This document was produced by SDC and III in performance of contract SD-97 and subcontract 65-107.

TM- 2710/330/00

AUTHOR
E. Book, SDC

TECHNICAL
C. Weissman, SDC

RELEASE
S. L. Kameny, SDC
for D. L. Drukey, SDC

DATE 2 November 1965 PAGE 1 OF 11 PAGES

AUTHOR DELIVERED

TECH MEMO

a working paper

System Development Corporation / 2500 Colorado Avenue / Santa Monica, California 90406
Information International Inc. / 200 Sixth Street / Cambridge, Massachusetts 02142

The LISP Version of the Meta Compiler

ABSTRACT

This paper describes a meta-compiler program which processes a BNF-like language and produces a LISP II intermediate language program. The program produced is a syntax translator.

The version of the compiler described here exists as a LISP 1.5 program and operates on Q-32 LISP 1.5. It will produce itself as a LISP II intermediate language program.

The work reported herein is based upon the accomplishments of Val Schorre and Lee Schmidt, of the Los Angeles Chapter of ACM, SIGPLAN Working Group I.

1. INTRODUCTION

The LISP II programming language is to be processed into LISP II intermediate language by a syntax-directed compiler. This compiler is to be produced by using a meta-compiler. This document describes the Meta Compiler. The technique is based on the work done by Working Group I of the Los Angeles Chapter of ACM.

The Meta Compiler is a model of a machine with an input tape and a push down accumulator; the accumulator is referred to as the star stack and is symbolized by *. The compiler also has a true/false indicator cell called SIGNAL. The Meta Compiler translates a program written in its input language, which resembles BNF* with extensions, into a tree structure. This tree structure is a LISP II intermediate language program. The program so translated is usually referred to as a compiler. The Meta Compiler used here is itself a Q-32 LISP 1.5 program.

A meta-language program is organized into a body of rules. Each rule corresponds to a syntax equation of BNF. A Rosetta paper follows:

BNF	META	Meaning
⟨something⟩	SOMETHING	Meta-linguistic variables
A	'A'	Terminal character or string
|	/	Alternation
{ }	( )	Meta-linguistic parentheses

Writing two entities side by side (such as AB) means that an A is followed by a B. The ending symbol of a rule in LISP-META is the semicolon. BNF has no ending symbol for its syntax equations.

Identifiers are meta-linguistic variables, to wit, other definitions. They may also be the names of subroutines. If an identifier is followed by square brackets, the identifier then is the name of a routine to be executed. Its parameters are enclosed by the brackets and are separated by commas. Strings are groups of characters enclosed in primes. These correspond to terminal characters. If a prime is to be used within a string, two primes are written.

The remainder of this document is organized as follows:

A description of the various routines which are not defined in the syntax equation of the Meta Compiler, appears first. If the meta language refers to these routines by other than their names, the encoding is also shown. The next section is an English-language description of selected equations. A listing of the Meta Compiler, written in its own language, appears last.

---

*Backus Naur Form.

2. SYNTACTIC ROUTINES

These routines are principally concerned with asking questions about the characters on the input tape and what to do with them.

2.1 META LANGUAGE: 'CHARACTERS'

Routine: CMPR string (Compare)

Meaning:

When a string is written in a syntax equation it means: "If the next group of characters on the input tape matches the exhibited string, move the read past those characters and report true. Otherwise report false and do not move the read head".

2.2 META LANGUAGE: + 'CHARACTERS'

Routine: COMPS string (Compare and store)

Meaning:

This expression has the same effect as is presented in Section 2.1, except that if the answer is true the matched characters are put into the accumulator (*).

2.3 META LANGUAGE: - 'CHARACTERS'

Routine: NCOMP string (No compare)

Meaning:

If the next characters on the input tape match the exhibited string, report false. If there is no match, report true. However, do not move the read head in either event.

2.4 META LANGUAGE: ↑ 'CHARACTERS'

Routine: CMPR2 string

Meaning:

If the next characters on the input tape match the exhibited string, make a token (atom) out of the characters, push the token into the accumulator, move the read head past those characters, and report true. Otherwise, report false and do not move the read head.

