC   LPARAT 004C   LSTAD2 0130   LSTADD 012D
LSTEN0 0115   LSTEND 0133   LSTINI 012A   LSTPTR 0036   NAMPTR 0032
NILATM 004A   NLAMAT 0058   NLP    0046   NNOVMS 0559   NOGCMG 056A
NSUBAT 005C   NUMFRM 0148   NUMINV 0145   NUMOVR 04A9   OBEGAD 03D0
PCRLF  711E   PEEKC  00F0   POPX   0115   PRINR  0169   PRINT  0106
PROPSH 0160   PSTRNG 7118   PUSHX  0112   PUTC   0151   PUTSYM 014E
QUOATM 0054   RBRKAT 006A   READ   0103   RETEI2 0511   RETEOI 050E
RPARAT 004E   RSETAT 0062   RSTFLG 00F1   SETATM 011E   SONPTR 006E
SPCPTR 002C   SPSAVE 003C   SQUTAT 0052   STE2   03D6   STKFRG 015A
STKPTR 003E   STP1   0026   STP2   0028   STR1   0022   STR2   0024
STREQ  03D0   SUBLS2 0066   SUBLS3 0076   SUBRAT 005A   SYMLST 0030
SYMPTR 0038   TATOM  005E   TOPX   0118   TYPSWT 0109   UNEQ   03F8
XTMP   0026   XTMP2  0028   ZERO   00FE
```

```
                        OPT    NOG
05A0            OBEGAD  EQU    $5A0        $582 ENDS LOOKUP
0000            CAR     EQU    0
0002            CDR     EQU    2
0004            EOI     EQU    4
                * DIRECT PAGE STORAGE CELLS
0020                    ORG    $20
0020 06 8C      BEGPTR  FDB    BEGADR      THIS SHOULD BE IN EACH MODULE
0022            STR1    RMB    2
0024            STR2    RMB    2
0026            STP1    RMB    2
0028            STP2    RMB    2
0026            XTMP    EQU    STP1
0028            XTMP2   EQU    STP2
002A            FREPTR  RMB    2
002C            SPCPTR  RMB    2
002E            ENDPTR  RMB    2
0030            SYMLST  RMB    2
0032            NAMPTR  RMB    2
0034            FORM    RMB    2
0036            LSTPTR  RMB    2
0038            SYMPTR  RMB    2
003A            CELPTR  RMB    2
003C            SPSAVE  RMB    2
003E            STKPTR  RMB    2
0040            CMPTMP  RMB    2
0042            CMPTM2  RMB    2
0044            ALP     RMB    2
0046            NLP     RMB    2
0048            FLP     RMB    2
004A            NILATM  RMB    2
004C            LPARAT  RMB    2
004E            RPARAT  RMB    2
0050            DOTATM  RMB    2
0052            SQUTAT  RMB    2
0054            QUOATM  RMB    2
0056            LAMATM  RMB    2
0058            NLAMAT  RMB    2
005A            SUBRAT  RMB    2
005C            NSUBAT  RMB    2
005E            TATOM   RMB    2
0060            CONATM  RMB    2
0062            RSETAT  RMB    2
0064            EOTATM  RMB    2
0066            SUBLS2  RMB    2
0068            LBRKAT  RMB    2
006A            RBRKAT  RMB    2
006C            CUREVL  RMB    2
006E            SONPTR  RMB    2
0070            DADPTR  RMB    2
0072            GCTEMP  RMB    2
0074            GCFREE  RMB    2          NUMBER OF FREE CELLS AFTER LAST GCOL
0076            SUBLS3  RMB    2          INITIALIZED IN SUBRS3
                * SINGLE CHAR SLOTS ...
00F0                    ORG    $F0
00F0            PEEKC   RMB    1
00F1            RSTFLG  RMB    1
                * GLOBAL CONSTANTS
00FC                    ORG    $FC
00FC            ENDMEM  RMB    2          INITIALIZED IN LISP
                * ZERO WORD
00FE 00 00      ZERO    FDB    0
                * GLOBAL VECTORS
0100                    ORG    $100
0100            EVAL    RMB    3
0103            READ    RMB    3
0106 7E 06 2E           JMP    PRINT
0109 7E 06 89           JMP    UNDEF       ** USED TO BE TYPSWT **
010C            GETCEL  RMB    3
010F            FRECEL  RMB    3
0112            PUSHX   RMB    3
0115            POPX    RMB    3
0118            TOPX    RMB    3
011B            EVLATM  RMB    3
011E            SETATM  RMB    3
0121            LOOKUP  RMB    3
0124            GNXTL   RMB    3
0127            GFNXTL  RMB    3
012A            LSTINI  RMB    3
012D            LSTADD  RMB    3
0130            LSTAD2  RMB    3
0133            LSTEND  RMB    3
0115            LSTENQ  EQU    POPX
0136            ERREX   RMB    3
0139            ATMINI  RMB    3
013C            ISDTPR  RMB    3
013F            ISATOM  RMB    3
0142            GETC    RMB    3
0145            NUMINV  RMB    3
0148 7E 06 0B           JMP    NUMFRM
014B 7E 05 E9           JMP    FRMNUM
014E 7E 05 A0           JMP    PUTSYM
0151            PUTC    RMB    3
0154            ERRBRK  RMB    3
0157            GETSYM  RMB    3
015A            STKFRG  RMB    3
015D            GCOL    RMB    3
0160            PROPSH  RMB    3
0163            GETCAR  RMB    3
0166            ISVAR   RMB    3
0169 7E 06 3F           JMP    PRINRO
016C            PROPOP  RMB    3
                *
7118            PSTRNG  EQU    $7118
711E            PCRLF   EQU    $711E
                *
05A0                    ORG    OBEGAD
                *
05A0            PUTSYM  EQU    *
                * PUTSYM EXPECTS ATOM PTR IN X-REG
                * USES PUTC TO OUTPUT CHARS
05A0 DF 34              STX    FORM
05A2 2B 17              BMI    PUTNUM      NUMBER--> PUTNUM
05A4 26 02              BNE    PSY2
05A6 DE 4A              LDX    NILATM
05A8            PSY2    EQU    *
05A8 09                 DEX                CONVERT ATOM TO POINTER
05A9 EE 00              LDX    CAR,X
05AB BD 03              BSR    PUTSTR      OUTPUT THE NAME
05AD DE 34              LDX    FORM
05AF 39                 RTS
                *
05B0            PUTSTR  EQU    *
                * PUTSTR EXPECTS ADDR OF ZERO-TERMINATED CHAR STRING
                * IN X-REG  (WHICH IS CLOBBERED!)
                * USES PUTC TO OUTPUT INDIVIDUAL CHARS.
05B0            PSNXT   EQU    *
05B0 A6 00              LDA A  0,X
05B2 27 06              BEQ    PSRT
05B4 BD 01 51           JSR    PUTC
05B7 08                 INX
05B8 20 F6              BRA    PSNXT
05BA            PSRT    EQU    *
05BA 39                 RTS
                *
05BB            PUTNUM  EQU    *
                * PUTNUM EXPECTS FORM IN X-REG
                * USES PUTC TO OUTPUT IT
05BB BD 2C              BSR    FRMNUM      CONVERT FORM TO NUMBER
05BD 2A 08              BPL    PNM2
05BF 86 2D              LDA A  #'-         NUMBER IS NEGATIVE
05C1 BD 01 51           JSR    PUTC
05C4 BD 01 45           JSR    NUMINV      CONVERT TO ITS ABSOLUTE VALUE
05C7            PNM2    EQU    *
05C7 DF 26              STX    XTMP        LOAD NUM INTO A AND B REGS
05C9 D6 26              LDA B  XTMP
05CB 96 27              LDA A  XTMP+1
05CD DE 34              LDX    FORM        RESTORE X-REG FOR RETURN
05CF            PNUMR   EQU    *
05CF 5D                 TST B              ANY HIGHER ORDER DIGITS?
05D0 26 04              BNE    PNMR2
05D2 81 09              CMP A  #$09
05D4 23 0E              BLS    PNMR3
05D6            PNMR2   EQU    *
05D6 36                 PSH A              YEP, SAVE A
05D7 54                 LSR B              SHIFT RIGHT 4
05D8 46                 ROR A
05D9 54                 LSR B
05DA 46                 ROR A
05DB 54                 LSR B
05DC 46                 ROR A
05DD 54                 LSR B
05DE 46                 ROR A
05DF BD EE              BSR    PNUMR       RECURSE TO PRINT OTHER DIGITS
05E1 32                 PUL A              RESTORE A
05E2 84 0F              AND A  #$0F        AND OUT HIGHER DIGITS
05E4            PNMR3   EQU    *
05E4 8A 30              ORA A  #'0         FORM ASCII CODE
05E6 7E 01 51           JMP    PUTC        PUT DIGIT AND RETURN
                *
05E9            FRMNUM  EQU    *
                * EXPECTS FORM IN X-REG
                * RETURNS 4-DIGIT BCD NUMBER IN X-REG
                * USING 10'S COMPLEMENT
                *  N-BIT SET IF NUMBER IS NEGATIVE
                *  Z-BIT SET IF NUM IS ZERO
05E9 DF 34              STX    FORM        STORE THE FORM
05EB 96 FC              LDA A  ENDMEM      SET UP HIGH BYTE OF FRMNUM TABLE ADDRESS
05ED 97 26              STA A  XTMP
05EF 96 35              LDA A  FORM+1      CONVERT LOW BYTE
05F1 16                 TAB
05F2 0D                 SEC
05F3 46                 ROR A
05F4 44                 LSR A
05F5 BD 30              BSR    FNX
05F7 56                 ROR B              SET UP LOW BIT
05F8 49                 ROL A
05F9 97 29              STA A  XTMP2+1     AND STORE RESULT
05FB 96 34              LDA A  FORM        CONVERT HIGH BYTE
05FD 44                 LSR A
05FE BD 27              BSR    FNX
0600 49                 ROL A              SET LOW BIT
0601 97 28              STA A  XTMP2
0603 DE 28              LDX    XTMP2       LOAD RESULT
0605 27 03              BEQ    FNMRET      SET UP Z AND N BIT
0607 BB 30              ADD A  #$30
0609 4D                 TST A
060A            FNMRET  EQU    *
060A 39                 RTS
                *
0608            NUMFRM  EQU    *
                * EXPECTS BCD NUMBER IN X-REG (10'S COMPLEMENT)
                * CONVERTS TO FORM, WITH HIGH BIT SET, AND BIT 1=0 FOR GCOL
060B DF 28              STX    XTMP2       STORE THE NUMBER
060D 96 FC              LDA A  ENDMEM      SET UP HIGH BYTE OF NUMFRM TABLE ADDR
060F 97 26              STA A  XTMP
0611 96 29              LDA A  XTMP2+1     CONVERT LOW BYTE
0613 16                 TAB
0614 BD 0F              BSR    NFX
0616 48                 ASL A
0617 56                 ROR B              SET UP LOW BIT
0618 49                 ROL A
0619 97 35              STA A  FORM+1      STORE THE RESULT
061B 96 28              LDA A  XTMP2       CONVERT THE HIGH BYTE
061D BD 06              BSR    NFX
061F 49                 ROL A              SET UP THE LOW BIT
0620 97 34              STA A  FORM
0622 DE 34              LDX    FORM        RETURN THE RESULT
0624 39                 RTS
                *
0625            NFX     EQU    *
0625 0D                 SEC
0626 46                 ROR A              LOW BIT==> C-BIT, 1==> HIGH BIT
0627            FNX     EQU    *
0627 97 27              STA A  XTMP+1      STORE LOW BYTE OF FRMNUM/NUMFRM TABLE ADDR
0629 DE 26              LDX    XTMP
062B A6 00              LDA A  0,X         LOAD CONVERTED VALUE
062D 39                 RTS                AND RETURN
                *
                * "PRINT" TYPES OUT FORM, FOLLOWED BY CR,LF
                * USES RECURSIVE ROUTINE "PRINR" TO DO THE WORK
                * FORM RETURNED IN X-REG AND "FORM"
062E            PRINT   EQU    *
062E 96 F1              LDA A  RSTFLG      IN A RESET?
0630 27 01              BEQ    PRINT1
0632            PRINRT  EQU    *
0632 39                 RTS                YEP, RETURN IMMEDIATELY
0633            PRINT1  EQU    *
0633 BD 0E              BSR    PRINR       CALL RECURSIVE PRINT ROUTINE
0635 DF 34              STX    FORM        SAVE IN FORM
0637 86 0D              LDA A  #$D         CR,LF
0639 BD 4B              BSR    PPUTC
063B 86 0A              LDA A  #$A
063D 20 47              BRA    PPUTC
                *
                * TYPE OUT FORM, WITHOUT CR,LF
063F            PRINRO  EQU    *
063F 96 F1              LDA A  RSTFLG      IN A RESET?
0641 26 EF              BNE    PRINRT      YEP, SIMPLY RETURN
0643            PRINR   EQU    *
                * PRINT TAKES ADDR OF OBJECT IN X-REG
                * ASSUMES EITHER DTPR, NIL, ATOM, OR NUMBER
                * CALLS PUTSYM AND PUTC
                * ASSUMES "NIL" ATOM STORED AT "NILATM"
                * USES PUSHX SUBR TO STACK X-REG
                * ALSO POPX AND TOPX TO REFERENCE  STACK
0643 BD 01 3F           JSR    ISATOM      SWITCH ON TYPE OF FORM
0646 25 03              BCS    DTPRT
0648 7E 05 A0           JMP    PUTSYM      NIL, ATOM, OR NUMBER
                *
064B            DTPRT   EQU    *
064B BD 01 60           JSR    PROPSH      SAVE X-REG FOR RESTORE LATER
064E BD 01 5A           JSR    STKFRG      FRAGMENT THE STACK
                * GOT A DTPR, PRINT "("
0651 86 28              LDA A  #'(
0653 BD 31              BSR    PPUTC
0655 EE 00              LDX    CAR,X       PRINT THE CAR
0657 BD EA              BSR    PRINR       RECURSIVELY!
0659 BD 01 18           JSR    TOPX        RESTORE X-REG
065C            PRT2    EQU    *
065C EE 02              LDX    CDR,X       AND FOLLOW LIST
065E BD 01 3C           JSR    ISDTPR      CHECK TYPE OF FORM
0661 27 1A              BEQ    PRT4        NIL ==> END OF LIST
0663 25 0E              BCS    PRT35       A NUMBER OR AN ATOM(?!)
0665            PRT3    EQU    *
0665 BD 1D              BSR    PPUTSP      PRINT A SPACE
```

```
0667 BD 01 12           JSR    PUSHX     SAVE X-REG
066A EE 00              LDX    CAR,X     PRINT THE CAR
066C BD D5              BSR    PRINR     RECURSIVELY!
066E BD 01 15           JSR    POPX      RESTORE X-REG
0671 20 E9              BRA    PRT2
0673            PRT35   EQU    *
                * ATOM OR NUMBER AT END OF LIST
                * USE "." SYNTAX
0673 BD 0F              BSR    PPUTSP    PUT OUT " . "
0675 B6 2E              LDA A  #'.
0677 BD 0D              BSR    PPUTC
0679 BD 09              BSR    PPUTSP
067B BD C6              BSR    PRINR     RECURSE TO PRINT OBJECT
067D            PRT4    EQU    *
067D B6 29              LDA A  #')       END THE LIST
067F BD 05              BSR    PPUTC
0681 7E 01 6C           JMP    PROPOP    RESTORE X-REG AND RETURN
                *
0684            PPUTSP  EQU    *
                * PUT OUT A SPACE
0684 B6 20              LDA A  #' 
0686            PPUTC   EQU    *         USED TO AVOID LONG JSR
0686 7E 01 51           JMP    PUTC
                *
0689            UNDEF   EQU    *
0689 3F                 SWI               STOP, CALLED TYPSWT FROM SOMEWHERE
068A 20 FD              BRA    UNDEF
                *
068C            BEGADR  EQU    *
                        END
```

NO ERROR(S) DETECTED

SYMBOL TABLE:

```
ALP    0044   ATMINI 0139   BEGADR 068C   BEGPTR 0020   CAR    0000
CDR    0002   CELPTR 003A   CMPTM2 0042   CMPTMP 0040   CONATM 0060
CUREVL 008C   DADPTR 0070   DOTATM 0050   DTPRT  064B   ENDMEM 00FC
ENDPTR 002E   EOI    0004   EOIATM 0064   ERRBRK 0154   ERREX  0136
EVAL   0100   EVLATM 011B   FLP    004B   FNMRET 060A   FNX    0627
FORM   0034   FRECEL 010F   FREPTR 002A   FRMNUM 05E9   GCFREE 0074
GCOL   015D   GCTEMP 0072   GETC   0142   GETCAR 0163   GETCEL 010C
GETSYM 0157   GFNXTL 0127   GNXTL  0124   ISATOM 013F   ISDTPR 013C
ISVAR  0166   LAMATM 0056   LBRKAT 006B   LOOKUP 0121   LPARAT 004C
LSTAD2 0130   LSTADD 012D   LSTEND 0133   LSTENO 011A   LSTINI 012A
LSTPTR 0036   NAMPTR 0032   NEX    0625   NILATM 004A   NLAMAT 0058
NLP    0046   NSUBAT 005C   NUMFRM 060B   NUMINV 0145   OBEGAD 05A0
PCRLF  711E   PEEKC  00F0   PNM2   05C7   PNMR2  05D6   PNMR3  05E4
PNUMR  05CF   POPX   0115   PPUTC  0686   PPUTSP 0684   PRINR  0643
PRINR0 063F   PRINRT 0632   PRINT  062E   PRINT1 0633   PROPOP 016C
PROPSH 0160   PRT2   065C   PRT3   0665   PRT35  0673   PRT4   067D
PSNXT  05B0   PSRT   05BA   PSTRNG 7118   PSY2   05A8   PUSHX  0112
PUTC   0151   PUTNUM 05BB   PUTSTR 05B0   PUTSYM 05A0   QUOATM 0054
RBRKAT 006A   READ   0103   RPARAT 004E   RSETAT 0062   RSTFLG 00F1
SETATM 011E   SONPTR 006E   SPCPTR 002C   SPSAVE 003C   SQUTAT 0052
STKFRG 015A   STKPTR 003E   STP1   0026   STP2   0028   STR1   0022
STR2   0024   SUBLS2 0066   SUBLS3 0076   SUBRAT 005A   SYMLST 0030
SYMPTR 0038   TATOM  005E   TOPX   0118   UNDEF  0689   XTMP   0026
XTMP2  0028   ZERO   00FE
```