2.5 META LANGUAGE: ..

Routine: MARK

Meaning:

Syntax equations which have a double period instead of an equal sign are used to collect characters and to make tokens out of them. The routine MARK is executed when these syntax equations are entered. MARK skips blanks on the input tape and stops at the first non-blank character. The routine then sets the skip blanks flag to "off" so that blanks become significant to all routines which look at characters on the input tape; that is, the routines do not bypass leading blanks while this flag is off. MARK then sets a mark in the accumulator so that all characters put into the accumulator on top of this mark will be collected as one token in a first-in first-out manner.

2.6 META LANGUAGE: ; (AT THE END OF .. EQUATION)

Routine: TOKEN

Meaning:

TOKEN collects all characters, starting with the character above the mark and going to the top of the accumulator. These characters are formed into a token. TOKEN then sets the skip blanks flag to"on,"so that routines which look at the input tape characters ignore leading blanks.

2.7 META LANGUAGE: ANY

Routine: ANY

Meaning:

Put the next character on the input tape into the accumulator.

2.8 META LANGUAGE: DELETE

Routine: DELETE

Meaning:

Skip the next character on the input tape.

2.9 META LANGUAGE: $

Routine: $

Meaning:

Recognize zero or more of the following syntactic entities.

3. SEMANTIC ROUTINES

These routines are concerned with building up the tree structure which reflects the parse of the syntax.

3.1 META LANGUAGE: ⟨

Routine: FLAG

Meaning:

Set a flag in the accumulator so that a sub-tree will be formed out of the tokens and expressions collected until SEQ (⟩) is executed.

3.2 META LANGUAGE: ⟩

Routine: SEQ

Meaning:

Completes the formation of a sub-tree out of whatever has been collected since FLAG was executed.

3.3 META LANGUAGE: *n

Routine: STARn

Meaning:

STARn produces the nth element of the accumulator and removes it from the accumulator.

3.4 META LANGUAGE: +*n

Routine: STARnP

Meaning:

Copies the nth element of the accumulator onto the top of the accumulator without removing it.

3.5 META LANGUAGE: *[and $[

Routine: PUSH parameter

Meaning:

Creates a list or node out of the parameters of *[or $[and leaves it on top of the accumulator (*).

3.6 META LANGUAGE: , 'CHARACTERS'

Routine: INSERT string

Meaning:

Push the string of characters shown in the syntax equation into the accumulator.

3.7 META LANGUAGE: ↑ IDENTIFIER

Routine: LOAD x

Meaning:

Push the identifier into the stack.

3.8 META LANGUAGE: GN1 or GN2

Routine: GN1 or GN2

Meaning:

The GN1 and GN2 routines are concerned with obtaining labels for transfer points. They manipulate a stack called GEN, which is organized into pairs. The first element of each pair concerns the GN1 routine; the second element of each pair concerns the GN2 routines. If the first (second) element of the top pair is empty, a symbol is generated and put there. The first (second) element of the top pair is always produced as output.

3.9 META LANGUAGE: GEN1 or GEN2

Routine: GEN1 or GEN2

Meaning:

Push the output of GN1 (GN2) into the accumulator.

3.10 META LANGUAGE: MAKEATOM

Routine: MAKEATOM

Meaning:

Replace the string of characters on top of the accumulator by an atom with the same print name.

3.11 META LANGUAGE: MAKENUMBER

Routine: MAKENUMBER

Meaning:

Replace the string of digits on top of the accumulator with its integer value.

4. BACKUP ROUTINES

If more than one syntax equation or alternative start with the same construct, there is a possibility that an ambiguous situation will arise where backup over that first construct must occur in order to go on with the parsing. In order to accomplish the backup, the state of the machine must be saved and restored at critical places. Six routines, a stack called BACK and one called NAME are used to attempt to recover from ambiguous situations.

4.1 META LANGUAGE: RPT1

Routine: RPT1

Meaning:

This routine is invoked at the top of a loop set up by the sequence operation ($). It increments a cell called BACKUP-COUNT.