```
                OPT    NOG
06A0            OBEGAD EQU    $6A0      PRINT ENDS AT $6BB
0032            STKMIN EQU    50        ALWAYS AT LEAST 50 BYTES OF STACK AFTER STKFRG
0014            MINFRE EQU    20        MIN. NO. OF FREE CELLS REQUIRED BY GETCEL
0002            MRKBIT EQU    2         BIT USED BY GCOL
                *
0000            CAR    EQU    0
0002            CDR    EQU    2
0004            EOI    EQU    4
                * DIRECT PAGE STORAGE CELLS
0020                   ORG    $20
0020 09 0A      BEGPTR FDB    BEGADR    THIS SHOULD BE IN EACH MODULE
0022            STR1   RMB    2
0024            STR2   RMB    2
0026            STP1   RMB    2
0028            STP2   RMB    2
0026            XTMP   EQU    STP1
0028            XTMP2  EQU    STP2
002A            FREPTR RMB    2
002C            SPCPTR RMB    2
002E            ENDPTR RMB    2
0030            SYMLST RMB    2
0032            NAMPTR RMB    2
0034            FORM   RMB    2
0036            LSTPTR RMB    2
0038            SYMPTR RMB    2
003A            CELPTR RMB    2
003C            SPSAVE RMB    2
003E            STKPTR RMB    2
0040            CMPTMP RMB    2
0042            CMPTM2 RMB    2
0044            ALP    RMB    2
0046            NLP    RMB    2
0048            FLP    RMB    2
004A            NILATM RMB    2
004C            LPARAT RMB    2
004E            RPARAT RMB    2
0050            DOTATM RMB    2
0052            SQUTAT RMB    2
0054            QUOATM RMB    2
0056            LAMATM RMB    2
0058            NLAMAT RMB    2
005A            SUBRAT RMB    2
005C            NSUBAT RMB    2
005E            TATOM  RMB    2
0060            CONATM RMB    2
0062            RSETAT RMB    2
0064            EOIATM RMB    2
0066            SUBLS2 RMB    2
0068            LBRKAT RMB    2
006A            RBRKAT RMB    2
006C            CUREVL RMB    2
006E            SONPTR RMB    2
0070            DADPTR RMB    2
0072            GCTEMP RMB    2
0074            GCFREE RMB    2         NUMBER OF FREE CELLS AFTER LAST GCOL
0076            SUBLS3 RMB    2         INITIALIZED IN SUBRS3
                * SINGLE CHAR SLOTS ...
00F0                   ORG    $F0
00F0            PEEKC  RMB    1
00F1            RSTFLG RMB    1
                * GLOBAL CONSTANTS
00FC                   ORG    $FC
                * END OF MEMORY, MINUS 196 FOR NUMFRM, FRMNUM TABLES
00FC            ENDMEM RMB    2         INITIALIZED IN LISP
                * ZERO WORD
00FE 00 00      ZERO   FDB    0
                * GLOBAL VECTORS
0100                   ORG    $100
```

```
0100            EVAL   RMB    3
0103            READ   RMB    3
0106            PRINT  RMB    3
0109                   RMB    3
010C 7E 07 1D          JMP    GETCEL
010F 7E 07 A9          JMP    FRECEL
0112 7E 07 61          JMP    PUSHX
0115 7E 07 7A          JMP    POPX
0118 7E 07 93          JMP    TOPX
011B            EVLATM RMB    3
011E            SETATM RMB    3
0121            LOOKUP RMB    3
0124            GNXTL  RMB    3
0127            GFNXTL RMB    3
012A            LSTINI RMB    3
012D            LSTADD RMB    3
0130            LSTAD2 RMB    3
0133            LSTEND RMB    3
077A            LSTENO EQU    POPX
0136            ERREX  RMB    3
0139            ATMINI RMB    3
013C            ISDTPR RMB    3
013F            ISATOM RMB    3
0142            GETC   RMB    3
0145            NUMINV RMB    3
0148            NUMFRM RMB    3
014B            FRMNUM RMB    3
014E            PUTSYM RMB    3
0151            PUTC   RMB    3
0154            ERRBRK RMB    3
0157            GETSYM RMB    3
015A 7E 06 A0          JMP    STKFRG
015D 7E 07 B4          JMP    GCOL
0160 7E 07 5E          JMP    PROPSH
0163            GETCAR RMB    3
0166 7E 07 9A          JMP    ISVAR
0169            PRINR  RMB    3
016C 7E 07 7A          JMP    PROPOP
                *
06A0                   ORG    OBEGAD
                *
06A0            STKFRG EQU    *
                * JSR STKFRG  WILL MAKE ROOM FOR MORE STACK, POSSIBLY
                *   FRAGMENTING THE STACK.
                *
                * (XREG+4) CONTAINS OLD SP OF FORMER FRAGMENT
                * (XREG+2) CONTAINS ADDR OF "UNRAVL" SUBROUTINE
                * (XREG+0) CONTAINS RETURN ADDRESS FOR THIS SUBROUTINE
                *
06A0 9F 28              STS    XTMP2     CHECK LOW BYTE OF SP FOR MIN. STACKLIM
06A2 96 29              LDA A  XTMP2+1
06A4 81 32              CMP A  #STKMIN
06A6 23 01              BLS    STKF2
06A8 39                 RTS               ENOUGH ROOM, JUST RETURN
                *
06A9            STKF2  EQU    *
06A9 DF 28              STX    XTMP2     SAVE X-REG FOR RESTORE LATER
06AB 96 2E              LDA A  ENDPTR
06AD D6 2F              LDA B  ENDPTR+1  ADJUST ENDPTR TO 256-BYTE BOUNDARY
06AF 27 04              BEQ    STKF4
06B1 BD 47              BSR    BLKFRE    FREE UP BLOCK DOWN TO 256-BYTE BOUNDARY
06B3 DF 2E              STX    ENDPTR    UPDATE ENDPTR
06B5            STKF4  EQU    *
06B5 4A                 DEC A            ALLOCATE NEW 256-BYTE STACK FRAGMENT
06B6 91 2C              CMP A  SPCPTR    BUMPING INTO NAME SPACE?
06B8 23 20              BLS    STKFUL
06BA 97 2E              STA A  ENDPTR    NOPE, UPDATE ENDPTR
06BC 97 26              STA A  XTMP
06BE B6 FA              LDA A  #250
```

```
06C0 97 27              STA A  XTMP+1    SET UP NEW STACK POINTER
06C2 DE 26              LDX    XTMP      INITIALIZE NEW STACK FRAGMENT
06C4 32                 PUL A            RETURN ADDR
06C5 A7 00              STA A  0,X
06C7 32                 PUL A
06C8 A7 01              STA A  1,X
06CA AF 04              STS    4,X       OLD SP
06CC B6 06 E0           LDA A  UNRAVA    ADDR OF UNRAVL ROUTINE
06CF A7 02              STA A  2,X
06D1 B6 06 E1           LDA A  UNRAVA+1
06D4 A7 03              STA A  3,X
06D6 35                 TXS              SET NEW SP
06D7 DE 28              LDX    XTMP2     RESTORE X-REG
06D9 39                 RTS              AND RETURN
                *
06DA            STKFUL EQU    *
06DA CE 08 F8           LDX    #STKEMG   'NO MORE ROOM FOR STACK FRAGMENT'
06DD 7E 01 36           JMP    ERREX     BYEBYE!
                *
06E0 06 E2      UNRAVA FDB    UNRAVL    ADDR OF UNRAVL ROUTINE
                *
06E2            UNRAVL EQU    *
                *  UNRAVEL STACK FRAGMENT.
                *  IF END OF FRAGMENT STILL EQUALS ENDPTR, SIMPLY RESET ENDPTR,
                *   OTHERWISE ADD SPACE TO FREE LIST
06E2 DF 28              STX    XTMP2     SAVE X-REG
06E4 30                 TSX
06E5 AE 00              LDS    0,X       RESTORE OLD SP
06E7 DF 26              STX    XTMP
06E9 96 26              LDA A  XTMP      END OF FRAG = ENDPTR?
06EB 91 2E              CMP A  ENDPTR
06ED 26 06              BNE    UNRFRE
06EF 4C                 INC A            YEP, JUST UPDATE ENDPTR
06F0 97 2E              STA A  ENDPTR    (BY ADDING 256)
06F2            UNRVRT EQU    *
06F2 DE 28              LDX    XTMP2     RESTORE X-REG
06F4 39                 RTS              AND RETURN
                *
06F5            UNRFRE EQU    *
                * FREE ALL CELLS WITHIN FRAGMENT
06F5 5F                 CLR B
06F6 BD 02              BSR    BLKFRE
06F8 20 F8              BRA    UNRVRT    RETURN
                *
06FA            BLKFRE EQU    *
                * FREE BLOCK, A&B-REG SPECIFY TOP OF BLOCK
                *  AT RETURN, X-REG, A&B-REG, AND XTMP=A-REG POINT TO BOTTOM OF BLOCK
                *  XTMP+1 IS UNSPECIFIED ON ENTRY AND RETURN
06FA 97 26              STA A  XTMP
06FC C0 04              SUB B  #4        FREE CELLS UNTIL DOWN TO 256-BYTE BOUNDARY
06FE D7 27              STA B  XTMP+1
0700 DE 26              LDX    XTMP
0702 20 06              BRA    BLKF2
0704            BLKF1  EQU    *
0704 A7 02              STA A  CDR,X
0706 E7 03              STA B  CDR+1,X
0708 EE 02              LDX    CDR,X
070A            BLKF2  EQU    *
070A C0 04              SUB B  #4
070C 24 F6              BCC    BLKF1
070E D6 2B              LDA B  FREPTR+1  FILL IN LAST CELL
0710 E7 03              STA B  CDR+1,X
0712 D6 2A              LDA B  FREPTR
0714 E7 02              STA B  CDR,X
0716 D6 27              LDA B  XTMP+1    UPDATE FREPTR
0718 D7 2B              STA B  FREPTR+1
071A 97 2A              STA A  FREPTR
071C 39                 RTS
                *
```

```
071D            GETCEL  EQU     *
071D DE 2A              LDX     FREPTR      ANY CELLS ON FREE LIST?
071F 26 26              BNE     OLDGET
0721 96 2F              LDA A   ENDPTR+1    NOPE, ALLOCATE OFF ENDPTR
0723 26 10              BNE     GCL2
                * ABOUT TO BREAK INTO NEW 256-BYTE PAGE
0725 BD 07 B4           JSR     GCOL
0728 96 75              LDA A   GCFREE+1    A LOT OF CELLS FREED?
072A 81 14              CMP A   #MINFRE
072C 24 EF              BHS     GETCEL      YEP, LOOP AROUND
072E 96 74              LDA A   GCFREE
0730 26 E8              BNE     GETCEL      (YEP, MORE THAN 256)
0732 96 2E              LDA A   ENDPTR
0734 4A                 DEC A
0735 91 2C              CMP A   SPCPTR      ABOUT TO RUN OUT OF SPACE?
0737 23 1F              BLS     NOGCEL
0739 97 2E              STA A   ENDPTR      NOPE, ALLOCATE NEW DTPR OFF ENDPTR
073B 96 2F              LDA A   ENDPTR+1
073D            GCL2    EQU     *
073D 80 04              SUB A   #4
073F 97 2F              STA A   ENDPTR+1
0741 DE 2E              LDX     ENDPTR      POINT TO NEW CELL
0743 DF 3A              STX     CELPTR
0745 20 08              BRA     NEWGET      REJOIN COMMON CODE
0747            OLDGET  EQU     *
0747 DF 3A              STX     CELPTR      PULL CELL OFF FREE LIST
0749 EE 02              LDX     CDR,X
074B DF 2A              STX     FREPTR
074D DE 3A              LDX     CELPTR      ZERO OUT NEW CELL
074F            NEWGET  EQU     *
074F 6F 00              CLR     0,X
0751 6F 01              CLR     1,X
0753 6F 02              CLR     2,X
0755 6F 03              CLR     3,X
0757 39                 RTS
0758 CE 08 E5   NOGCEL  LDX     #NOGCMG
075B 7E 01 36           JMP     ERREX
                *
075E            PROPSH  EQU     *
                * PUSH X-REG, WITH LOW BIT SET IN DTPR TO SPECIFY GCOL TRACING
075E 7C 00 3F           INC     STKPTR+1
0761            PUSHX   EQU     *
                * STACK X-REG ON STKPTR, AND SET CC'S FOR X-REG
0761 DF 26              STX     XTMP        SAVE X TEMPORARILY
0763 BD B8              BSR     GETCEL      GET A CELL FOR THE ADDITION TO STACK
0765 96 26              LDA A   XTMP        SAVE X-REG IN CAR
0767 A7 00              STA A   CAR,X
0769 96 27              LDA A   XTMP+1
076B A7 01              STA A   CAR+1,X
076D 96 3E              LDA A   STKPTR      SAVE NEXT CELL IN CDR
076F A7 02              STA A   CDR,X
0771 96 3F              LDA A   STKPTR+1
0773 A7 03              STA A   CDR+1,X
0775 DF 3E              STX     STKPTR      UPDATE STKPTR
0777 DE 26              LDX     XTMP        RESTORE X-REG AND SET CC'S
0779 39                 RTS
                *
077A            POPX    EQU     *
                * PULL X-REG OFF OF STKPTR LIST
                * ERROR IF STKPTR = 0
077A DE 3E              LDX     STKPTR
077C 27 0F              BEQ     POPXER      OOPS!
077E A6 02              LDA A   CDR,X       UPDATE STKPTR
0780 97 3E              STA A   STKPTR
0782 A6 03              LDA A   CDR+1,X
0784 84 FE              AND A   #$FE        ENSURE THAT LOW BIT IS CLEAR
0786 97 3F              STA A   STKPTR+1
0788 BD 1F              BSR     FRECEL
078A EE 00              LDX     CAR,X       **ASSUME FRECEL DOESN'T TOUCH CAR
078C 39                 RTS
078D            POPXER  EQU     *
078D CE 08 D5           LDX     #EOSTKM     STACK UNDERFLOW!
0790 7E 01 36           JMP     ERREX
                *
077A            PROPOP  EQU     POPX        FOR SAFETY'S SAKE, POPX ALWAYS CLEARS LOW BIT ANYWAY
                *
0793            TOPX    EQU     *
                * RETURN X-REG STORED AT TOP OF STKPTR STACK
0793 DE 3E              LDX     STKPTR
0795 27 F6              BEQ     POPXER
0797 EE 00              LDX     CAR,X
0799 39                 RTS
                *
079A            ISVAR   EQU     *
                * CHECK THAT FORM CAN BE USED AS TARGET FOR "SET" OR AS A FORMAL ARGUMENT
                * RETURN WITH C-BIT CLEAR IF FORM IS LEGAL VAR
079A BD 01 3F           JSR     ISATOM      FIRST, IT MUST BE A SYMBOLIC ATOM
079D 25 08              BCS     ISVNO
079F 2F 06              BLE     ISVNO       (NIL OR NUMBER)
07A1 9C 5E              CPX     TATOM       SECOND, IT MUST NOT BE CERTAIN SPECIAL ATOMS
07A3 27 02              BEQ     ISVNO
07A5 0C                 CLC
07A6 39                 RTS
07A7            ISVNO   EQU     *
07A7 0D                 SEC
07A8 39                 RTS
                *
07A9            FRECEL  EQU     *
                * TAKES ADDR OF CELL IN X-REG
                * ADDS TO FREPTR LIST
                * X-REG, AND CAR,X NOT DISTURBED
07A9 96 2A              LDA A   FREPTR      SET NEW CDR
07AB A7 02              STA A   CDR,X
07AD 96 2B              LDA A   FREPTR+1
07AF A7 03              STA A   CDR+1,X
07B1 DF 2A              STX     FREPTR      UPDATE FREPTR
07B3 39                 RTS
                ***
07B4            GCOL    EQU     *
07B4 BD 43              BSR     GCTRCE      TRACE ALL CELLS
07B6 DE FE              LDX     ZERO        INITIALIZE THE FREE LIST
07B8 DF 2A              STX     FREPTR
07BA DF 74              STX     GCFREE      WILL BE COUNT OF CELLS
07BC 9F 72              STS     GCTEMP      SAVE STACK POINTER FOR STACK PAGE CHECK
07BE 7F 00 73           CLR     GCTEMP+1
07C1 DE 2E              LDX     ENDPTR      NOW SCAN THE CELLS
07C3            GCF1    EQU     *
07C3 9C 72              CPX     GCTEMP      IS THIS A STACK PAGE?
07C5 26 0B              BNE     GCF3
07C7 A6 FE              LDA A   254,X       YEP, SET UP POINTER TO NEXT PAGE
07C9 7C 00 72           INC     GCTEMP      SKIP UP TO NEXT DTPR
07CC DE 72              LDX     GCTEMP
07CE 97 72              STA A   GCTEMP
07D0 20 F1              BRA     GCF1
                *
07D2            GCF3    EQU     *
07D2 9C FC              CPX     ENDMEM      REACHED END OF MEM?
07D4 27 22              BEQ     GCFDUN
07D6 A6 03              LDA A   CDR+1,X     IS THIS CELL MARKED?
07D8 85 02              BIT A   #MRKBIT
07DA 26 0C              BNE     GCFCLR
07DC BD CB              BSR     FRECEL      NO, FREE UP CELL
07DE 7C 00 75           INC     GCFREE+1    INCREMENT FREE CELL COUNTER
07E1 26 0F              BNE     GCFINX
07E3 7C 00 74           INC     GCFREE
07E6 20 0A              BRA     GCFINX      AND ADVANCE TO NEXT CELL
                *
07E8            GCFCLR  EQU     *
```

```
07E8 88 02              EOR A   #MRKBIT     CLEAR MARK BITS
07EA A7 03              STA A   CDR+1,X
07EC A6 01              LDA A   CAR+1,X
07EE 88 02              EOR A   #MRKBIT
07F0 A7 01              STA A   CAR+1,X
07F2            GCFINX  EQU     *
07F2 08                 INX
07F3 08                 INX
07F4 08                 INX
07F5 08                 INX
07F6 20 CB              BRA     GCF1        LOOP TO CHECK NEXT CELL
                *
07F8            GCFDUN  EQU     *
07F8 39                 RTS
                *
07F9            GCTRCE  EQU     *
                * TRACE ALL CELLS OF INTEREST:
                *  FIRST THE X-REG STACK (INCLUDING XTMP IF STKPTR ODD)
                * THEN THE SYMBOL LIST,
                * THEN THE COREVL LIST AND FORM(!).
                * PRESERVES XTMP, FORM.  CLOBBERS XTMP2, SONPTR, DADPTR, GCTEMP.
07F9 DE FE              LDX     ZERO        INITIALIZE DADPTR
07FB DF 70              STX     DADPTR
07FD DE 26              LDX     XTMP        COPY XTMP TO GCTEMP
07FF DF 72              STX     GCTEMP
0801 96 3F              LDA A   STKPTR+1    TRACE XTMP?
0803 46                 ROR A
0804 DE 3E              LDX     STKPTR      SET UP XTMP2
0806 24 0B              BCC     GCT2
0808            GCT1    EQU     *
0808 DF 28              STX     XTMP2
080A DE 72              LDX     GCTEMP
080C BD 32              BSR     GCTOBJ      YEP...
080E 7A 00 29           DEC     XTMP2+1     CLEAR LOW BIT
0811 DE 28              LDX     XTMP2       POINT TO NEXT CELL
0813            GCT2    EQU     *
0813 27 06              BEQ     GCT4        (ALL DONE)
0815            GCT3    EQU     *
0815 BD 11              BSR     GCTMRK      MARK THIS CELL, SET UP GCTEMP
0817 22 FC              BHI     GCT3        (=BCC+BNE -- NO NEED TO MARK THIS CELL)
0819 25 ED              BCS     GCT1        AND LOOP
                *
081B            GCT4    EQU     *
                * TRACE THE SYMBOL LIST
081B DE 30              LDX     SYMLST
081D BD 21              BSR     GCTOBJ
                * NOW TRACE COREVL AND FORM (NOTE!! FORM MUST ALWAYS BE A FORM!!)
081F DE 6C              LDX     COREVL
0821 BD 1D              BSR     GCTOBJ
0823 DE 34              LDX     FORM
0825 BD 19              BSR     GCTOBJ
0827 39                 RTS             ALL DONE
                *
0828            GCTMRK  EQU     *
                * MARK THIS CELL AND STORE CAR IN GCTEMP, CDR IN X-REG
                * RETURN WITH C-BIT SET BY LOW BIT OF CDR
                * Z&N-BIT SET BY VALUE OF CDR
0828 A6 00              LDA A   CAR,X
082A 97 72              STA A   GCTEMP
082C A6 01              LDA A   CAR+1,X
082E 97 73              STA A   GCTEMP+1
0830 88 02              EOR A   #MRKBIT     (USE EOR BECAUSE MAY NOT BE FORM)
0832 A7 01              STA A   CAR+1,X
0834 A6 03              LDA A   CDR+1,X
0836 8A 02              ORA A   #MRKBIT
0838 A7 03              STA A   CDR+1,X
083A 46                 ROR A           SET C-BIT
083B EE 02              LDX     CDR,X       SET UP X-REG, Z & N-BIT
083D 09                 DEX
083E 09                 DEX
083F 39                 RTS
                *
0840            GCTOBJ  EQU     *
                * AND NOW THE FUN!!
                * SONPTR AND DADPTR POINT TO CURRENT CELL AND ITS FATHER
                * SET GC BIT IN CAR, THEN TRACE THE CAR.
                * SET GC BIT IN CDR, AND TRACE IT.
                * POINTERS ARE REVERSED SO THAT DADPTR MAY BE SET ON THE WAY BACK UP
                *  CLOBBERS ONLY A,B,X-REGS, DADPTR, AND SONPTR.
0840 DF 6E              STX     SONPTR      SET UP SONPTR, AND COND. CODES
0842 2E 48              BGT     GCOGO       AND ENTER TRACE LOOP
0844            GCTRET  EQU     *
0844 39                 RTS             (NIL OR NUM)
                *
0845            GCODAT  EQU     *
                * DONE WITH ATOM, CARRY BIT IS ASSUMED ON(!)
0845 09                 DEX             ADJUST TO POINT TO START OF CELL
0846 A6 03              LDA A   CDR+1,X     LOAD UP ACC
                * AND DROP INTO GCOCDD ...
0848            GCOCDD  EQU     *
0848 88 02              EOR A   #MRKBIT     CLEAR MARK BIT IN BACK POINTER
084A 97 71              STA A   DADPTR+1
084C A6 02              LDA A   CDR,X
084E 97 70              STA A   DADPTR
0850 96 6E              LDA A   SONPTR      RESET CDR
0852 A7 02              STA A   CDR,X
0854 96 6F              LDA A   SONPTR+1
0856 DF 6E              STX     SONPTR      SET UP NEW SONPTR
0858 24 03              BCC     GCOCD2      WAS IT AN ATOM?
085A 7C 00 6F           INC     SONPTR+1    YEP, SET LOW BIT
085D            GCOCD2  EQU     *
085D 8A 02              ORA A   #MRKBIT     SET CDR MARK
085F A7 03              STA A   CDR+1,X
0861            GCODUN  EQU     *
                * ON THE WAY BACK UP, CHECK IF CDR ALREADY TRACED
0861 DE 70              LDX     DADPTR
0863 27 DF              BEQ     GCTRET      (ALL DONE)
0865 D6 71              LDA B   DADPTR+1    TRACING AN ATOM?
0867 56                 ROR B
0868 25 DB              BCS     GCODAT      YEP, CLEARLY MUST BE DONE WITH IT
086A A6 03              LDA A   CDR+1,X     CDR MARKED?
086C 85 02              BIT A   #MRKBIT
086E 26 D8              BNE     GCOCDD
0870 E6 01              LDA B   CAR+1,X     NOPE, TRACE DOWN CDR
0872 E7 03              STA B   CDR+1,X     MOVE BACK POINTER
0874 D6 6F              LDA B   SONPTR+1
0876 CA 02              ORA B   #MRKBIT
0878 E7 01              STA B   CAR+1,X     RESET CAR WITH MARK BIT ON
087A 97 6F              STA A   SONPTR+1    SET NEW SONPTR
087C A6 02              LDA A   CDR,X       NOW THE SAME THING WITH THE HIGH BYTES
087E E6 00              LDA B   CAR,X
0880 E7 02              STA B   CDR,X
0882 D6 6E              LDA B   SONPTR
0884 E7 00              STA B   CAR,X
0886 97 6E              STA A   SONPTR
0888            GCOCA2  EQU     *
                * NOW START WITH A NEW FORM
0888 DE 6E              LDX     SONPTR
088A 2F D5              BLE     GCODUN      (NUM OR NIL)
088C            GCOGO   EQU     *
088C 96 6F              LDA A   SONPTR+1    ATOM?
088E 46                 ROR A
088F 25 1E              BCS     GCOATM      (YEP)
0891 E6 01              LDA B   CAR+1,X     IS DTPR ALREADY MARKED?
0893 C5 02              BIT B   #MRKBIT
0895 26 CA              BNE     GCODUN
0897 A6 00              LDA A   CAR,X       NOPE, NIL OR NUM?
0899 2F 10              BLE     GCOCAA
```

```
089B D7 6F          STA B  SONPTR+1  NOPE, TRACE DOWN CAR
089D 97 6E          STA A  SONPTR
089F 96 70          LDA A  DADPTR
08A1 A7 00          STA A  CAR,X
08A3 96 71          LDA A  DADPTR+1
08A5 8A 02          ORA A  #MRKBIT
08A7 A7 01          STA A  CAR+1,X
08A9 20 26          BRA    GCOCA4    JOIN COMMON CODE
                    *
08AB                GCOCAA EQU    *
                    * CAR IS NIL OR NUM, TRACE DOWN CDR
08AB A6 03          LDA A  CDR+1,X
08AD 20 09          BRA    GCOCA3    (JUST LIKE AN ATOM CELL)
                    *
08AF                GCOATM EQU    *
                    * FORM IS ATOM
08AF A6 02          LDA A  CDR+1-1,X ATOM ALREADY MARKED?
08B1 85 02          BIT A  #MRKBIT
08B3 26 AC          BNE    GCODUN
08B5 09             DEX            NOPE, POINT TO START OF CELL
08B6 E6 01          LDA B  CAR+1,X   COMPLEMENT GC BIT OF CAR
08B8                GCOCA3 EQU    *
                    * AT THIS POINT C-BIT SET IF WE HAVE AN ATOM...
08B8 C8 02          EOR B  #MRKBIT
08BA E7 01          STA B  CAR+1,X
08BC E6 02          LDA B  CDR,X     VALUE=NIL OR NUM?
08BE 2F 9D          BLE    GCOCD2    YEP, MARK CDR AND PROCEED
08C0 97 6F          STA A  SONPTR+1  NOPE, TRACE VALUE OF ATOM
08C2 D7 6E          STA B  SONPTR
08C4 96 70          LDA A  DADPTR
08C6 A7 02          STA A  CDR,X
08C8 96 71          LDA A  DADPTR+1
08CA 8A 02          ORA A  #MRKBIT
08CC A7 03          STA A  CDR+1,X
08CE 24 01          BCC    GCOCA4    DO WE HAVE AN ATOM?
08D0 08             INX            YEP, SET LOW BIT AGAIN
08D1                GCOCA4 EQU    *
08D1 DF 70          STX    DADPTR    SET NEW DADPTR
08D3 20 B3          BRA    GCOCA2    JOIN COMMON CODE
                    *
08D5 53             EOSTKN FCC    'STACK UNDERFLOW'
08E4 04                    FCB    4
08E5 4E             NOGCMG FCC    'NO MORE FREE SPACE'
08F7 04                    FCB    4
08F8 4E             STKEMG FCC    'NO ROOM FOR STACK'
0909 04                    FCB    4
                    *
090A                BEGADR EQU    *
                           END
```

```
NO ERROR(S) DETECTED

 SYMBOL TABLE:

ALP    0044   ATMINI 0139   BEGADR 090A   REGPTR 0020   BLKF1  0704
BLKF2  070A   BLKFRE 06FA   CAR    0000   CDR    0002   CELPTR 003A
CMPTM2 0042   CMPTMP 0040   CONATM 0060   CUREVL 006C   DADPTR 0070
DOTATM 0050   ENDMEM 00FC   ENDPTR 002E   EOI    0004   EOIATM 0064
EOSTKN 08D5   ERRBRK 0154   ERREX  0136   EVAL   0100   EVLATM 011B
FLP    0048   FORM   0034   FRECEL 07A9   FREPTR 002A   FRMNUM 014B
GCF1   07C3   GCF3   07D2   GCFCLR 07E8   GCFNUM 07FB   GCFINX 07F2
GCFREE 0074   GCL2   073D   GCOATM 08AF   GCOCA2 08B8   GCOCA3 08BB
GCOCA4 08D1   GCOCAA 08AB   GCOCD2 085D   GCOCDD 0848   GCODAT 0845
GCODUN 0861   GCOGO  088C   GCOL   07B4   GCT1   0809   GCT2   0813
GCT3   0815   GCT4   081B   GCTEMP 0072   GCTMRK 0828   GCTOBJ 0840
GCTRCE 07F9   GCTRET 0844   GETC   0142   GETCAR 0163   GETCEL 010D
GETSYM 0157   GFNXTL 0127   GNXTL  0124   ISATOM 013F   ISDTPR 013C
ISVAR  079A   ISVNO  07A7   LAMATM 0056   LBRKAT 0068   LOOKUP 0121
LPARAT 004C   LSTAD2 0130   LSTADD 012D   LSTEN0 077A   LSTEND 0133
LSTINI 012A   LSTPTR 0036   MINFRE 0014   MRKBIT 0002   NAMPTR 0032
NEWGET 074F   NILATM 004A   NLAMAT 0058   NLP    0046   NOGCEL 0758
NOGCMG 08E5   NSUBAT 005C   NUMFRM 0148   NUMINV 0145   OBEGAD 06A0
OLDGET 0747   PEEKC  00F0   POPX   077A   POPXER 078D   PRINR  0169
PRINT  0106   PROPOP 077A   PROPSH 075E   PUSHX  0761   PUTC   0151
PUTSYM 014E   QUOATM 0054   RBRKAT 006A   READ   0103   RPARAT 004E
RSETAT 0062   RSTFLG 00F1   SETATM 011E   SONPTR 006E   SPSAVE 003C
SQUTAT 0052   STKEMG 08F8   STKF2  06A9   STKF4  06B5   STKFRG 06A0
STKFUL 06DA   STKMIN 0032   STKPTR 003E   STP1   0026   STP2   0028
STR1   0022   STR2   0024   SUBLS2 0066   SUBLS3 0076   SUBRAT 005A
SYMLST 0030   SYMPTR 0038   TATOM  005E   TOPX   0793   UNRAVA 06E0
UNRAVL 06E2   UNRFRE 06F5   UNRVRT 06F2   XTMP   0026   XTMP2  0028
ZERO   00FE
```

```
                    OPT    NOG
0940                OBEGAD EQU    $940      $922 ENDS STACK
0000                CAR    EQU    0
0002                CDR    EQU    2
0004                EOI    EQU    4
                    * DIRECT PAGE STORAGE CELLS
0020                       ORG    $20
0020 0B A2          BEGPTR FDB    BEGADR    THIS SHOULD BE IN EACH MODULE
0022                STR1   RMB    2
0024                STR2   RMB    2
0026                STP1   RMB    2
0028                STP2   RMB    2
0026                XTMP   EQU    STP1
0028                XTMP2  EQU    STP2
002A                FREPTR RMB    2
002C                SPCPTR RMB    2
002E                ENDPTR RMB    2
0030                SYMLST RMB    2
0032                NAMPTR RMB    2
0034 00 00          FORM   FDB    0         ENSURE THAT FORM STARTS OUT AS LEGAL FORM
0036                LSTPTR RMB    2
0038                SYMPTR RMB    2
003A                CELPTR RMB    2
003C                SPSAVE RMB    2
003E                STKPTR RMB    2
0040                CMPTMP RMB    2
0042                CMPTM2 RMB    2
0044                ALP    RMB    2
0046                NLP    RMB    2
0048                FLP    RMB    2
004A                NILATM RMB    2
004C                LPARAT RMB    2
004E                RPARAT RMB    2
0050                DOTATM RMB    2
0052                SQUTAT RMB    2
0054                QUOATM RMB    2
0056                LAMATM RMB    2
0058                NLAMAT RMB    2
005A                SUBRAT RMB    2
005C                NSUBAT RMB    2
005E                TATOM  RMB    2
0060                CONATM RMB    2
0062                RSETAT RMB    2
0064                EOIATM RMB    2
0066                SUBLS2 RMB    2
0068                LBRKAT RMB    2
006A                RBRKAT RMB    2
006C                CUREVL RMB    2
006E                SONPTR RMB    2
0070                DADPTR RMB    2
0072                GCTEMP RMB    2
0074                GCFREE RMB    2
0076                SUBLS3 RMB    2
                    * SINGLE CHAR SLOTS ...
00F0                       ORG    $F0
00F0                PEEKC  RMB    1
00F1                RSTFLG RMB    1
                    * GLOBAL CONSTANTS
00FC                       ORG    $FC
                    * END OF MEMORY, MINUS 192 FOR NUMFRM, FRMNUM TABLES
00FC                ENDMEM RMB    2         INITIALIZED IN LISP
                    * ZERO WORD
00FE 00 00          ZERO   FDB    0
                    * GLOBAL VECTORS
0100                       ORG    $100
0100 7E 09 40              JMP    EVAL
0103                READ   RMB    3
0106                PRINT  RMB    3
0109                       RMB    3
```

```
010C                GETCEL RMB    3
010F                FRECEL RMB    3
0112                PUSHX  RMB    3
0115                POPX   RMB    3
0118                TOPX   RMB    3
011B 7E 09 4C              JMP    EVLATM
011E 7E 0B 65              JMP    SETATM
0121                LOOKUP RMB    3
0124 7E 0B 33              JMP    GNXTL
0127 7E 0B 4D              JMP    GFNXTL
012A 7E 0A FA              JMP    LSTINI
012D 7E 0B 06              JMP    LSTADD
0130 7E 0B 08              JMP    LSTAD2
0133 7E 0B 25              JMP    LSTEND
                    *LSTEN0 EQU POPX
0136                ERREX  RMB    3
0139                ATMINI RMB    3
013C 7E 0A E3              JMP    ISDTPR
013F 7E 0A EE              JMP    ISATOM
0142                GETC   RMB    3
0145                NUMINV RMB    3
0148                NUMFRM RMB    3
014B                FRMNUM RMB    3
014E                PUTSYM RMB    3
0151                PUTC   RMB    3
0154                ERRBRK RMB    3
0157                GETSYM RMB    3
015A                STKFRG RMB    3
015D                GCOL   RMB    3
0160                PROPSH RMB    3
0163                GETCAR RMB    3
0166                ISVAR  RMB    3
0169                PRINR  RMB    3
016C                PROPOP RMB    3
                    *
0940                       ORG    OBEGAD
                    *
0940                EVAL   EQU    *
                    * EVAL EXPECTS FORM PASSED IN X-REG
                    * RETURNS EVALUATION OF FORM IN X-REG AND "FORM"
0940 7D 00 F1              TST    RSTFLG    INSIDE A RESET?
0943 26 0C                 BNE    EVLEOI    YEP, RETURN EOIATM
0945 BD 0A EE              JSR    ISATOM
0948 25 0B                 BCS    EVLDTP    (DTPR)
094A 2F 02                 BLE    EVLRT     (NIL OR NUM)
094C                EVLATM EQU    *
094C EE 01                 LDX    CDR-1,X   RETURN CURRENT BINDING
094E                EVLRT  EQU    *
094E DF 34                 STX    FORM
0950 39                    RTS
0951                EVLEOI EQU    *
0951 DE 64                 LDX    EOIATM    RETURN EOIATM
0953 20 F9                 BRA    EVLRT
0955                EVLDTP EQU    *
0955 DF 34                 STX    FORM      SAVE X-REG TEMPORARILY (AND PROTECT IT)
0957 BD 01 5A              JSR    STKFRG    FRAGMENT THE STACK
095A 96 45                 LDA A  ALP+1     SAVE ALP,NLP,FLP ON STACK
095C 36                    PSH A
095D 96 44                 LDA A  ALP
095F 36                    PSH A
0960 96 47                 LDA A  NLP+1
0962 36                    PSH A
0963 96 46                 LDA A  NLP
0965 36                    PSH A
0966 96 49                 LDA A  FLP+1
0968 36                    PSH A
0969 96 48                 LDA A  FLP
096B 36                    PSH A
096C BD 01 0C              JSR    GETCEL    GET A CELL TO ADD TO CUREVL LIST
```

```
096F 96 34          LDA A  FORM
0971 A7 00          STA A  CAR,X
0973 96 35          LDA A  FORM+1
0975 A7 01          STA A  CAR+1,X
0977 96 6C          LDA A  CUREVL
0979 A7 02          STA A  CDR,X
097B 96 6D          LDA A  CUREVL+1
097D A7 03          STA A  CDR+1,X
097F DF 6C          STX    CUREVL
0981 DE 34          LDX    FORM        SAVE POINTER TO ARG LIST
0983 EE 02          LDX    CDR,X
0985 DF 44          STX    ALP
0987 DE 34          LDX    FORM        POINT BACK TO THIS FORM
0989 EE 00          LDX    CAR,X       POINT TO FUNC. OR EXPLICIT LAMBDA
098B BD B3          BSR    EVAL        RECURSE TO EVAL IT
098D BD 0A E3       JSR    ISDTPR      IS RESULT A DTPR?
0990 25 1A          BCS    EVLERR      NOPE
                * NOW X-REG SHOULD BE EXPLICIT LAMBDA EXPR
0992 DF 34          STX    FORM        SAVE FOR LATER
0994 EE 02          LDX    CDR,X
0996 DF 48          STX    FLP         (FUNCTION DEF LIST POINTER)
0998 DE 34          LDX    FORM        POINT BACK TO BEG. OF LIST
099A EE 00          LDX    CAR,X       NOW LOOK FOR LAMBDA,NLAMBDA,SUBR,NSUBR
099C 9C 5A          CPX    SUBRAT
099E 27 18          BEQ    EVLSUB
09A0 9C 5C          CPX    NSUBAT
09A2 27 0D          BEQ    EVLNSU
09A4 9C 56          CPX    LAMATM
09A6 27 6D          BEQ    EVLLAM
09A8 9C 58          CPX    NLAMAT
09AA 27 5C          BEQ    EVLNLA
09AC          EVLERR EQU   *
09AC CE 0B 8C       LDX    #EVLEMS
09AF 20 33          BRA    EVLER2
                *
09B1          EVLNSU EQU   *
                * EVALUATE (NSUBR.FUNCADDR)
09B1 DE FE          LDX    ZERO        SET UP A NULL LIST ON TOP OF STACK FOR POPFRE
09B3 BD 01 60       JSR    PROPSH
09B6 20 09          BRA    EVNSU2
                *
09B8          EVLSUB EQU   *
                * EVALUATE (SUBR.FUNCADDR)
09B8 BD 2F          BSR    EVLALS
09BA DE 64          LDX    EOIATM
09BC 7D 00 F1       TST    RSTFLG      IN A RESET?
09BF 26 18          BNE    EVLSB5      YEP, RETURN EOIATM
09C1          EVNSU2 EQU   *
09C1 DE 48          LDX    FLP         PICK UP THE ADDR
09C3 BD 0A EE       JSR    ISATOM
09C6 25 17          BCS    EVLSER      GACK!
09C8 2F 15          BLE    EVLSER      NIL OR NUMBER JUST AS BAD!
09CA 09             DEX                CONVERT TO CELL POINTER
09CB EE 00          LDX    CAR,X       GET ADDR OF FUNCTION
09CD A6 00          LDA A  0,X         CHECK MAGIC BYTE
09CF 81 21          CMP A  #'!
09D1 26 0C          BNE    EVLSER
09D3 6D 01          TST    1,X
09D5 26 08          BNE    EVLSER
                * ALP NOW HAS ARG LIST
09D7 AD 02          JSR    2,X         CALL THE FUNCTION
09D9          EVLSB5 EQU   *
09D9 DF 34          STX    FORM        SAVE RESULT
09DB BD 69          BSR    POPFRE      FREE THE ARG LIST
09DD 20 4A          BRA    EVLRT5      AND CLEAN UP
09DF          EVLSER EQU   *
09DF BD 65          BSR    POPFRE      FREE THE ARG LIST
09E1 CE 0B 6E       LDX    #EVLSMS     'BAD SUBR/NSUBR'
09E4          EVLER2 EQU   *
09E4 BD 01 54       JSR    ERRBRK
09E7 20 3E          BRA    EVLRT4
                *
09E9          EVLALS EQU   *
09E9 BD 0A FA       JSR    LSTINI      BUILD UP LIST ON STACK OF EVAL'D ARGS
09EC          EVLSB2 EQU   *
09EC DE 44          LDX    ALP
09EE BD 0B 33       JSR    GNXTL
09F1 25 0C          BCS    EVLSB3
09F3 DF 44          STX    ALP
09F5 DE 26          LDX    XTMP
09F7 BD 09 40       JSR    EVAL        EVAL NEXT ARG
09FA BD 0B 06       JSR    LSTADD      ADD TO LIST
09FD 20 ED          BRA    EVLSB2      AND LOOP
09FF          EVLSB3 EQU   *
09FF BD 01 15       JSR    LSTENO      END THE LIST
0A02 BD 01 18       JSR    TOPX        GET LIST AND STORE IN ALP
0A05 DF 44          STX    ALP
0A07 39             RTS
                *
0A08          EVLNLA EQU   *
                * EVALUATE (NLAMBDA (L) EXPR)
                * SIMULATE EVLALS BY MAKING ONE-ELEMENT LIST
                * WITH UNEVAL'D ARGLIST AS SINGLE "ACTUAL" ARG.
0A08 BD 0A FA       JSR    LSTINI
0A0B DE 44          LDX    ALP
0A0D BD 0B 06       JSR    LSTADD
0A10 BD 01 15       JSR    LSTENO
0A13 20 02          BRA    EVLNL2      JOIN COMMON CODE
                *
0A15          EVLLAM EQU   *
                * EVALUATE (LAMBDA (X Y) EXPR)
0A15 8D D2          BSR    EVLALS      EVALUATE ACTUAL ARGUMENTS
0A17          EVLNL2 EQU   *
0A17 BD 6A          BSR    EVLNSV      SAVE OLD AND SET NEW VALUES OF FORMALS
                * TOP OF STACK NOW POINTS TO LIST OF FORMALS
                * JUST BELOW IS LIST OF THEIR OLD VALUES
0A19          EVLBOD EQU   *
0A19 DE 48          LDX    FLP         EVAL BODY OF LAMBDA/NLAMBDA
0A1B BD 01 63       JSR    GETCAR      POINT TO BODY OF LAMBDA
0A1E BD 09 40       JSR    EVAL
0A21 DF 48          STX    FLP         SAVE RESULT IN FLP
0A23 8D 38          BSR    EVLRST      GO RESTORE OLD VALS OF FORMALS
0A25 DE 48          LDX    FLP         TRANSFER FINAL RESULT TO FORM
0A27          EVLRT4 EQU   *
0A27 DF 34          STX    FORM
0A29          EVLRT5 EQU   *
0A29 DE 6C          LDX    CUREVL      POP CUREVL LIST
0A2B 8D 25          BSR    GFNXTA
0A2D DE 44          LDX    ALP
0A2F DF 6C          STX    CUREVL
0A31 32             PUL A              RESTORE FLP,NLP,ALP
0A32 97 48          STA A  FLP
0A34 32             PUL A
0A35 97 49          STA A  FLP+1
0A37 32             PUL A
0A38 97 46          STA A  NLP
0A3A 32             PUL A
0A3B 97 47          STA A  NLP+1
0A3D 32             PUL A
0A3E 97 44          STA A  ALP
0A40 32             PUL A
0A41 97 45          STA A  ALP+1
0A43 DE 34          LDX    FORM        RETURN WITH RESULT IN X-REG
0A45 39             RTS
                *
0A46          POPFRE EQU   *
                * FREE THE ARG LIST (LIST HEAD IS AT TOP OF STACK)
0A46 BD 01 6C       JSR    PROPOP
```