4.2 META LANGUAGE: RPT2

Routine: RPT2

Meaning:

This routine is invoked at the bottom of a loop set up by the sequence operation ($). It decrements a cell called BACKUP-COUNT. Whenever this cell is greater than zero, nothing is saved and backup does not take place.

4.3 META LANGUAGE: ENTER X

Routine: ENTER

Meaning:

This routine is used upon entering a syntax equation. The name of the syntax equation being entered is saved on a NAME list. A "blip" is pushed into the top of BACK for constructs which are collected by this syntax equation. A blip is an empty list and is used to collect information.

The GEN stack has two blips pushed into it, in case generated labels are needed by this syntax equation. Then the next entity on the INPUT tape is examined. If it is the name of the routine being entered, that name is removed from the input tape and SIGNAL is set to true.

4.4 META LANGUAGE: LEAVE

Routine: LEAVE

Meaning:

Invoked upon leaving a syntax equation. The GEN stack has its top two elements popped off. If the BACKUP-COUNT cell is zero, then SIGNAL is checked. If SIGNAL is true, then the name of the routine being left is put on top of BACK. Otherwise the top element of BACK is popped. In any event, the top element of NAME is popped.

4.5 META LANGUAGE: SAVER

Routine: SAVER

Meaning:

This routine is called at the beginning of an expression and it merely pushes a blip into BACK. The blip gets filled by all constructs which the expression collects.

4.6 META LANGUAGE: RSTOR

Routine: RSTOR

Meaning:

This routine is called at the end of expressions. If SIGNAL is true, it takes the constructs which have been collected by the expression just processed, and groups them with the constructs being collected by the next higher level expression, which may be a syntax equation. Otherwise it puts those constructs on the INPUT tape. In either event BACK is popped.

Now it is possible to examine Figure 1, which shows the Meta Compiler, and get an approximate idea of what it does. A few sample syntax equations are followed through; the remainder of the equations can be used as exercises for the interested reader. The extreme left column shows line numbers and is not part of the syntax equations.

```
4500  ID .. LET $(LET / DGT / '-' ,'.') MAKEATOM;
```