```
0A49 27 06          BEQ    FFRDUN      (ALL DONE)
0A4B          PFR2   EQU   *
0A4B 8D 05          BSR    GFNXTA      ADVANCE ALP
0A4D          PFR3   EQU   *
0A4D DE 44          LDX    ALP         AND LOOP FOR CDR
0A4F 26 FA          BNE    PFR2
0A51          PFRDUN EQU   *
0A51 39             RTS
                *
0A52          GFNXTA EQU   *
0A52 A6 02          LDA A  CDR,X       ADVANCE ALP
0A54 97 44          STA A  ALP
0A56 A6 03          LDA A  CDR+1,X
0A58 97 45          STA A  ALP+1
0A5A 7E 01 0F       JMP    FRECEL      FREE UP THE CELL, AND RETURN
                *
0A5D          EVLRST EQU   *
0A5D BD 01 6C       JSR    PROPOP      SET UP NLP,ALP FOR RESTORE
0A60 DF 46          STX    NLP
0A62          EVLRS0 EQU   *
0A62 BD 01 6C       JSR    PROPOP
0A65          EVLRS1 EQU   *
0A65 DF 44          STX    ALP
0A67 27 04          BEQ    EVLRS3
0A69          EVLRS2 EQU   *
0A69 BD E7          BSR    GFNXTA      FREE CELL AND ADVANCE ALP
0A6B EE 00          LDX    CAR,X       (FRECEL DOESN'T TOUCH CAR)
0A6D          EVLRS3 EQU   *
0A6D DF 34          STX    FORM
0A6F DE 46          LDX    NLP
0A71 BD 0B 33       JSR    GNXTL
0A74 25 07          BCS    PFR3        ALL DONE, FREE UP REST OF ALP LIST
0A76 DF 46          STX    NLP
0A78 DE 26          LDX    XTMP
0A7A BD 0B 65       JSR    SETATM      STORE NEW VALUE FOR ATOM
0A7D DE 44          LDX    ALP
0A7F 27 EC          BEQ    EVLRS3
0A81 20 E6          BRA    EVLRS2
                *
0A83          EVLNSV EQU   *
                * COMMON CODE TO SAVE OLD AND SET NEW VALUES OF FORMALS
0A83 DE 3E          LDX    STKPTR      SET UP ALP AS ACTUAL ARG LIST POINTER
0A85 09             DEX
0A86 09             DEX
0A87 DF 44          STX    ALP
                * SET UP NLP AS FORMAL ARG LIST POINTER
0A89 DE 48          LDX    FLP
0A8B BD 0B 33       JSR    GNXTL
0A8E DF 48          STX    FLP         (AND ADVANCE FLP)
0A90 DE 26          LDX    XTMP
0A92 BD 01 60       JSR    PROPSH      SAVE FORMAL ARG LIST POINTER ON STACK
0A95          EVLNS1 EQU   *
0A95 BD 0B 33       JSR    GNXTL       GET NEXT FORMAL ARG AND ADVANCE NLP
0A98 25 3E          BCS    EVLNS5      (ALL DONE)
0A9A DF 46          STX    NLP
                * NEXT FORMAL ARG NOW IN XTMP
                * NOW GET PTR TO NEXT ACTUAL ARG
0A9C DE 44          LDX    ALP
0A9E EE 02          LDX    CDR,X       ANY MORE ACTUAL ARGS?
0AA0 26 0F          BNE    EVLNS2
                * NO MORE ACTUAL ARGS, CREATE ONE = NIL AND ADD TO LIST
0AA2 BD 01 0C       JSR    GETCEL
0AA5 DE 44          LDX    ALP
0AA7 96 3A          LDA A  CELPTR
0AA9 A7 02          STA A  CDR,X
0AAB 96 3B          LDA A  CELPTR+1
0AAD A7 03          STA A  CDR+1,X
0AAF DE 3A          LDX    CELPTR
0AB1          EVLNS2 EQU   *
                * ADVANCE ALP
0AB1 DF 44          STX    ALP
0AB3 EE 00          LDX    CAR,X       STORE NEXT ACTUAL ARG IN XTMP2
0AB5 DF 28          STX    XTMP2
                * GET OLD VALUE OF FORMAL ARG, SAVE IN LIST
0AB7 DE 26          LDX    XTMP        BUT FIRST, IS FORMAL ARG A LEGAL ATOM?
0AB9 BD 01 66       JSR    ISVAR
0ABC 25 1B          BCS    EVLNER      (NOPE!)
0ABE BD 09 4C       JSR    EVLATM
                * OLD VALUE NOW IN FORM, SAVE IN LIST
0AC1 DE 44          LDX    ALP
0AC3 96 34          LDA A  FORM
0AC5 A7 00          STA A  CAR,X
0AC7 96 35          LDA A  FORM+1
0AC9 A7 01          STA A  CAR+1,X
                * SET UP NEW VALUE FOR ATOM
0ACB DE 28          LDX    XTMP2
0ACD DF 34          STX    FORM
0ACF DE 26          LDX    XTMP
0AD1 BD 0B 65       JSR    SETATM
0AD4 DE 46          LDX    NLP         MOVE ON TO NEXT FORMAL ARG
0AD6 26 BD          BNE    EVLNS1
0AD8          EVLNS5 EQU   *
0AD8 39             RTS                ALL DONE
                * WE GOT A BAD FORMAL ARG, COMPLAIN...
0AD9          EVLNER EQU   *
0AD9 DF 34          STX    FORM
0ADB CE 0B 7D       LDX    #EVLNMS     'BAD FORMAL ARG'
0ADE BD 01 54       JSR    ERRBRK
0AE1 20 F5          BRA    EVLNS5      IGNORE RETURNED VALUE
                *
0AE3          ISDTPR EQU   *
                * RETURN C-BIT SET IF NOT DTPR
                * RETURN Z-BIT SET IF IS NIL
                * BCC WILL BRANCH ON DTPR
                * BEQ WILL BRANCH ON NIL
0AE3 0D             SEC
0AE4 DF 26          STX    XTMP        STORE AND SET CC'S
0AE6 2F 05          BLE    ISDTRT      NIL OR NUMBER==> RETURN WITH Z&C CORRECT
0AE8 96 27          LDA A  XTMP+1      CHECK LOW  BIT
0AEA 46             ROR A              SET C-BIT IF NOT DTPR
0AEB DE 26          LDX    XTMP        CLEAR Z-BIT
0AED          ISDTRT EQU   *
0AED 39             RTS                C-BIT NOW SET PROPERLY
                *
0AEE          ISATOM EQU   *
                * RETURN WITH C-BIT IF NOT ATOM
                * RETURN WITH Z-BIT IF NIL
                * RETURN WITH N-BIT SET IF NUMBER
                * BHI WILL BRANCH ON NON-NIL ATOM
0AEE 0C             CLC                SET UP C-BIT FOR ATOM
0AEF DF 26          STX    XTMP        STORE AND SET Z&N-BITS
0AF1 2F 06          BLE    ISATRT      NIL OR NUM, RET WITH Z,N,C CORRECT
0AF3 96 27          LDA A  XTMP+1      CHECK LOW BIT
0AF5 4C             INC A              (COMPLEMENTED --> C-BIT)
0AF6 46             ROR A
0AF7 DE 26          LDX    XTMP        (CLEAR Z-BIT)
0AF9          ISATRT EQU   *
0AF9 39             RTS                CC'S NOW SET UP CORRECTLY
                *
0AFA          LSTINI EQU   *
                * THIS SETS UP THE STACK FOR LSTADD
                * FIRST IT STACKS A NIL, THEN A POINTER
                * TO THE NIL
                *  CALL LSTINX IF X-REG POINTS TO LIST HEADER
0AFA DE FE          LDX    ZERO
0AFC BD 01 60       JSR    PROPSH
0AFF DE 3E          LDX    STKPTR
0B01          LSTINX EQU   *
```

```
0B01 09                 DEX
0B02 09                 DEX
0B03 7E 01 12           JMP     PUSHX       PUSH POINTER AND RETURN
                *
0B06            LSTADD  EQU     *
                * ADD FORM IN X-REG TO LIST POINTER ON STACK
                *  CALL LSTAD2 IF ALREADY STORED IN 'FORM'
0B06 DF 34              STX     FORM
0B08            LSTAD2  EQU     *
0B08 BD 01 0C           JSR     GETCEL      GET A NEW CELL
0B0B 96 34              LDA A   FORM        FILL IN CAR
0B0D A7 00              STA A   CAR,X
0B0F 96 35              LDA A   FORM+1
0B11 A7 01              STA A   CAR+1,X
0B13 BD 01 18           JSR     TOPX        GET LIST POINTER
0B16 96 3A              LDA A   CELPTR      (CELPTR FILLED IN BY GETCEL)
0B18 A7 02              STA A   CDR,X
0B1A D6 3B              LDA B   CELPTR+1
0B1C E7 03              STA B   CDR+1,X     NOW ADDED TO LIST
0B1E DE 3E              LDX     STKPTR      UPDATE LIST POINTER
0B20 A7 00              STA A   CAR,X
0B22 E7 01              STA B   CAR+1,X
0B24 39                 RTS
                *
0B25            LSTEND  EQU     *
                * FILL IN CDR OF LAST CELL IN LIST
                * ALSO POP OFF LIST POINTER
                *  CALL LSTEN2 IF FORM ALREADY STORED IN 'FORM'
0B25 DF 34              STX     FORM
0B27            LSTEN2  EQU     *
0B27 BD 01 15           JSR     POPX        POP OFF LIST POINTER
0B2A 96 34              LDA A   FORM
0B2C A7 02              STA A   CDR,X
0B2E 96 35              LDA A   FORM+1
0B30 A7 03              STA A   CDR+1,X
0B32 39                 RTS
                *
0115            LSTENO  EQU     POPX        USE THIS IF LIST ENDED WITH NIL
                *
0B33            GNXTL   EQU     *
                * GET NEXT LIST ELEMENT
                * X-REG CONTAINS LIST POINTER
                *  RETURNS CDR IN X-REG, CAR IN XTMP, C-BIT SET IF NOT DTPR
0B33 DF 26              STX     XTMP        DTPR?
0B35 2F 10              BLE     GNLNO
0B37 96 27              LDA A   XTMP+1
0B39 46                 ROR A
0B3A 25 0B              BCS     GNLNO
0B3C A6 00              LDA A   CAR,X       YEP, RETURN CAR IN XTMP
0B3E 97 26              STA A   XTMP
0B40 A6 01              LDA A   CAR+1,X
0B42 97 27              STA A   XTMP+1
0B44 EE 02              LDX     CDR,X
0B46 39                 RTS                 RETURN WITH C-BIT CLEAR
0B47            GNLNO   EQU     *
0B47 DE FE              LDX     ZERO        RETURN NIL IN XTMP
0B49 DF 26              STX     XTMP
0B4B 0D                 SEC
0B4C 39                 RTS                 RETURN WITH C-BIT SET
                *
0B4D            GFNXTL  EQU     *
                * SAME AS GNXTL, EXCEPT ALSO FREE CELL AFTER
                * RETURNING ITS CONTENTS
0B4D BD 0A E3           JSR     ISDTPR      DTPR?
0B50 25 F5              BCS     GNLNO
0B52 A6 02              LDA A   CDR,X       YEP. SAVE CDR
0B54 97 28              STA A   XTMP2
0B56 A6 03              STA A   CDR+1,X
0B58 97 29              STA A   XTMP2+1
0B5A BD 01 0F           JSR     FRECEL      FREE UP CELL
0B5D EE 00              LDX     CAR,X       SET UP XTMP
0B5F DF 26              STX     XTMP
0B61 DE 28              LDX     XTMP2       RETURN WITH X-REG ADVANCED
0B63 0C                 CLC                 AND C-BIT CLEAR
0B64 39                 RTS
                *
0B65            SETATM  EQU     *
                * EXPECTS ATOM IN X-REG
                * NEW BINDING IN 'FORM'
                * COMPENSATE FOR CELPTR =  ATOM-1
0B65 96 34              LDA A   FORM
0B67 A7 01              STA A   CDR-1,X
0B69 96 35              LDA A   FORM+1
0B6B A7 02              STA A   CDR+1-1,X
0B6D 39                 RTS
                *
                * ERROR MESSAGES
0B6E 42         EVLSMS  FCC     'BAD SUBR/NSUBR'
0B7C 04                 FCB     4
0B7D 42         EVLNMS  FCC     'BAD FORMAL ARG'
0B8B 04                 FCB     4
0B8C 49         EVLEMS  FCC     'ILLEGAL FUNCTION EVAL'
0BA1 04                 FCB     4
                *
0BA2            BEGADR  EQU     *
                END
```

NO ERROR(S) DETECTED

SYMBOL TABLE:

```
ALP     0044    ATMINI 0139   BEGADR 0BA2   BEGPTR 0020   CAR    0000
CDR     0002    CELPTR 003A   CMPTM2 0042   CMPTMP 0040   CONATM 0060
CUREVL 006C     DADPTR 0070   DOTATM 0050   ENDMEM 00FC   ENDPTR 002E
EQI     0004    EQIATM 0064   ERRBRK 0154   ERREX  0136   EVAL   0940
EVLALS 09E9     EVLATM 094C   EVLBOD 0A19   EVLDTP 0955   EVLEMS 0B8C
EVLEQI 0951     EVLER2 09E4   EVLERR 09AC   EVLLAM 0A15   EVLNER 0AD9
EVLNL2 0A17     EVLNLA 0A08   EVLNMS 0B7D   EVLNS1 0A95   EVLNS2 0AB1
EVLNS5 0AD8     EVLNSU 09B1   EVLNSV 0AB3   EVLRS0 0A62   EVLRS1 0A65
EVLRS2 0A69     EVLRS3 0A6D   EVLRST 0A5B   EVLRT  094E   EVLRT4 0A27
EVLRT5 0A29     EVLSB2 09EC   EVLSB3 09FF   EVLSB5 09D9   EVLSER 09DF
EVLSMS 0B6E     EVLSUB 09B8   EVNSU2 09C1   FLP    004B   FORM   0034
FRECEL 010F     FREPTR 002A   FRMNUM 0148   GCFREE 0074   GCOL   015D
GCTEMP 0072     GETC   0142   GETCAR 0163   GETCEL 010C   GETSYM 0157
GFNXTA 0A52     GFNXTL 0B4D   GNLNO  0B47   GNXTL  0B33   ISATOM 0AEE

ISATRT 0AF9     ISDTPR 0AE3   ISDTRT 0AED   ISVAR  0166   LAMATM 0056
LBRKAT 006B     LOOKUP 0121   LPARAT 004C   LSTAD2 0B08   LSTADD 0B06
LSTENO 0115     LSTEN2 0B27   LSTEND 0B25   LSTINI 0AFA   LSTINX 0B01
LSTPTR 0036     NAMPTR 0032   NILATM 004A   NLAMAT 0058   NLP    0046
NSUBAT 005C     NUMFRM 0148   NUMINV 0145   OBEGAD 0940   PEEKC  00F0
PFR2    0A4B    PFR3   0A4D   PFRDUN 0A51   POPFRE 0A46   POPX   0115
PRINR   0169    PRINT  0106   PROPOP 016C   PROPSH 0160   PUSHX  0112
PUTC    0151    PUTSYM 014E   QUOATM 0054   RBRKAT 006A   READ   0103
RPARAT 004E     RSETAT 0062   RSTFLG 00F1   SETATM 0B65   SONPTR 006E
SPCPTR 002C     SPSAVE 003C   SQUTAT 0052   STKFRG 015A   STKPTR 003E
STP1    0024    STP2   0028   STR1   0022   STR2   0024   SUBLS2 0066
SUBLS3 0076     SUBRAT 005A   SYMLST 0030   SYMPTR 0038   TATOM  005E
TOPX    0118    XTMP   0026   XTMP2  0028   ZERO   00FE
```

```
                OPT     NOG
0B80            OBEGAD  EQU     $0B80       (0B97 ENDS EVALSUBS)
0000            CAR     EQU     0
0002            CDR     EQU     2
0004            EQI     EQU     4
                * DIRECT PAGE STORAGE CELLS
0020                    ORG     $20
0020 0E 7D      BEGPTR  FDB     BEGADR      THIS SHOULD BE IN EACH MODULE
0022            STR1    RMB     2
0024            STR2    RMB     2
0026            STP1    RMB     2
0028            STP2    RMB     2
0026            XTMP    EQU     STP1
0028            XTMP2   EQU     STP2
002A            FREPTR  RMB     2
002C            SPCPTR  RMB     2
002E            ENDPTR  RMB     2
0030            SYMLST  RMB     2
0032            NAMPTR  RMB     2
0034            FORM    RMB     2
0036            LSTPTR  RMB     2
0038            SYMPTR  RMB     2
003A            CELPTR  RMB     2
003C            SPSAVE  RMB     2
003E            STKPTR  RMB     2
0040            CMPTMP  RMB     2
0042            CMPTM2  RMB     2
0044            ALP     RMB     2
0046            NLP     RMB     2
0048            FLP     RMB     2
004A            NILATM  RMB     2
004C            LPARAT  RMB     2
004E            RPARAT  RMB     2
0050            DOTATM  RMB     2
0052            SQUTAT  RMB     2
0054            QUOATM  RMB     2
0056            LAMATM  RMB     2
0058            NLAMAT  RMB     2
005A            SUBRAT  RMB     2
005C            NSUBAT  RMB     2
005E            TATOM   RMB     2
0060            CONATM  RMB     2
0062            RSETAT  RMB     2
0064            EQIATM  RMB     2
0066            SUBLS2  RMB     2
0068            LBRKAT  RMB     2
006A            RBRKAT  RMB     2
006C            CUREVL  RMB     2
006E            SONPTR  RMB     2
0070            DADPTR  RMB     2
0072            GCTEMP  RMB     2
0074            GCFREE  RMB     2           NUMBER OF FREE CELLS AFTER LAST GCOL
0076            SUBLS3  RMB     2           INITIALIZED IN SUBRS3
                * SINGLE CHAR SLOTS ...
00F0                    ORG     $F0
00F0            PEEKC   RMB     1
00F1            RSTFLG  RMB     1
                * GLOBAL CONSTANTS
00FC                    ORG     $FC
00FC            ENDMEM  RMB     2           INITIALIZED IN LISP
00FE 00 00      ZERO    FDB     0
                * GLOBAL VECTORS
0100                    ORG     $100
0100            EVAL    RMB     3
0103            READ    RMB     3
0106            PRINT   RMB     3
0109                    RMB     3
010C            GETCEL  RMB     3
010F            FRECEL  RMB     3
0112            PUSHX   RMB     3
0115            POPX    RMB     3
0118            TOPX    RMB     3
011B            EVLATM  RMB     3
011E            SETATM  RMB     3
0121            LOOKUP  RMB     3
0124            GNXTL   RMB     3
0127            GFNXTL  RMB     3
012A            LSTINI  RMB     3
012D            LSTADD  RMB     3
0130            LSTAD2  RMB     3
0133            LSTEND  RMB     3
0115            LSTENO  EQU     POPX
0136            ERREX   RMB     3
0139 7E 0B B0           JMP     ATMINI
013C            ISDTPR  RMB     3
013F            ISATOM  RMB     3
0142            GETC    RMB     3
0145            NUMINV  RMB     3
0148            NUMFRM  RMB     3
014B            FRMNUM  RMB     3
014E            PUTSYM  RMB     3
0151            PUTC    RMB     3
0154            ERRBRK  RMB     3
0157            GETSYM  RMB     3
015A            STKFRG  RMB     3
015D            GCOL    RMB     3
0160            PROPSH  RMB     3
0163            GETCAR  RMB     3
0166            ISVAR   RMB     3
0169            PRINR   RMB     3
016C            PROPOP  RMB     3
                *
0B80                    ORG     OBEGAD
7103            WARMS   EQU     $7103       RETURN ADDR FOR (SYS)
0048            NOTHAT  EQU     FLP         JUST USED TEMPORARILY
                *
0B80            ATMINI  EQU     *
                * INITIALIZE BUILT-IN ATOMS
                * IN THE FORM OF NULL-TERM'D STRING
                * FOLLOWED BY 8-BIT ADDR
0B80 CE 0D 8B           LDX     #ATILST     POINT TO BEGINNING OF ATOMS
0B83 DF 38              STX     SYMPTR
0B85            ATI2    EQU     *
0B85 BD 01 21           JSR     LOOKUP      LOOK IT UP, AND PUT ON SYMLST
                * (RESULT FROM LOOKUP ALSO IN FORM)
0B88 DE 38              LDX     SYMPTR      SCAN TO END OF ATOM NAME
0B8A            ATI3    EQU     *
0B8A 08                 INX
0B8B 6D 00              TST     0,X
0B8D 26 FB              BNE     ATI3
0B8F DF 38              STX     SYMPTR      SAVE POINTER
0BC1 EE 00              LDX     0,X         GET ADDR OF SLOT (NULL IS HIGH 8 BITS)
0BC3 96 34              LDA A   FORM        STORE ADDR OF ATOM
0BC5 A7 00              STA A   0,X
0BC7 96 35              LDA A   FORM+1
0BC9 A7 01              STA A   1,X
0BCB DE 38              LDX     SYMPTR      POINT TO NEXT NAME (IF ANY)
0BCD 08                 INX
0BCE 08                 INX
0BCF 6D 00              TST     0,X         AT END OF LIST?
0BD1 26 E2              BNE     ATI2
                * YEP, ALL DONE
                * INITIALIZE EQIATM
0BD3 CE 0B E0           LDX     #EQINAM
0BD6 BD 01 21           JSR     LOOKUP
0BD9 DF 64              STX     EQIATM
                * INITIALIZE "NOTHING" AND EQIATM TO POINT TO EQIATM
0BDB BD 2B              BSR     SSETAT
```

```
0BDD DE 48          LDX   NOTHAT
                  * (EDIATM STILL STORED IN "FORM")
0BDF BD 27          BSR   SSETAT
                  * INITIALIZE TATOM TO POINT TO ITSELF
0BE1 DE 5E          LDX   TATOM
0BE3 EF 01          STX   CDR-1,X
                  *
0BE5 CE 0D E2       LDX   #SUBRLS   INITIALIZE PRIMARY SUBRS AND NSUBRS
0BE8 BD 21          BSR   SNSINI
0BEA DE 5A          LDX   SUBRAT    COPY SELFUN TO LAMBDA/NLAMBDA, ETC.
0BEC EE 01          LDX   CDR-1,X
0BEE DF 34          STX   FORM
0BF0 DE 5C          LDX   #SUBAT
0BF2 BD 14          BSR   SSETAT
0BF4 DE 56          LDX   LAMATM
0BF6 BD 10          BSR   SSETAT
0BF8 DE 58          LDX   NLAMAT
0BFA BD 0C          BSR   SSETAT
0BFC DE 60          LDX   CONATM
0BFE BD 08          BSR   SSETAT
0C00 DE 66          LDX   SUBLS2    INITIALIZE SECONDARY LISTS
0C02 BD 07          BSR   SNSINI
0C04 DE 76          LDX   SUBLS3    INITIALIZE TERTIARY LISTS
0C06 20 03          BRA   SNSINI
                  *
0C08          SSETAT  EQU   *
0C08 7E 01 1E       JMP   SETATM    ALLOW FOR BSR'S
                  *
0C0B          SNSINI  EQU   *
0C0B C6 5A          LDA B #SUBRAT   INITIALIZE SUBRS
0C0D BD 03          BSR   SUBINI
0C0F 08             INX           POINT TO NSUBR LIST
0C10 C6 5C          LDA B #NSUBAT
                  *  BRA SUBINI   AND RETURN THROUGH SUBINI
                  *
0C12          SUBINI  EQU   *
                  * B-REG HOLDS ADDR OF ATOM (IN DIR. PAGE)
                  * X-REG POINTS TO LIST OF NAMES
                  *  FOLLOWED BY ADDRESS OF FUNCTION
                  * FUNCTION MUST START WITH FCB '! ; FCB 0
0C12 DF 38          STX   SYMPTR
0C14 7F 00 26       CLR   XTMP
0C17 D7 27          STA B XTMP+1
0C19 DE 26          LDX   XTMP      GET SUBR OR NSUBR ATOM
0C1B EE 00          LDX   0,X
0C1D DF 44          STX   ALP       SAVE IT
0C1F          SBILUP  EQU   *
0C1F DE 38          LDX   SYMPTR    POINT TO ATOM NAME
0C21 6D 00          TST   0,X       ALL DONE?
0C23 27 4D          BEQ   SBIRT     YUP
0C25 BD 01 21       JSR   LOOKUP    FORM ATOM (RETURNED IN 'FORM')
0C28 DF 46          STX   NLP       SAVE IT FOR LATER
0C2A DE 38          LDX   SYMPTR    SKIP PAST ATOM NAME
0C2C          SBILP2  EQU   *
0C2C 08             INX
0C2D 6D 00          TST   0,X
0C2F 26 FB          BNE   SBILP2
0C31 08             INX           POINT TO FUNCTION ADDR
0C32 DF 38          STX   SYMPTR    SAVE FOR LATER
0C34 EE 00          LDX   0,X       GET FUNC POINTER
0C36 A6 00          LDA A 0,X       CHECK FOR MAGIC BYTES
0C38 81 21          CMP A #'!
0C3A 26 37          BNE   SBIBAD
0C3C 6D 01          TST   1,X
0C3E 26 33          BNE   SBIBAD
0C40 DF 48          STX   FLP       SAVE FUNC POINTER
0C42 BD 01 0C       JSR   GETCEL    GET A CELL FOR FUNC PSEUDO ATOM
0C45 96 48          LDA A FLP       FILL IN CAR
0C47 A7 00          STA A CAR,X
0C49 96 49          LDA A FLP+1
0C4B A7 01          STA A CAR+1,X
0C4D 08             INX           MAKE AN ATOM-LIKE POINTER
0C4E DF 34          STX   FORM      SAVE IN FORM FOR SAFETY
0C50 BD 01 0C       JSR   GETCEL    GET A CELL FOR (SUBR.FUNCAD)
0C53 96 44          LDA A ALP       FILL IN CAR
0C55 A7 00          STA A CAR,X
0C57 96 45          LDA A ALP+1
0C59 A7 01          STA A CAR+1,X
0C5B 96 34          LDA A FORM      FILL IN CDR
0C5D A7 02          STA A CDR,X
0C5F 96 35          LDA A FORM+1
0C61 A7 03          STA A CDR+1,X
0C63 DF 34          STX   FORM      SET UP FOR SETATM
0C65 DE 46          LDX   NLP       GET ATOM
0C67 BD 01 1E       JSR   SETATM
0C6A DE 38          LDX   SYMPTR    INCREMENT PAST FUNC ADDR
0C6C 08             INX
0C6D 08             INX
0C6E DF 38          STX   SYMPTR
0C70 20 AD          BRA   SBILUP    AND LOOP AROUND
0C72          SBIRT   EQU   *
0C72 39             RTS
                  *
0C73          SBIBAD  EQU   *
0C73 CE 0E 3C       LDX   #SBIEMS   'BAD SUBR/NSUBR IN SUBINI'
0C76 7E 01 36       JMP   ERREX
                  ***
                  * BUILT-IN FUNCTIONS
2100          MAGURD  EQU   $2100     MAGIC BYTES
0C79          SELFUN  EQU   *
0C79 21 00          FDB   MAGURD
                  * RETURN SELF AS VALUE (USED FOR LAMBDA/NLAMBDA/SUBR/NSUBR/CONT)
0C7B DE 6C          LDX   CUREVL
0C7D EE 00          LDX   CAR,X
0C7F 39             RTS
0C80          SYSFUN  EQU   *
0C80 21 00          FDB   MAGURD
                  * (SYS)   RETURN TO DOS
0C82 7E 71 03       JMP   WARMS
0C85          EVLFUN  EQU   *
0C85 21 00          FDB   MAGURD
                  * GET FIRST ARG
0C87 BD 74          BSR   GCARA
0C89 7E 01 00       JMP   EVAL      AND EVAL IT, AND RETURN
0C8C          REDFUN  EQU   *
0C8C 21 00          FDB   MAGURD
0C8E 7E 01 03       JMP   READ
0C91          PRIFUN  EQU   *
0C91 21 00          FDB   MAGURD
                  * GET FIRST ARG
0C93 BD 68          BSR   GCARA
0C95 7E 01 06       JMP   PRINT
0C98          CARFUN  EQU   *
0C98 21 00          FDB   MAGURD
                  * GET FIRST ARG
0C9A BD 61          BSR   GCARA
0C9C BD 01 3C       JSR   ISDTPR
0C9F 27 05          BEQ   CARNIL    RETURN EOIATM AS (CAR NIL)
0CA1 25 12          BCS   CARERR    OOPS!
0CA3 EE 00          LDX   CAR,X
0CA5 39             RTS
0CA6          CARNIL  EQU   *
0CA6 DE 64          LDX   EOIATM
0CA8 39             RTS
0CA9          CDRFUN  EQU   *
0CA9 21 00          FDB   MAGURD
                  * GET FIRST ARG
0CAB BD 50          BSR   GCARA
```



```
0CAB BD 01 3C       JSR   ISDTPR
0CB0 25 03          BCS   CARERR    OOPS!
0CB2 EE 02          LDX   CDR,X
0CB4 39             RTS
0CB5          CARERR  EQU   *
0CB5 CE 0E 55       LDX   #CADEMS   'BAD ARG TO CAR/CDR'
0CB8 7E 01 54       JMP   ERRBRK    RETURN THROUGH ERRBRK
                  *
0CBB          CNSFUN  EQU   *
0CBB 21 00          FDB   MAGURD
0CBD BD 43          BSR   GNXTA     GET NEW CAR
0CBF DF 48          STX   FLP       SAVE IT
0CC1 BD 3A          BSR   GCARA     GET NEW CDR
0CC3 DF 46          STX   NLP       SAVE IT
0CC5 BD 01 0C       JSR   GETCEL
0CC8 96 48          LDA A FLP
0CCA A7 00          STA A CAR,X     FILL IN CAR AND CDR
0CCC 96 49          LDA A FLP+1
0CCE A7 01          STA A CAR+1,X
0CD0 96 46          LDA A NLP
0CD2 A7 02          STA A CDR,X
0CD4 96 47          LDA A NLP+1
0CD6 A7 03          STA A CDR+1,X
0CD8 39             RTS
0CD9          QUTFUN  EQU   *
0CD9 21 00          FDB   MAGURD
                  * GET FIRST ARG
0CDB BD 20          BSR   GCARA
0CDD 39             RTS
0CDE          SETFUN  EQU   *
0CDE 21 00          FDB   MAGURD
0CE0 BD 20          BSR   GNXTA
0CE2 BD 01 66       JSR   ISVAR
0CE5 25 0E          BCS   SETERR
0CE7 DF 46          STX   NLP
0CE9 BD 12          BSR   GCARA     GET VALUE
0CEB DF 34          STX   FORM      SET UP FOR SETATM
0CED DE 46          LDX   NLP
0CEF BD 01 1E       JSR   SETATM
0CF2 DE 34          LDX   FORM      RETURN WITH VALUE OF ATOM
0CF4 39             RTS
0CF5          SETERR  EQU   *
0CF5 DF 34          STX   FORM      SAVE FORM FOR PRINTOUT
0CF7 CE 0E 68       LDX   #SETEMS   'BAD ARG TO SET/PATOM'
0CFA 7E 01 54       JMP   ERRBRK    RETURN THROUGH ERRBRK
                  *
0CFD          GCARA   EQU   *
0CFD DE 44          LDX   ALP
0CFF 7E 01 63       JMP   GETCAR
                  *
0D02          GNXTA   EQU   *
0D02 DE 44          LDX   ALP
0D04 BD 01 24       JSR   GNXTL
0D07 DF 44          STX   ALP
0D09 DE 26          LDX   XTMP
0D0B 39             RTS
                  *
0D0C          EQFUN   EQU   *
0D0C 21 00          FDB   MAGURD
0D0E BD F2          BSR   GNXTA     COMPARE THE TWO ARGS
0D10 DF 34          STX   FORM
0D12 BD E9          BSR   GCARA
0D14 9C 34          CPX   FORM
0D16 27 03          BEQ   EQYES
0D18 DE FE          LDX   ZERO      RETURN WITH  NIL
0D1A 39             RTS
0D1B          EQYES   EQU   *
0D1B DE 5E          LDX   TATOM     RETURN WITH 'T'
0D1D 39             RTS
                  *
0D1E          CNDFUN  EQU   *
0D1E 21 00          FDB   MAGURD
                  * (COND (P1 E1) (P2 E2) ... (T ET))
0D20          CNDLUP  EQU   *
0D20 BD E0          BSR   GNXTA     GET NEXT (P E ...)
0D22 25 30          BCS   CNDRT     ALL DONE  (VALUE = NIL)
0D24 BD 01 3C       JSR   ISDTPR    IS IT A DTPR?
0D27 25 2C          BCS   CNDERT    NOPE, EVAL AND RETURN
0D29 A6 02          LDA A CDR,X     ADVANCE NLP AND GET FIRST ELEM
0D2B 97 46          STA A NLP
0D2D A6 03          LDA A CDR+1,X
0D2F 97 47          STA A NLP+1
0D31 EE 00          LDX   CAR,X
0D33 9C 5E          CPX   TATOM     IS IT = 'T'
0D35 27 07          BEQ   CNDTRU    YUP, ALWAYS TRUE
0D37 BD 01 00       JSR   EVAL      EVAL IT
0D3A DF 34          STX   FORM
0D3C 27 E2          BEQ   CNDLUP    LOOP AROUND IF = NIL(FALSE)
0D3E          CNDTRU  EQU   *
0D3E DE 46          LDX   NLP       PICK UP FIRST 'E'
0D40 DF 34          STX   FORM      INITIALIZE FORM IN CASE NO 'E'S
0D42          CNDTLP  EQU   *
0D42 BD 01 24       JSR   GNXTL
0D45 25 0B          BCS   CNDFRT    ALL DONE, RETURN WITH 'FORM'
0D47 DF 46          STX   NLP
0D49 DE 26          LDX   XTMP
0D4B BD 01 00       JSR   EVAL      EVAL THE 'E'
0D4E DE 46          LDX   NLP       POINT TO NEXT 'E'
0D50 2E F0          BGT   CNDTLP    AND LOOP
0D52          CNDFRT  EQU   *
0D52 DE 34          LDX   FORM
0D54          CNDRT   EQU   *
0D54 39             RTS
0D55          CNDERT  EQU   *
0D55 7E 01 00       JMP   EVAL      EVAL AND RETURN
                  *
0D58          PATFUN  EQU   *
0D58 21 00          FDB   MAGURD
0D5A BD A1          BSR   GCARA     GET FIRST ARG
0D5C BD 01 3F       JSR   ISATOM    IS IT AN ATOM?
0D5F 25 94          BCS   SETERR    NOPE!
0D61 2E 03          BGT   PATF2
0D63          PATF1   EQU   *
0D63 7E 01 4E       JMP   PUTSYM    NIL OR NUMBER, JUST LET PUTSYM TAKE IT
0D66          PATF2   EQU   *
0D66 DF 34          STX   FORM      SAVE FORM
0D68 09             DEX           GET PRINT NAME POINTER
0D69 EE 00          LDX   CAR,X
0D6B A6 00          LDA A 0,X       IS IT QUOTED?
0D6D 81 22          CMP A #'"
0D6F 27 04          BEQ   PATF3
0D71 DE 34          LDX   FORM      NOPE, LET PRINT TAKE OVER
0D73 20 EE          BRA   PATF1
0D75          PATF3   EQU   *
0D75 08             INX
0D76 A6 00          LDA A 0,X
0D78 27 0E          BEQ   PATFRT    ALL DONE
0D7A 81 22          CMP A #'"       IS IT A "?
0D7C 26 05          BNE   PATF4     NOPE, JUST PRINT IT
0D7E 08             INX           YEP, ADVANCE TO NEXT CHAR
0D7F A6 00          LDA A 0,X
0D81 27 05          BEQ   PATFRT    ALL DONE
0D83          PATF4   EQU   *
0D83 BD 01 51       JSR   PUTC
0D86 20 ED          BRA   PATF3     PRINT IT AND LOOP AROUND
0D88          PATFRT  EQU   *
0D88 DE 34          LDX   FORM      RETURN WITH ATOM AS VALUE
0D8A 39             RTS
```