This is the definition of an identifier. The double dot shows we are forming a token and are going to skip leading blanks on the input tape. When the first non-blank character is encountered, the SKIP-BLANKS flag is set to false. We then transfer to a routine called LET which sees whether the next character on the input tape is a letter. If not, we exit ID false. If it is, we look for a sequence, which may be empty of LET letters or DGT digits or minus signs, which represent hyphens. By examining the LET and DGT equations, we see that if

```
0000100-.META
0000200-SYNTAX = '.META'
0000300-      < $(RULES / -'.FINISH' GOBBLE) '.FINISH' > COMPILE ;
0000400-RULE = ID < $[ENTER, $[QUOTE, *1]]
0000500-        $[IF,SIGNAL ,$[GO ,GN1[]]]
0000600-      ('=' EXPR /
0000700-      '.:' $[MARK] EXPR $[TOKEN] )
0000800-      ';' GEN1 $[LEAVE] >
0000900-      $[FUNCTION, *2,NIL,*[BLOCK,NIL,*1]];
0001000-EXPR = $[SAVER] SUBEXP
0001100-      $('/' $[IF, SIGNAL, $[GO, GN1[] ]]
0001200-      SUBEXP) GEN1 $[RSTOR] ;
0001300-SUBEXP = $(TESTS
0001400-      $[IF, $[NOT, SIGNAL], $[GO, GN1[] ]] /
0001500-      ACTION) GEN1;
0001600-TESTS =
0001700-      ID ('[' PARAMSQ ']' *[ *2, *1] / .EMPTY) /
0001800-      STRING[ $[CMPR, *1] /
0001900-      '*' STRING1 $[COMPS, *1] /
0002000-      '-' STRING1 $[NCOMP, *1] /
0002100-      '↑' (ID $[LOAD, $[QUOTE, *1]] /
0002200-      STRING1 $[CMPR2, *1] ) /
0002300-      '(' EXPR ')' ;
0002400-ACTION = '.EMPTY' $[SET, SIGNAL, TRUE] /
0002500-      ',' STRING1 $[INSERT, *1] /
0002600-      '$[' PARAMSQ ']' $[PUSH, *[LIST, *1]] /
0002700-      '*[' PARAMSQ1 ']' $[PUSH, *1] /
0002800-      '<' $[FLAG] / '>' $[SEQ] /
0002900-      REPEAT;
0003000-PARAM = ID('[' PARAMSQ ']' *[ *2, *1] /
0003100-      .EMPTY $[LOAD, $[QUOTE, *1]]) /
0003200-      '*1' $[STAR1] / '*2' $[STAR2] / '*3' $[STAR3] /
0003300-      '+*[' $[CAR, STAR] / '+*2' $[CADR, STAR] /
0003400-      '+*3' $[CADDR, STAR] /
0003500-      '$[' PARAMSQ ']' *[LIST, *1] /
0003600-      '*[' PARAMSQ1 ']' /
0003700-      NUM / STRING1 ;
0003800-PARAMSQ = < (PARAM $(',' PARAM) / .EMPTY) > ;
0003900-PARAMSQ1 = PARAM (',' PARAMSQ1
0004000-      $[CONS, *2, *1] / .EMPTY) ;
0004100-REPEAT = -'$[' '$' $[RPT1] GEN1 TESTS
0004200-      $[IF, SIGNAL, $[GO, GN1[] ]] $[RPT2] ;
0004300-GOBBLE = ERROR $( -';' DELETE) ';' ;
0004400-STRING1 .. '''' $ (-'''' ANY /'''''' ,'''') '''' ;
0004500-ID .. LET $(LET / DGT / '-' ,':') MAKEATOM;
0004600-NUM .. DGT $ DGT MAKENUMBER;
0004700-LET = +'A' / +'B' / +'C' / +'D' / +'E' / +'F' / +'G' /
0004800-      +'H' / +'I' / +'J' / +'K' / +'L' / +'M' /
0004900-      +'N' / +'O' / +'P' / +'Q' / +'R' / +'S' /
0005000-      +'T' / +'U' / +'V' / +'W' / +'X' / +'Y' / +'Z' ;
0005100-DGT = +'0' / +'1' / +'2' / +'3' / +'4' /
0005200-      +'5' / +'6' / +'7' / +'8' / +'9' ;
0005300-.FINISH
```

Figure 1. A Listing of the LISP Meta Compiler

an appropriate character is recognized it is pushed into the accumulator, *, because of the + before each string. However, the ID equation indicates that the minus, if recognized, is not put into the accumulator. The comma followed by the string period, '.' shows that a period is inserted instead. After processing such a sequence of characters, a routine called MAKEATOM is called. This subroutine takes all the characters collected as an identifier token and makes an atom of them.

```
200  SYNTAX = '.META'
          < $(RULES / -'.FINISH' GOBBLE) '.FINISH' > COMPILE ;
```

This equation defines a meta-language program as starting with .META. A flag is set up so that the entire program will be collected as one list. Then we encounter zero or more of RULES. If we do not find the characters .FINISH, we call GOBBLE (4300). A cursory glance at GOBBLE shows that it goes to ERROR and then reads the input tape deleting characters until it finds a semi-colon, at which time it throws that away also and returns to SYNTAX. If we do find .FINISH, we close the list which contains the parse of the program being defined. Then we go to the LISP compiler.

It is hoped that the availability of a meta compiler in LISP will make it possible to produce the syntax translator for LISP II to intermediate language more easily. A meta compiler should also facilitate the processing of modifications and improvements to the LISP II source language.

The method of production used was to take the already existing Meta Compiler on the Q-32 and make it produce LISP 1.5 output. By the well known bootstrap cannibalism it reproduced itself as a LISP 1.5 program. The syntax equations are modified to produce LISP II intermediate language and the cannibal eats again.