```
                        *
0D8B            ATILST  EQU     *       DATA FOR ATMINI SUBR
0D8B 6E                 FCC     'nil'
0D8E 00                 FCB     0
0D8F 4A                 FCB     NILATM
0D90 74                 FCC     't'
0D91 00                 FCB     0
0D92 5E                 FCB     TATOM
0D93 28                 FCC     '('
0D94 00                 FCB     0
0D95 4C                 FCB     LPARAT
0D96 29                 FCC     ')'
0D97 00                 FCB     0
0D98 4E                 FCB     RPARAT
0D99 5B                 FCC     '['
0D9A 00                 FCB     0
0D9B 6B                 FCB     LBRKAT
0D9C 5D                 FCC     ']'
0D9D 00                 FCB     0
0D9E 6A                 FCB     RBRKAT
0D9F 2E                 FCC     '.'
0DA0 00                 FCB     0
0DA1 50                 FCB     DOTATM
0DA2 27                 FCB     ''
0DA3 00                 FCB     0
0DA4 52                 FCB     SQUTAT
0DA5 71                 FCC     'quote'
0DAA 00                 FCB     0
0DAB 54                 FCB     QUOATM
0DAC 6C                 FCC     'lambda'
0DB2 00                 FCB     0
0DB3 56                 FCB     LAMATM
0DB4 6E                 FCC     'nlambda'
0DBB 00                 FCB     0
0DBC 58                 FCB     NLAMAT
0DBD 73                 FCC     'subr'
0DC1 00                 FCB     0
0DC2 5A                 FCB     SUBRAT
0DC3 6E                 FCC     'nsubr'
0DC8 00                 FCB     0
0DC9 5C                 FCB     NSUBAT
0DCA 72                 FCC     'reset'
0DCF 00                 FCB     0
0DD0 62                 FCB     RSETAT
0DD1 63                 FCC     'cont'
0DD5 00                 FCB     0
0DD6 60                 FCB     CONATM
0DD7 6E                 FCC     'nothing'
0DDE 00                 FCB     0
0DDF 48                 FCB     NOTHAT
                *  EOINAM SERVES TO END ATILST
0DE0 00 00      EOINAM  FDB     0       ZERO LENGTH PRINTNAME
                *
0DE2            SUBRLS  EQU     *
0DE2 65                 FCC     'eval'
0DE6 00                 FCB     0
0DE7 0C 85              FDB     EVLFUN
0DE9 63                 FCC     'car'
0DEC 00                 FCB     0
0DED 0C 98              FDB     CARFUN
0DEF 63                 FCC     'cdr'
0DF2 00                 FCB     0
0DF3 0C A9              FDB     CDRFUN
0DF5 70                 FCC     'print'
0DFA 00                 FCB     0
0DFB 0C 91              FDB     PRIFUN
0DFD 70                 FCC     'patom'
0E02 00                 FCB     0
0E03 0D 58              FDB     PATFUN
0E05 63                 FCC     'cons'
0E09 00                 FCB     0
0E0A 0C BB              FDB     CNSFUN
0E0C 73                 FCC     'set'
0E0F 00                 FCB     0
0E10 0C DE              FDB     SETFUN
0E12 65                 FCC     'eq'
0E14 00                 FCB     0
0E15 0D 0C              FDB     EQFUN
                *
0E17 00                 FCB     0
                *
0E18            NSUBLS  EQU     *       MUST FOLLOW SUBRLS IMMEDIATELY
0E18 72                 FCC     'read'
0E1C 00                 FCB     0
0E1D 0C 8C              FDB     REDFUN
0E1F 71                 FCC     'quote'
0E24 00                 FCB     0
0E25 0C D9              FDB     QUTFUN
0E27 73                 FCC     'sys'
0E2A 00                 FCB     0
0E2B 0C 80              FDB     SYSFUN
0E2D 63                 FCC     'cond'
0E31 00                 FCB     0
0E32 0D 1E              FDB     CNDFUN
0E34 73                 FCC     'subr'
0E38 00                 FCB     0
0E39 0C 79              FDB     SELFUN
                *
0E3B 00                 FCB     0
                *
0E3C 42         SBIEMS  FCC     'BAD SUBR/NSUBR IN SUBINI'
0E54 04                 FCB     4
0E55 42         CADEMS  FCC     'BAD ARG TO CAR/CDR'
0E67 04                 FCB     4
0E68 42         SETEMS  FCC     'BAD ARG TO SET/PATOM'
0E7C 04                 FCB     4
                *
0E7D            BEGADR  EQU     *
                        END
```

```
NO ERROR(S) DETECTED

 SYMBOL TABLE:

ALP    0044   AT12   0BB5   AT13   0BBA   ATILST 0D8B   ATMINI 0BB0


BEGADR 0E7D   BEGPTR 0020   CADEMS 0E55   CAR    0000   CARERR 0CB5
CARFUN 0C98   CARNIL 0CA6   CDR    0002   CDRFUN 0CA9   CELPTR 003A
CMPTM2 0042   CMPTMP 0040   CNDERT 0D55   CNDFRT 0D52   CNDFUN 0D1E
CNDLUP 0D20   CNDRT  0D54   CNDTLP 0D42   CNDTRU 0D3E   CNSFUN 0CBB
CONATM 0060   CUREVL 006C   DADPTR 0070   DOTATM 0050   ENDMEM 00FC
ENDPTR 002E   EOI    0004   EOIATM 0064   EOINAM 0DE0   EQFUN  0D0C
EQYES  0D1B   ERRBKK 0154   ERREX  0136   EVAL   0100   EVLATM 011B
EVLFUN 0C85   FLP    0048   FORM   0034   FRECEL 010F   FREPTR 002A
FRNNUM 0148   GCARA  0CFD   GCFREE 0074   GCOL   015D   GCTEMP 0072
GETC   0142   GETCAR 0163   GETCEL 010C   GETSYN 0157   GFNXTL 0127
GNXTA  0D02   GNXTL  0124   ISATOM 013F   ISBTPR 013C   ISVAR  0166
LAMATM 0056   LBRKAT 0068   LOOKUP 0121   LPARAT 004C   LSTAD2 0130
LSTADD 012D   LSTEMO 0115   LSTEMB 0133   LSTINI 012A   LSTPTR 0036
MAGWRD 2100   NAMPTR 0032   NILATM 004A   NLAMAT 0058   NLP    0046
NOTHAT 0048   NSUBAT 005C   NSUBLS 0E18   NUMFRM 0148   NUMTMV 0145
OBEGAD 0B80   PATF1  0D63   PATF2  0D66   PATF3  0D75   PATF4  0D83
PATFRT 0D88   PATFUN 0D58   PEEKC  00F0   POPX   0115   PRIFUN 0C91
PRINK  0169   PRINT  0106   PROPOP 016C   PRUPSH 0160   PUSHX  0112
PUTC   0151   PUTSYN 014E   QUOATM 0054   QUTFUN 0CD9   RBRKAT 006A
READ   0103   REDFUN 0C8C   RPARAT 004E   RSETAT 0062   RSTFLG 00F1
SBIBAD 0C73   SBIEMS 0E3C   SBILP2 0C2C   SBILUP 0C1F   SBIRT  0C72
SELFUN 0C79   SETATM 011E   SETEMS 0E68   SETERR 0CF5   SETFUN 0CDE
SNSINI 0C08   SONPTR 006E   SPCPTR 002C   SPSAVE 003C   SQUTAT 0052
SSETAT 0C08   STKFRG 015A   STKPTR 003E   STP1   0026   STP2   0028
STR1   0022   STR2   0024   SUBINI 0C12   SUBLS2 0066   SUBLS3 0076
SUBRAT 005A   SUBRLS 0DE2   SYMLST 0030   SYMPTR 0038   SYSFUN 0C80
TATOM  005E   TOPX   0118   WARMS  7103   XTMP   0026   XTMP2  0028
ZERO   00FE
```

```
                        OPT     NOG
0E90            OBEGAD  EQU     $E90    $E7B ENDS SUBRS1
0000            CAR     EQU     0
0002            CDR     EQU     2
0004            EOI     EQU     4
                * DIRECT PAGE STORAGE CELLS
0020                    ORG     $20
0020 11 C4      BEGPTR  FDB     BEGADR  THIS SHOULD BE IN EACH MODULE
0022            STR1    RMB     2
0024            STR2    RMB     2
0026            STP1    RMB     2
0028            STP2    RMB     2
0026            XTMP    EQU     STP1
0028            XTMP2   EQU     STP2
002A            FREPTR  RMB     2
002C            SPCPTR  RMB     2
002E            ENDPTR  RMB     2
0030            SYMLST  RMB     2
0032            NAMPTR  RMB     2
0034            FORM    RMB     2
0036            LSTPTR  RMB     2
0038            SYMPTR  RMB     2
003A            CELPTR  RMB     2
003C            SPSAVE  RMB     2
003E            STKPTR  RMB     2
0040            CMPTMP  RMB     2
0042            CMPTM2  RMB     2
0044            ALP     RMB     2
0046            NLP     RMB     2
0048            FLP     RMB     2
004A            NILATM  RMB     2
004C            LPARAT  RMB     2
004E            RPARAT  RMB     2
0050            DOTATM  RMB     2
0052            SQUTAT  RMB     2
0054            QUOATM  RMB     2
0056            LAMATM  RMB     2
0058            NLAMAT  RMB     2
005A            SUBRAT  RMB     2
005C            NSUBAT  RMB     2
005E            TATOM   RMB     2
0060            CONATM  RMB     2
0062            RSETAT  RMB     2
0064            EOIATM  RMB     2
0066 10 96              FDB     SUBLS2
0068            LBRKAT  RMB     2
006A            RBRKAT  RMB     2
006C            CUREVL  RMB     2
006E            SONPTR  RMB     2
0070            DADPTR  RMB     2
0072            GCTEMP  RMB     2
0074            GCFREE  RMB     2       NUMBER OF FREE CELLS AFTER LAST GCOL
0076            SUBLS3  RMB     2       INITIALIZED IN SUBRS3
                * SINGLE CHAR SLOTS ...
00F0                    ORG     $F0
00F0            PEEKC   RMB     1
00F1            RSTFLG  RMB     1
                * GLOBAL CONSTANTS
00FC                    ORG     $FC
00FC            ENDMEM  RMB     2       INITIALIZED IN LISP
                * ZERO WORD
00FE 00 00      ZERO    FDB     0
                * GLOBAL VECTORS
0100                    ORG     $100
0100            EVAL    RMB     3
0103            READ    RMB     3
0106            PRINT   RMB     3
0109                    RMB     3
010C            GETCEL  RMB     3
```

```
010F            FRECEL  RMB     3
0112            PUSHX   RMB     3
0115            POPX    RMB     3
0118            TOPX    RMB     3
011B            EVLATM  RMB     3
011E            SETATM  RMB     3
0121            LOOKUP  RMB     3
0124            GNXTL   RMB     3
0127            GFNXTL  RMB     3
012A            LSTINI  RMB     3
012D            LSTADD  RMB     3
0130            LSTAD2  RMB     3
0133            LSTEND  RMB     3
0115            LSTENO  EQU     POPX
0136            ERREX   RMB     3
0139            ATNINI  RMB     3
013C            ISDTPR  RMB     3
013F            ISATOM  RMB     3
0142 7E 0E 90   GETC    JMP     GTCTTY
0145            NUMINV  RMB     3
0148            NUMFRM  RMB     3
014B            FRMNUM  RMB     3
014E            PUTSYM  RMB     3
0151 7E 71 12   PUTC    JMP     PTCTTY
0154            ERRBRK  RMB     3
0157            GETSYM  RMB     3
015A            STKFRG  RMB     3
015D            GCOL    RMB     3
0160            PROPSH  RMB     3
0163            GETCAR  RMB     3
0166            ISVAR   RMB     3
0169            PRINR   RMB     3
016C            PROPOP  RMB     3
                *
0E90            ORG     OREGAD
                *
2100            MAGWRD  EQU     $2100   ('!')
                *
709A            DOSPCH  EQU     $709A   DOS PREV CHAR.
7082            DOSEOC  EQU     $7082   DOS END OF COMMAND CHAR.
7094            DOSBPT  EQU     $7094   DOS LINE BUFFER POINTER
7115            INBUFF  EQU     $7115   DOS, READ NEXT LINE
7121            NXTCH   EQU     $7121   DOS, GET NEXT CHAR FROM BUF
7806            DOSFNS  EQU     $7806
7740            DOSFCB  EQU     $7740
712D            DOSTEX  EQU     $712D
713C            DOSRPT  EQU     $713C   DOS, REPORT ERROR
708C            DOSCWD  EQU     $708C   DOS, CURRENT WORKING DRIVEN
711E            PCRLF   EQU     $711E   PRINT <CR>,<LF>
7112            PTCTTY  EQU     $7112   PUT A CHAR
                *
0E90            GTCTTY  EQU     *
                * RETURNS NEXT CHAR IN A-REG
                * USES SYSTEM INPUT BUFFER
                * FORCES SYSTEM TO IGNORE E.O.COMMAND CHAR
0E90 B6 70 9A           LDA A  DOSPCH   GET PREV CHAR
0E93 B1 70 82           CMP A  DOSEOC   IS IT E.O.COMMAND?
0E96 26 0B              BNE    GETC2
0E98 7C 70 95           INC    DOSBPT+1 YES,  INCREMENT POINTER PAST IT
0E9B 26 03              BNE    GETC3
0E9D 7C 70 94           INC    DOSBPT
0EA0            GETC3   EQU    *
0EA0 7E 71 21           JMP    NXTCH    GET NEXT CHAR AND RETURN
                *
0EA3            GETC2   EQU    *
0EA3 B1 0D              CMP A  #$0D     WAS LAST CHAR <CR>?
0EA5 26 F9              BNE    GETC3
0EA7 BD 71 15           JSR    INBUFF   YEP, REQUEST A NEW BUFFER FULL
0EAA BD 71 1E           JSR    PCRLF    ECHO <CR,LF>
```

```
0EAD 7F 70 9A           CLR    DOSPCH   RETURN <LF>
0EB0 86 0A              LDA A  #$0A
0EB2 39                 RTS
                *
0EB3            LODFUN  EQU    *
0EB3 21 00              FDB    MAGWRD
                * (LOAD 'PROG.TXT)
                * OPENS FILE, CHANGES GETC TO READ DISK FOR CHARS,
                * LOOPS READ AND EVAL UNTIL READ RETURNS EOIATM.
                * RETURNS NIL
0EB5 CE 77 40           LDX    #DOSFCB  SET UP FLP FOR OPENFL/OPNTTY
0EB8 DF 48              STX    FLP
0EBA BD 17              BSR    GNXTA    PICK UP ARG
0EBC BD 0F 19           JSR    OPENFL   GO OPEN FILE
0EBF 25 0F              BCS    LODRT    OPEN FAILED, SIMPLY RETURN NIL
0EC1            LODLUP  EQU    *
0EC1 BD 01 03           JSR    READ     READ NEXT FORM
0EC4 9C 64              CPX    EOIATM   EOI?
0EC6 27 05              BEQ    LODRT1   YEP, ALL DONE
0EC8 BD 01 00           JSR    EVAL     NOPE, EVAL IT
0ECB 20 F4              BRA    LODLUP   AND LOOP AROUND
0ECD            LODRT1  EQU    *
0ECD BD 0F 02           JSR    CLSFN1   RESET GETC
0ED0            LODRT   EQU    *
0ED0 DE FE              LDX    ZERO     RETURN NIL
0ED2 39                 RTS
                *
0ED3            GNXTA   EQU    *
0ED3 DE 44              LDX    ALP
0ED5 BD 01 24           JSR    GNXTL
0ED8 DF 44              STX    ALP
0EDA DE 26              LDX    XTMP
0EDC 39                 RTS
                *
0EDD            OPNFUN  EQU    *
0EDD 21 00              FDB    MAGWRD
                * (OPEN 'FILE.TXT)
                * OPENS FILE, SUBSEQUENT  READS  WILL
                * READ FROM FILE
                * RETURNS ATOM IF SUCCEEDS
                *  RETURNS NIL IF OPEN FAILS
0EDF CE 77 40           LDX    #DOSFCB  SET UP FLP FOR READ
0EE2            OPNF2   EQU    *
0EE2 DF 48              STX    FLP
0EE4 BD ED              BSR    GNXTA
0EE6 DF 34              STX    FORM
0EE8 BD 2F              BSR    OPENFL
0EEA 25 03              BCS    OPNFER
0EEC DE 34              LDX    FORM     RETURN WITH ATOM
0EEE 39                 RTS
0EEF            OPNFER  EQU    *
0EEF DE FE              LDX    ZERO     RETURN WITH NIL
0EF1 39                 RTS
                *
0EF2            OPOFUN  EQU    *
0EF2 21 00              FDB    MAGWRD
                * (OPENO 'OFILE.TXT)
                * OPENS FILE FOR OUTPUT, RETURNS NIL IF OPEN FAILS
0EF4 CE 11 04           LDX    #OUTFCB
0EF7 20 E9              BRA    OPNF2
                *
0EF9            CLOFUN  EQU    *
0EF9 21 00              FDB    MAGWRD
                * (CLOSEO)
                * CLOSES FILE WHICH WAS OPEN FOR OUTPUT
0EFB            CLOFN1  EQU    *
0EFB CE 11 04           LDX    #OUTFCB
0EFE 20 05              BRA    CLSF2
                *
```

```
0F00            CLSFUN  EQU    *
0F00 21 00              FDB    MAGWRD
                * CLOSES FILE, SUBSEQUENT READS WILL COME  FROM
                * TTY
0F02            CLSFN1  EQU    *
0F02 CE 77 40           LDX    #DOSFCB
0F05            CLSF2   EQU    *
0F05 DF 48              STX    FLP
0F07 86 04              LDA A  #4
0F09 A7 00              STA A  0,X
0F0B BD 78 06           JSR    DOSFNS
0F0E BD 0F 97           JSR    OPNTTY   RESET GETC/PUTC
0F11 20 DC              BRA    OPNFER   AND RETURN WITH NIL
                *
0F13            OPNERR  EQU    *
0F13 CE 10 01           LDX    #OPNENS  'BAD ARG TO OPEN/LOAD'
0F16 BD 01 54           JSR    ERRBRK   PICK UP REPLACEMENT ATOM
                *  BRA OPENFL  AND LOOP TO RE-CHECK
                *
0F19            OPENFL  EQU    *
                * EXPECTS ATOM IN X-REG, USES PRINT NAME OF ATOM
                *  FOR FILENAME  (THROWS  AWAY ALL '"'S)
0F19 BD 01 3F           JSR    ISATOM   IS IT AN ATOM?
0F1C 25 F5              BCS    OPNERR   NOPE...
0F1E 2B F3              BLT    OPNERR   YEP, BUT IT IS A NUMBER!
                * NIL MEANS SWITCH BACK TO TTY INPUT
                * T MEANS SWITCH BACK TO DISK INPUT (MUST ALREADY
                *  BE OPENED)
                * OTHERWISE, USE PRINT NAME AS FILENAME
0F20 27 75              BEQ    OPNTTY
0F22 9C 64              CPX    EOIATM   EOIATM?
0F24 27 71              BEQ    OPNTTY   YEP, SWITCH BACK TO TTY (E.G. ON RESET)
0F26 9C 5E              CPX    TATOM    T?
0F28 27 56              BEQ    OPNDSK   SWITCH BACK TO DISK
                * MOVE CHARS INTO FCB
                * BY PUSHING ONTO STACK, AND THEN POPPING
                *  OFF INTO NAME AREA
0F2A C6 03              LDA B  #3       LIMIT ON EXTENSION
0F2C D7 26              STA B  XTMP     STORE FOR LATER
0F2E C6 08              LDA B  #8       LIMIT ON FILE NAME
0F30 09                 DEX            POINT TO PRINT NAME
0F31 EE 00              LDX    CAR,X
0F33            OPNLP1  EQU    *
0F33 A6 00              LDA A  0,X      GET NEXT CHAR
0F35 27 10              BEQ    OPNL1E   DONE WITH FIRST LOOP
0F37 08                 INX            ADVANCE POINTER
0F38 81 22              CMP A  #'"      QUOTE?
0F3A 27 F7              BEQ    OPNLP1   YEP, LOOK AT NEXT CHAR
0F3C 81 2E              CMP A  #'.      DOT?
0F3E 27 07              BEQ    OPNL1E   YEP, ALL DONE
0F40 5D                 TST B          MORE CHARS TO GO?
0F41 27 F0              BEQ    OPNLP1   NOPE, LOOP AROUND
0F43 36                 PSH A          CHAR OK, PUSH IT ON STACK
0F44 5A                 DEC B          DECREMENT COUNTER
0F45 20 EC              BRA    OPNLP1   AND LOOP AROUND
0F47            OPNL1E  EQU    *
0F47 4F                 CLR A          CLEAR A FOR PADDING
0F48 5D                 TST B          NEED TO PAD?
0F49 27 04              BEQ    OPNL1R   NOPE, RETURN
0F4B            OPNL2E  EQU    *
0F4B 36                 PSH A          YEP, PAD WITH ZEROS
0F4C 5A                 DEC B
0F4D 26 FC              BNE    OPNL2E
0F4F            OPNL1R  EQU    *
0F4F D6 26              LDA B  XTMP     MORE TO DO?
0F51 27 05              BEQ    OPNL2R   NOPE
0F53 7F 00 26           CLR    XTMP     YEP, CLEAR XTMP AND LOOP AROUND
0F56 20 DB              BRA    OPNLP1
0F58            OPNL2R  EQU    *
```

```
                * NOW POP OFF INFO FCB
0F58 DE 48              LDX    FLP      POINT TO FCB
0F5A C6 0B              LDA B  #11      CHAR COUNTER
0F5C            OPNLP2  EQU    *
0F5C 32                 PUL A          PULL OFF CHAR
0F5D A7 0E              STA A  14,X     STORE IT
0F5F 09                 DEX            DECR POINTER
0F60 5A                 DEC B          DECR COUNTER
0F61 26 F9              BNE    OPNLP2   AND LOOP UNTIL DONE
                *
0F63 DE 48              LDX    FLP      FILL IN EXTENSION IF NOT PRESENT
0F65 B6 70 8C           LDA A  DOSCWD   (BUT FIRST, FILL IN DRIVE NUM)
0F68 A7 03              STA A  3,X
0F6A 86 01              LDA A  #1       (.TXT)
0F6C BD 71 2D           JSR    DOSTEX
0F6F DE 48              LDX    FLP      OPEN FOR READING
0F71 86 01              LDA A  #1
0F73 BC 11 04           CPX    #OUTFCB  (OR WRITING?)
0F76 26 01              BNE    OPNRED
0F78 4C                 INC A
0F79            OPNRED  EQU    *
0F79 A7 00              STA A  0,X
0F7B BD 78 06           JSR    DOSFNS
0F7E 26 2E              BNE    OPNOER   OPEN ERROR
0F80            OPNDSK  EQU    *
                * FLAG GETC TO READ FROM DISK
                * OR PUTC TO WRITE TO DISK
0F80 DE 48              LDX    FLP
0F82 BC 11 04           CPX    #OUTFCB
0F85 27 08              BEQ    OPODSK
0F87 CE 10 28           LDX    #GTCDSK
0F8A FF 01 43           STX    GETC+1
0F8D            OPNRET  EQU    *
0F8D 0C                 CLC            AND RETURN
0F8E 39                 RTS
0F8F            OPODSK  EQU    *
0F8F CE 10 71           LDX    #PTCDSK
0F92 FF 01 52           STX    PUTC+1
0F95 20 F6              BRA    OPNRET
                *
0F97            OPNTTY  EQU    *
                * RESET GETC TO READ FROM TTY
                * OR PUTC TO WRITE TO TTY
0F97 DE 48              LDX    FLP
0F99 BC 11 04           CPX    #OUTFCB
0F9C 27 08              BEQ    OPOTTY
0F9E CE 0E 90           LDX    #GTCTTY
0FA1 FF 01 43           STX    GETC+1
0FA4 20 0B              BRA    OPNERT   RETURN WITH C-BIT
0FA6            OPOTTY  EQU    *
0FA6 CE 71 12           LDX    #PTCTTY
0FA9 FF 01 52           STX    PUTC+1
0FAC 20 03              BRA    OPNERT
                *
0FAE            OPNOER  EQU    *
0FAE BD 71 3C           JSR    DOSRPT   REPORT ERROR
0FB1            OPNERT  EQU    *
0FB1 0D                 SEC            RETURN WITH C-BIT SET
0FB2 39                 RTS
                *
0FB3            RATFUN  EQU    *
                * (RATOM)
0FB3 21 00              FDB    MAGWRD
0FB5 7E 01 57           JMP    GETSYM   PASS THE BUCK TO GETSYM
                *
0FB8            GNMFUN  EQU    *
0FB8 21 00              FDB    MAGWRD
                * (GENNAME)   RETURNS UNIQUE ATOM OF FORM 'INPABC'
0FBA            GNM1    EQU    *
```

```
0FBA DE 2C          LDX   SPCPTR   SAVE POINTER TO BEGINNING OF NEW NAME
0FBC DF 38          STX   SYMPTR
0FBE FE 10 22       LDX   GNMLAS   GET LAST ATOM NAME USED
0FC1 26 05          BNE   GNM2
0FC3 CE 10 24       LDX   #GNMFIR  NEVER INITIALIZED, USE FIRST
0FC6 20 03          BRA   GNM3
0FC8          GNM2  EQU   *
0FC8 09             DEX             POINT TO CELL
0FC9 EE 00          LDX   CAR,X     POINT TO NAME
0FCB          GNM3  EQU   *
              * COPY TO NAME SPACE
0FCB A6 00          LDA A 0,X
0FCD DF 26          STX   XTMP
0FCF DE 2C          LDX   SPCPTR
0FD1 9C 2E          CPX   ENDPTR
0FD3 2A 47          BPL   GNMBAD
0FD5 A7 00          STA A 0,X
0FD7 08             INX
0FD8 DF 2C          STX   SPCPTR
0FDA 4D             TST A           ALL DONE?
0FDB 27 05          BEQ   GNM4
0FDD DE 26          LDX   XTMP
0FDF 08             INX
0FE0 20 E9          BRA   GNM3      NOPE, LOOP UNTIL NULL BYTE
0FE2          GNM4  EQU   *
0FE2 DE 38          LDX   SYMPTR    POINT TO FIRST VARIABLE CHAR
0FE4 08             INX
0FE5 08             INX
0FE6 08             INX
0FE7          GNM41 EQU   *
0FE7 A6 00          LDA A 0,X
0FE9 26 10          BNE   GNM42
              * UP TO NULL BYTE, SET TO 'A'
0FEB 86 41          LDA A #'A
0FED A7 00          STA A 0,X
0FEF 08             INX
0FF0 9C 2E          CPX   ENDPTR
0FF2 2A 28          BPL   GNMBAD
0FF4 6F 00          CLR   0,X
0FF6 08             INX
0FF7 DF 2C          STX   SPCPTR
0FF9 20 0E          BRA   GNM51     GO CHECK IF REALLY NEW
              *
0FFB          GNM42 EQU   *
0FFB 4C             INC A
0FFC 81 5A          CMP A #'Z       UP TO Z?
0FFE 23 07          BLS   GNM5
1000 86 41          LDA A #'A       YEP, RESET TO AA
1002 A7 00          STA A 0,X
1004 08             INX             GO ON TO NEXT BYTE
1005 20 E0          BRA   GNM41
1007          GNM5  EQU   *
1007 A7 00          STA A 0,X       ALL DONE
1009          GNM51 EQU   *
1009 DE 38          LDX   SYMPTR
100B BD 01 21       JSR   LOOKUP
100E FF 10 22       STX   GNMLAS
1011 09             DEX
1012 EE 00          LDX   CAR,X     IS THIS REALLY NEW?
1014 9C 38          CPX   SYMPTR
1016 26 A2          BNE   GNM1      NOPE, START OVER
1018 FE 10 22       LDX   GNMLAS    YEP, RETURN WITH NEW ATOM
101B 39             RTS
              *
101C          GNMBAD EQU  *
101C CE 10 E6       LDX   #GNMMSG   'NO MORE ROOM FOR TEMP NAME'
101F 7E 01 36       JMP   ERREX
              *
1022 00 00    GNMLAS FDB  0

1024 54       GNMFIR FCC  'TMP'
1027 00              FCB  0
              *
1028          GTCDSK EQU  *
              * RETURN NEXT CHAR FROM DISK
              * RETURN "EOI" ON ANY ERROR
1028 DF 26           STX   XTMP      SAVE X-REG
102A CE 77 40        LDX   #DOSFCB
102D DF 48           STX   FLP       SAVE FOR OPNTTY, ETC.
102F 6D 00           TST   0,X       OPEN FOR READING?
1031 26 1E           BNE   GTCDE2    NOPE, ERROR
1033 A6 02           LDA A 2,X
1035 81 01           CMP A #1
1037 26 18           BNE   GTCDE2    NOPE, ERROR
1039 BD 78 06        JSR   DOSFMS    YEP, GET NEXT CHAR
103C 26 05           BNE   GTCDER    ERROR ON READ
103E 8D 1E           BSR   CASWIT    UPPER<-->LOWER CASE
1040          GTCDRT EQU   *
1040 DE 26           LDX   XTMP      RESTORE X-REG AND RETURN
1042 39              RTS
1043          GTCDER EQU   *
1043 DE 48           LDX   FLP       CLOSE FILE
1045 86 04           LDA A #4
1047 A7 00           STA A 0,X
1049 BD 78 06        JSR   DOSFMS
104C 86 04           LDA A #4        MAXIMUM NUMBER OF 'EOI'S RETURNED
104E B7 10 5D        STA A GTCECT
1051          GTCDE2 EQU   *
1051 7A 10 5D        DEC   GTCECT    TOO MANY 'EOI'S ALREADY RETURNED?
1054 2E 03           BGT   GTCDE3
1056 BD 0F 97        JSR   OPNTTY    YEP, RESET  GETC
1059          GTCDE3 EQU   *
1059 86 04           LDA A #EOI      RETURN "EOI"
105B 20 E3           BRA   GTCDRT
              *
105D 00       GTCECT FCB   0         USED TO RECOVER FROM END OF FILE ERROR
              *
105E          CASWIT EQU   *
              * SWITCH UPPER AND LOWER CASE ***HACK***
105E 81 41           CMP A #'A
1060 25 0E           BLO   CASURT
1062 81 5A           CMP A #'Z
1064 23 08           BLS   CASW2
1066 81 61           CMP A #'a
1068 25 06           BLO   CASURT
106A 81 7A           CMP A #'z
106C 22 02           BHI   CASURT
106E          CASW2  EQU   *
106E 88 20           EOR A #'a-'A
1070          CASURT EQU   *
1070 39              RTS
              *
1071          PTCDSK EQU   *
              * PUT CHAR OUT TO DISK
1071 37              PSH B           SAVE B, X-REG
1072 DF 26           STX   XTMP
1074 CE 11 04        LDX   #OUTFCB
1077 DF 48           STX   FLP       SET UP FLP FOR OPNTTY, ETC
1079 6D 00           TST   0,X       OPEN FOR WRITING?
107B 26 14           BNE   PTCDE2
107D E6 02           LDA B 2,X
107F C1 02           CMP B #2
1081 26 0E           BNE   PTCDE2
1083 8D B9           BSR   CASWIT    UPPER<-->LOWER CASE
1085 BD 78 06        JSR   DOSFMS    OUT GOES THE BYTE
1088 26 04           BNE   PTCDER    OOPS!
108A          PTCDRT EQU   *
108A DE 26           LDX   XTMP      RESTORE X,B-REG
108C 33              PUL B

108D 39              RTS
              *
108E          PTCDER EQU   *
108E BD 71 3C        JSR   DOSRPT    REPORT ERROR
1091          PTCDE2 EQU   *
1091 BD 0E FB        JSR   CLOFN1    CLOSE FILE
1094 20 F4           BRA   PTCDRT    AND RETURN
              *
1096          SUBLS2 EQU   *
1096 6C              FCC   'load'
109A 00              FCB   0
109B 0E B3           FDB   LODFUN
109D 6F              FCC   'open'
10A1 00              FCB   0
10A2 0E DD           FDB   OPNFUN
10A4 6F              FCC   'openo'
10A9 00              FCB   0
10AA 0E F2           FDB   OPOFUN
              *
10AC 00              FCB   0
10AD          NSUBL2 EQU   *
10AD 63              FCC   'close'
10B2 00              FCB   0
10B3 0F 00           FDB   CLSFUN
10B5 63              FCC   'closeo'
10BB 00              FCB   0
10BC 0E F9           FDB   CLOFUN
10BE 72              FCC   'ratom'
10C3 00              FCB   0
10C4 0F B3           FDB   RATFUN
10C6 67              FCC   'genname'
10CD 00              FCB   0
10CE 0F B8           FDB   GNMFUN
              *
10D0 00              FCB   0
              *
10D1 42       OPNEMS FCC   'BAD ARG TO OPEN/LOAD'
10E5 04              FCB   4
10E6 4E       GNMEMS FCC   'NO SPACE FOR NEW GENNAME ATOM'
1103 04              FCB   4
              *
1104          OUTFCB EQU   *
1104 00 00           FDB   0
1106 00 00           FDB   0
1108                 RMB   24


1120 00 00           FDB   0
1122                 RMB   162
              *
11C4          BEGADR EQU   *
                     END
```

```
NO ERROR(S) DETECTED

 SYMBOL TABLE:

ALP    0044   ATMINI 0139   BEGADR 11C4   BEGPTR 0020   CAR    0000
CASW2  106E   CASWIT 105E   CASURT 1070   CDR    0002   CELPTR 003A
CLOFN1 0EFB   CLOFUN 0EF9   CLSF2  0F05   CLSFN1 0F02   CLSFUN 0F00
CMPTM2 0042   CMPTMP 0040   CONATM 0060   CUREVL 006C   DADPTR 0070
DOSRPT 7094   DOSCMD 708C   DOSEOC 7082   DOSFCB 7740   DOSFMS 7806
DOSPCH 709A   DOSRPT 713C   DOSTEX 712D   DOTATM 0050   ENDMEM 00FC
ENDPTR 002E   EOI    0004   EOIATM 0064   ERRBRK 0154   ERREX  0136
EVAL   0100   EVLATM 0118   FLP    0048   FORM   0034   FRECEL 010F
FREPTR 002A   FRMNUM 0148   GCFREE 0074   GCOL   015D   GCTEMP 0072
GETC   0142   GETC2  0EA3   GETC3  0EA0   GETCAR 0163   GETCEL 010C
GETSYM 0157   GFNXTL 0127   GNM1   0FBA   GNM2   0FC8   GNM3   0FCB
GNM4   0FE2   GNM41  0FE7   GNM42  0FFB   GNM5   1007   GNM51  1009
GNMBAD 101C   GNMEMS 10E6   GNMFIR 1024   GNMFUN 0FB8   GNMLAS 1022
GNXTA  0ED3   GNXTL  0124   GTCDE2 1051   GTCDE3 1059   GTCDER 1043
GTCDRT 1040   GTCDSK 1028   GTCECT 105D   GTCTTY 0E90   INBUFF 7115
ISATOM 013F   ISDTPR 013C   ISVAR  0166   LAMATM 0056   LBRKAT 0068
LODFUN 0EB3   LODLUP 0EC1   LODRT  0ED0   LODRT1 0ECD   LOOKUP 0121
LPARAT 004C   LSTAD2 0130   LSTADD 012D   LSTEN0 0115   LSTEND 0133
LSTINI 012A   LSTPTR 0036   MAGURD 2100   NAMPTR 0032   NILATM 004A
NLAMAT 0058   NLP    0046   NSUBAT 005C   NSUBL2 10AD   NUMFRM 0148
NUMINV 0145   NXTCH  7121   OBEGAD 0E90   OPENFL 0F19   OPNDSK 0F80
OPNEMS 10D1   OPNERR 0F13   OPNERT 0FB1   OPNF2  0EE2   OPNFER 0EEF
OPNFUN 0EDD   OPNLIE 0F47   OPNLIR 0F4F   OPNL2E 0F49   OPNL2R 0F58
OPNLP1 0F33   OPNLP2 0F5C   OPNOER 0FAE   OPNRED 0F79   OPNRET 0F8D
OPNTTY 0F97   OPODSK 0FBF   OPOFUN 0EF2   OPOTTY 0FA6   OUTFCB 1104
PCRLF  711E   PEEKC  00F0   POPX   0115   PRINR  0169   PRINT  0106
PROPOP 016C   PROPSH 0160   PTCDE2 1091   PTCDER 108E   PTCDRT 108A
PTCDSK 1071   PTCTTY 7112   PUSHX  0112   PUTC   0151   PUTSYM 014E
QUOATM 0054   RATFUN 0FB3   RBRKAT 006A   READ   0103   RPARAT 004E
RSETAT 0062   RSTFLG 00F1   SETATM 011E   SONPTR 006E   SPCPTR 002C
SPSAVE 003C   SQUTAT 0052   STKFRG 015A   STKPTR 003E   STP1   0026
STP2   0028   STR1   0022   STR2   0024   SUBLS2 1096   SUBLS3 0076
SUBRAT 005A   SYMLST 0030   SYMPTR 0038   TATOM  005E   TOPX   0110
XTMP   0026   XTMP2  0028   ZERO   00FE
```

```
                        OPT     NOG
11E0            OBEGAD  EQU     $11E0     SUBRS2 ENDS AT $11C4
0000            CAR     EQU     0
0002            CDR     EQU     2
0004            EOI     EQU     4
                * DIRECT PAGE STORAGE CELLS
0020                    ORG     $20
0020 13 E5      BEGPTR  FDB     BEGADR    THIS SHOULD BE IN EACH MODULE
0022            STR1    RMB     2
0024            STR2    RMB     2
0026            STP1    RMB     2
0028            STP2    RMB     2
0026            XTMP    EQU     STP1
0028            XTMP2   EQU     STP2
002A            FREPTR  RMB     2
002C            SPCPTR  RMB     2
002E            ENDPTR  RMB     2
0030            SYMLST  RMB     2
0032            NAMPTR  RMB     2
0034            FORM    RMB     2
0036            LSTPTR  RMB     2
0038            SYMPTR  RMB     2
003A            CELPTR  RMB     2
003C            SPSAVE  RMB     2
003E            STKPTR  RMB     2
0040            CMPTMP  RMB     2
0042            CMPTM2  RMB     2
0044            ALP     RMB     2
0046            NLP     RMB     2
0048            FLP     RMB     2
004A            NILATM  RMB     2
004C            LPARAT  RMB     2
004E            RPARAT  RMB     2
0050            DOTATM  RMB     2
0052            SQUTAT  RMB     2
0054            QUOATM  RMB     2
0056            LAMATM  RMB     2
0058            NLAMAT  RMB     2
005A            SUBRAT  RMB     2
005C            MSUBAT  RMB     2
005E            TATOM   RMB     2
0060            CONATM  RMB     2
0062            RSETAT  RMB     2
0064            EOIATM  RMB     2
0066            SUBLS2  RMB     2
0068            LBRKAT  RMB     2
006A            RBRKAT  RMB     2
006C            CUREVL  RMB     2
006E            SONPTR  RMB     2
0070            BADPTR  RMB     2
0072            GCTEMP  RMB     2
0074            GCFREE  RMB     2         NUMBER OF FREE CELLS AFTER LAST GCOL
0076 13 5C              FDB     SUBLS3
                * SINGLE CHAR SLOTS ...
00F0                    ORG     $F0
00F0            PEEKC   RMB     1
00F1            RSTFLO  RMB     1
                * GLOBAL CONSTANTS
00FC                    ORG     $FC
00FC            ENDMEM  RMB     2         INITIALIZED IN LISP
                * ZERO WORD
00FE 00 00      ZERO    FDB     0
                * GLOBAL VECTORS
0100                    ORG     $100
0100            EVAL    RMB     3
0103            READ    RMB     3
0106            PRINT   RMB     3
0109                    RMB     3
010C            GETCEL  RMB     3
010F            FRECEL  RMB     3
0112            PUSHX   RMB     3
0115            POPX    RMB     3
0118            TOPX    RMB     3
011B            EVLATM  RMB     3
011E            SETATM  RMB     3
0121            LOOKUP  RMB     3
0124            GNXTL   RMB     3
0127            GFNXTL  RMB     3
012A            LSTINI  RMB     3
012D            LSTADD  RMB     3
0130            LSTAD2  RMB     3
0133            LSTEND  RMB     3
0115            LSTEN0  EQU     POPX
0136            ERREX   RMB     3
0139            ATMINI  RMB     3
013C            ISDTPR  RMB     3
013F            ISATOM  RMB     3
0142            GETC    RMB     3
0145 7E 12 E5           JMP     NUMINV
0148            NUMFRM  RMB     3
014B            FRMNUM  RMB     3
014E            PUTSYM  RMB     3
0151            PUTC    RMB     3
0154            ERRBRK  RMB     3
0157            GETSYM  RMB     3
015A            STKFRG  RMB     3
015D            GCOL    RMB     3
0160            PROPSH  RMB     3
0163            GETCAR  RMB     3
0166            ISVAR   RMB     3
0169            PRINR   RMB     3
016C            PROPOP  RMB     3
                *
11E0                    ORG     OBEGAD
                *
2100            MAGWRD  EQU     $2100     ('!')
                *
11E0            ATMFUN  EQU     *
11E0 21 00              FDB     MAGWRD
                * (ATOM X) RETURNS TRUE IF X EVALS TO ATOM, NUMBER, OR NIL
11E2 8D 6F              BSR     GNXTA
11E4 BD 01 3F           JSR     ISATOM
11E7 25 03              BCS     FALSE
11E9            TRUE    EQU     *
11E9 DE 5E              LDX     TATOM     RETURN 'T'
11EB 39                 RTS
11EC            FALSE   EQU     *
11EC DE FE              LDX     ZERO      RETURN NIL
11EE 39                 RTS
                *
11EF            NUMFUN  EQU     *
11EF 21 00              FDB     MAGWRD
11F1 8D 60              BSR     GNXTA
11F3 BD 01 3F           JSR     ISATOM
11F6 25 F4              BCS     FALSE
11F8 2C F2              BGE     FALSE
11FA 20 ED              BRA     TRUE
                *
11FC            ADDFUN  EQU     *
11FC 21 00              FDB     MAGWRD
11FE 8D 31              BSR     GNXTNM
1200 DF 48              STX     FLP
1202 8D 2D              BSR     GNXTNM
1204            ADDF2   EQU     *
1204 DF 46              STX     NLP
1206 96 49              LDA A   FLP+1     ADD THEM
1208 9B 47              ADD A   NLP+1
120A 19                 DAA
120B 97 49              STA A   FLP+1
120D 96 48              LDA A   FLP
120F 99 46              ADC A   NLP
1211 19                 DAA
1212 97 48              STA A   FLP
1214 DE 48              LDX     FLP
1216 7E 01 48           JMP     NUMFRM
                *
1219            SUBFUN  EQU     *
1219 21 00              FDB     MAGWRD
121B            SUBFN1  EQU     *
121B 8D 14              BSR     GNXTNM
121D DF 48              STX     FLP
121F 8D 10              BSR     GNXTNM
1221 BD 12 E5           JSR     NUMINV
1224 20 DE              BRA     ADDF2
                *
1226            GTRFUN  EQU     *
1226 21 00              FDB     MAGWRD
1228 8D F1              BSR     SUBFN1
122A BD 01 4B           JSR     FRMNUM
122D 2F B8              BLE     FALSE
122F 20 B8              BRA     TRUE
                *
1231            GNXTNM  EQU     *
1231 DE 44              LDX     ALP
1233 BD 01 24           JSR     GNXTL
1236 DF 44              STX     ALP
1238 DE 26              LDX     XTMP
123A            GNN2    EQU     *
123A BD 01 3F           JSR     ISATOM
123D 25 05              BCS     GNNERR
123F 2C 03              BGE     GNNERR
1241 7E 01 4B           JMP     FRMNUM
1244            GNNERR  EQU     *
1244 DF 34              STX     FORM      SAVE FORM FOR ERROR PRINTOUT
1246 CE 13 C8           LDX     MGNNERS   'BAD ARG, NUMBER REQUIRED'
1249 BD 01 54           JSR     ERRBRK
124C 96 F1              LDA A   RSTFLG    IN A RESET?
124E 27 EA              BEQ     GNN2
1250 DE FE              LDX     ZERO      YEP, USE VALUE OF ZERO
1252 39                 RTS
                *
1253            GNXTA   EQU     *
1253 DE 44              LDX     ALP
1255 BD 01 24           JSR     GNXTL
1258 DF 44              STX     ALP
125A DE 26              LDX     XTMP
125C 39                 RTS
                *
125D            CNWFUN  EQU     *
                * (CONCWHILE (COND ((...) (LIST ...)) (T NOTHING)) )
                * REPEATEDLY EVALS FORM, CONCATS RESULTS TOGETHER.
                * STOPS WHEN RESULT IS EQTATM (I.E. NOTHING)
125D 21 00              FDB     MAGWRD
125F BD 01 2A           JSR     LSTINI    START UP A LIST ON STACK
1262 8D EF              BSR     GNXTA
1264 25 1E              BCS     CNWDN2    NO ARG, RETURN NIL
1266 DF 48              STX     FLP
1268            CNWEVL  EQU     *
1268 BD 01 00           JSR     EVAL      EVAL ARG
126B BD 01 3C           JSR     ISDTPR    IS IT A LIST?
126E 27 07              BEQ     CNWNXT    RESULT IS NIL, RE-EVAL
1270 25 09              BCS     CNWDUN    RESULT IS ATOM, END LIST WITH ATOM
1272 BD 12 95           JSR     LSTCON    RESULT IS DTPR, CONCAT ON
1275 25 0D              BCS     CNWDN2    DOESN'T END WITH NIL, END OF LOOP
1277            CNWNXT  EQU     *
1277 DE 48              LDX     FLP       RE-POINT TO BODY
1279 20 ED              BRA     CNWEVL    AND GO RE-EVAL IT
                *
127B            CNWDUN  EQU     *
127B 9C 64              CPX     EOIATM    "NOTHING"?
127D 27 05              BEQ     CNWDN2    YEP, JUST END THE LIST NORMALLY
127F BD 01 33           JSR     LSTEND    NOPE, END THE LIST WITH THE ATOM
1282 20 03              BRA     CNWDN3
1284            CNWDN2  EQU     *
1284 BD 01 15           JSR     LSTEN0    END THE LIST WITH A NIL
1287            CNWDN3  EQU     *
1287 7E 01 6C           JMP     PROPOP    RETURN WITH THE LIST AS VALUE
                *
128A            LSTFUN  EQU     *
                * (LIST 'A 'B 'C)
128A 21 00              FDB     MAGWRD
128C DE 3E              LDX     STKPTR    CLEAR TOP OF STACK (TO AVOID FREEING LIST)
128E 6F 00              CLR     CAR,X
1290 6F 01              CLR     CAR+1,X
1292 DE 44              LDX     ALP       RETURN WITH ARG LIST
1294 39                 RTS
                *
1295            LSTCON  EQU     *
                * CONCAT A LIST TO LIST AT TOP OF STACK
                * RETURN WITH C-BIT SET IF CDR OF LAST DTPR NON-NIL
1295 DF 34              STX     FORM
1297 BD 01 18           JSR     TOPX      GET POINTER TO LIST
129A 96 34              LDA A   FORM      FILL IN NEW CDR
129C A7 02              STA A   CDR,X
129E 96 35              LDA A   FORM+1
12A0 A7 03              STA A   CDR+1,X
12A2            LSTC2   EQU     *
12A2 DF 28              STX     XTMP2     SAVE POINTER TO LAST CELL
12A4 EE 02              LDX     CDR,X     NOW POINT TO NEXT CELL IN LIST
12A6 BD 01 3C           JSR     ISDTPR    IS IT A DTPR?
12A9 24 F7              BCC     LSTC2     YEP, KEEP LOOKING
12AB 26 01              BNE     LSTC3     SKIP THE CLC IF NON-NIL CDR
12AD 0C                 CLC
12AE            LSTC3   EQU     *
12AE DE 3E              LDX     STKPTR    FILL IN NEW VALUE FOR LIST POINTER
12B0 96 28              LDA A   XTMP2
12B2 A7 00              STA A   CAR,X
12B4 96 29              LDA A   XTMP2+1
12B6 A7 01              STA A   CAR+1,X
12B8 DE 34              LDX     FORM      RESTORE X-REG
12BA 39                 RTS               RETURN WITH C-BIT CORRECT
                *
12BB            EVLSFN  EQU     *
                * (EVALLIST (CDR L))
                *  EVAL ALL ELEMENTS OF ARG (ASSUMED TO BE A LIST)
                *  RETURN WITH VALUE OF LAST ONE
12BB 21 00              FDB     MAGWRD
12BD DE 44              LDX     ALP
12BF BD 01 3C           JSR     ISDTPR
12C2 25 08              BCS     EVLLS2    NOT A DTPR, JUST RETURN THIS
12C4 EE 00              LDX     CAR,X     GET ARG  (SHOULD BE A LIST)
12C6 20 04              BRA     EVLLS2    AND CONTINUE WITH PROGN CODE
                *
12C8            PGNFUN  EQU     *
                * (PROGN A B C D)
                *  EVALS ALL ARGS AND RETURNS THE VALUE OF THE LAST ONE
12C8 21 00              FDB     MAGWRD
12CA DE 44              LDX     ALP
12CC            EVLLS2  EQU     *
12CC DF 48              STX     FLP       INITIALIZE RETURN VALUE
12CE            PGNFN2  EQU     *
12CE BD 01 3C           JSR     ISDTPR
12D1 25 0F              BCS     PGNDUN
12D3 DF 44              STX     ALP
12D5 EE 00              LDX     CAR,X     EVAL CAR
12D7 BD 01 00           JSR     EVAL
```

```
12DA BF 48              STX   FLP
12DC DE 44              LDX   ALP
12DE EE 02              LDX   CDR,X     SCAN TO END OF LIST
12E0 20 EC              BRA   PGNFN2
12E2            PGNDUN  EQU   *
12E2 DE 48              LDX   FLP       RETURN WITH LAST VAL
12E4 39                 RTS
                *
12E5            NUMINV  EQU   *
                * EXPECTS NUMBER IN X-REG (BCD 10'S COMPLEMENT)
                * PRODUCES 10'S COMPLEMENT
12E5 BF 26              STX   XTMP
12E7 27 14              BEQ   NMIVRT    IT IS ZERO, SAVE OUR  BREATH
12E9 B6 99              LDA A #$99      FIRST GET 9'S COMPLEMENT
12EB 16                 TAB
12EC B0 26              SUB B XTMP
12EE 90 27              SUB A XTMP+1
12F0 8B 01              ADD A #1        NOW ADD 1 TO GET 10'S COMPLEMENT
12F2 19                 DAA
12F3 97 27              STA A XTMP+1
12F5 17                 TBA
12F6 89 00              ADC A #0        FIXUP UP HIGH DIGITS
12F8 19                 DAA
12F9 97 26              STA A XTMP
12FB DE 26              LDX   XTMP
12FD            NMIVRT  EQU   *
12FD 39                 RTS
                *
12FE            RSTFUN  EQU   *
12FE 21 00              FDB   NAGURD
1300            RSTFN2  EQU   *
1300 B6 FF              LDA A #-1
1302            RSTFN3  EQU   *
1302 97 F1              STA A RSTFLG
1304            RTNEOI  EQU   *
1304 DE 64              LDX   EOIATM
1306            RTFRTN  EQU   *
1306 39                 RTS
                *
1307            RTBFUN  EQU   *
1307 21 00              FDB   NAGURD
1309 BD 12 53           JSR   GNXTA
130C BD 01 3F           JSR   ISATOM
130F 25 0B              BCS   RTBFN2
1311 2C 0B              BGE   RTBFN2
1313 BD 01 4B           JSR   FRNNUM
1316 2F E8              BLE   RSTFN2    ZERO OR NEG., SAME AS FULL RESET
1318 DF 26              STX   XTMP
131A 96 27              LDA A XTMP+1    SET UP RSTFLG
131C 20 E4              BRA   RSTFN3
                *
131E            RTBFN2  EQU   *
131E 86 CF              LDA A #$FF-'0   FORCE RESET TO CURRENT BREAK LEVEL
1320 20 E0              BRA   RSTFN3
                *
1322            BRKFUN  EQU   *
1322 21 00              FDB   NAGURD
1324 DE FE              LDX   ZERO
1326 DF 34              STX   FORM
1328 CE 13 DB           LDX   #BRKMSG
132B 7E 01 54           JMP   ERRBRK
                *
132E            BTFUN   EQU   *
132E 21 00              FDB   NAGURD
                * (BT NNNN)
1330 7F 00 49           CLR   FLP+1
1333 BD 12 53           JSR   GNXTA     IF ARG IS A NUMBER, RETURN FUN THAT FAR BACK ON CUREVL
1336 BD 01 3F           JSR   ISATOM
1339 25 07              BCS   BTFUN1
133B 2C 05              BGE   BTFUN1
133D BD 01 4B           JSR   FRNNUM
1340 DF 48              STX   FLP
1342            BTFUN1  EQU   *
1342 DE 6C              LDX   CUREVL    PRINT OUT CAR'S OF CUREVL LIST
1344 DF 44              STX   ALP
1346            BTFUN2  EQU   *
1346 BD 12 53           JSR   GNXTA
1349 25 B9              BCS   RTNEOI    RETURN EOI SO NO PRINT OUT
134B 96 49              LDA A FLP+1
134D 8B 99              ADD A #$99      SUBTRACT 1 (BCD-WISE)
134F 19                 DAA
1350 97 49              STA A FLP+1
1352 27 B2              BEQ   BTFRTN
1354 BD 01 63           JSR   GETCAR
1357 BD 01 06           JSR   PRINT
135A 20 EA              BRA   BTFUN2
                *
135C            SUBLS3  EQU   *
135C 61                 FCC   'alom'
1360 00                 FCB   0
1361 11 E0              FDB   ATMFUN
1363 6E                 FCC   'number'
1369 00                 FCB   0
136A 11 EF              FDB   NUMFUN
136C 61                 FCC   'add'
136F 00                 FCB   0
1370 11 FC              FDB   ADDFUN
1372 73                 FCC   'sub'
1375 00                 FCB   0
1376 12 19              FDB   SUBFUN
1378 67                 FCC   'greater'
137F 00                 FCB   0
1380 12 26              FDB   GTRFUN
1382 6C                 FCC   'list'
1386 00                 FCB   0
1387 12 8A              FDB   LSTFUN
1389 65                 FCC   'evallist'
1391 00                 FCB   0
1392 12 BB              FDB   EVLSFN
1394 72                 FCC   'retbrk'
139A 00                 FCB   0
139B 13 07              FDB   RTBFUN
139B 62                 FCC   'bl'
139F 00                 FCB   0
13A0 13 2E              FDB   BTFUN
                *
13A2 00                 FCB   0
13A3            NSUBL3  EQU   *
13A3 70                 FCC   'progn'
13A8 00                 FCB   0
13A9 12 C8              FDB   PGNFUN
13AB 63                 FCC   'concwhile'
13B4 00                 FCB   0
13B5 12 5B              FDB   CNWFUN
13B7 72                 FCC   'reset'
13BC 00                 FCB   0
13BD 12 FE              FDB   RSTFUN
13BF 62                 FCC   'break'
13C4 00                 FCB   0
13C5 13 22              FDB   BRKFUN
                *
13C7 00                 FCB   0
13C8 4E         GNNEMS  FCC   'NUMBER REQUIRED'
13D7 04                 FCB   4
13D8 42         BRKMSG  FCC   'BREAKING ...'
13E4 04                 FCB   4
                *
13E5            BEGADR  EQU   *
                        END
```


NO ERROR(S) DETECTED

SYMBOL TABLE:

```
ADDF2  1204   ADDFUN 11FC   ALP    0044   ATMFUN 11E0   ATMINI 0139
BEGADR 13E5   BEGPTR 0020   BRKFUN 1322   BRKMSG 13DB   BTFRTN 1306
BTFUN  132E   BTFUN1 1342   BTFUN2 1346   CAR    0000   CDR    0002
CELPTR 003A   CMPTM2 0042   CMPTMP 0040   CNWDN2 1284   CNWDN3 1287
CNWDUN 127B   CNWEVL 1268   CNWFUN 125D   CNWNXT 1277   CONATM 0060
CUREVL 006C   DADPTR 0070   DOTATM 0050   ENDMEM 00FC   ENDPTR 002E
EOI    0004   EOIATM 0064   ERRBRK 0154   ERREX  0136   EVAL   0100
EVLATM 0118   EVLLS2 12CC   EVLSFN 12BB   FALSE  11EC   FLP    0048
FORM   0034   FRECEL 010F   FREPTR 002A   FRNNUM 014B   GCFREE 0074
GCOL   015D   GCTEMP 0072   GETC   0142   GETCAR 0163   GETCEL 010C
GETSYM 0157   GFNXTL 0127   GNN2   123A   GNNEMS 13C8   GNNERR 1244
GNXTA  1253   GNXTL  0124   GNXTNM 1231   GTRFUN 1226   ISATOM 013F
ISDIFR 013C   ISVAR  0166   LANATM 0056   LBRKAT 0048   LOOKUP 0121

LPARAT 004C   LSTAD2 0130   LSTADD 012D   LSTC2  12A2   LSTC3  12AE
LSTCON 1295   LSTEN0 0115   LSTEND 0133   LSTFUN 128A   LSTINI 012A
LSTPTR 0036   NAGURD 2100   NAMPTR 0032   NILATM 004A   NLANAT 0058
NLP    0046   NMIVRT 12FD   NSUBAT 005C   NSUBL3 13A3   NUMFRM 0148
NUMFUN 11EF   NUMINV 12E5   OBEGAD 11E0   PEEKC  00F0   PGNDUN 12E2
PGNFN2 12CE   PGNFUN 12C8   POPX   0115   PRINR  0169   PRINT  0106
PROFOP 016C   PROPSH 0160   PUSHX  0112   PUTC   0151   PUTSYM 014E
QUOATM 0054   RBRKAT 006A   READ   0103   RPARAT 004E   RSETAT 0062
RSTFLG 00F1   RSTFN2 1300   RSTFN3 1302   RSTFUN 12FE   RTBFN2 131E
RTBFUN 1307   RTNEOI 1304   SETATM 011E   SONPTR 006E   SPCPTR 002C
SPSAVE 003C   SQUTAT 0052   STKFRG 015A   STKPTR 003E   STP1   0026
STP2   0028   STR1   0022   STR2   0024   SUBFN1 121B   SUBFUN 1219
SUBLS2 0066   SUBLS3 135C   SUBRAT 005A   SYNLST 0030   SYNPTR 0038
TATOM  005E   TOPX   0118   TRUE   11E9   XTMP   0026   XTMP2  0028
ZERO   00FE
```