The definition of META and LISPX written in META XI


```
0000100-.META (META PROGRAM)
0000200-SYNTAX = '.META' ( '(' ID ID ')'
0000300-      .[DEFINE, .[ .[ .[ *2,
0000400-      .[LAMBDA, .[X, Y], .[COMPLETE, .[INITIALIZE, X, Y],
0000500-      .[*1] ]] ]]] ] COMPILE / .EMPTY)
0000600-      $(RULE\ -'.FINISH' GOBBLE)
0000700-      '.FINISH' ;
0000800-RULE = ID ( '=' EXPR /
0000900-      '.,' EXPR .[TOKEN, .[MARK], *1] ) ';'
0001000-      .[DEFINE, .[ .[ .[ *2,
0001100-      .[LAMBDA, NIL, .[LEAVE, .[ENTER], *1]] ]]] ] COMPILE ;
0001200-EXPR = SUBEXP ( EXPR1 <$ EXPR1>
0001300-      .[AND, .[OR, *4, *3, *2, -[*1]-], OK] /
0001400-      '\' .[BACKUP, *1] <SUBEXP $ EXPR2>
0001500-      .[RESTORE, .[SAVER], .[AND, .[OR, *2, -[*1]-], OK]]
0001600-      / .EMPTY) ;
0001700-EXPR1 = '/' .[NOT, OK] SUBEXP;
0001800-EXPR2 = '\' .[BACKUP, *1]  SUBEXP;
0001900-SUBEXP = TESTS ( BACKTEST < $ BACKTEST >
0002000-      .[AND, *3, *2, -[*1]-] / .EMPTY) ;
0002100-BACKTEST = TESTS  .[OR, *1, .[SETQ, OK, F]];
0002200-TESTS =
0002300-      ID (PARAMSQ .[*2, -[*1]- ] / .EMPTY .[*1] ) /
0002400-      STRING1 .[CMPR, *1] /
0002500-      '+' STRING1 .[COMPS, *1] /
0002600-      '-' STRING1 .[NCOMP, *1] /
0002700-      '↑' (ID .[LOAD, .[QUOTE, *1]] /
0002800-      STRING1 .[CMPR2, *1] ) /
0002900-      '<'  EXPR '>' .[SEQ, .[FLAGS], *1] /
0003000-      '(' EXPR ')' / '.EMPTY' ↑TRUE /
0003100-      ';' STRING1 .[INSERT, *1] /
0003200-      LISTX .[LOAD, *1] /
0003300-      '.(' <$OUTPUT> .[PROG, NIL, -[*1]-, .[RETURN, T]] ')' /
0003400-      REPEAT ;
0003500-OUTPUT = STRING1 .[PRINST, *1] /
0003600-      ID ( PARAMSQ .[PRIN0, .[*2, -[*1]-]] /
0003700-      .EMPTY .[PRIN0, *1] ) /
0003800-      STACK .[PRIN0, *1] / '/' .[TERPRI] ;
0003900-REPEAT = '$' TESTS
0004000-      .[PROG, NIL, GN1[],
0004100-      .[COND, .[*1, .[GO, GN1[] ]]],
0004200-      .[RETURN, OK]];
0004300-STACK = '*1' .[STAR1] / '*2' .[STAR2] /
0004400-      '*3' .[STAR3] / '*4' .[STAR4] /
0004500-      '+*1' .[CAR,STAR] / '+*2' .[CADR, STAR] /
0004600-      '+*3' .[CADDR, STAR] ;
0004700-PARAMSQ = '[' < (EXPNQ $(',' EXPNQ ) /
0004800-      .EMPTY ) > ']' ;
0004900-GOBBLE = ERRORX $(-';' DELETEX) ';' ;
0005000-STRING1 .. '''' $ (-'''' ANY / '''''' , '''')
0005100-      '''' .[QUOTE, *1] ;
0005200-ID .. LET $(LET / DGT / '-' , '*') MAKEATOM ;
0005300-NUM .. DGT $ DGT MAKENUMBER ;
0005400-LET = ISIT[LETTER , T];
0005500-DGT = ISIT [DIGIT, T];
0005600-CHARACTER .. '#' ANY MAKECHR ;
```

The definition of META and LISPX written in META II


```
0005700-ELEMENT = NUM / STRING1 / CHARACTER / IDENT / STACK /
0005800-      LISTX / '(' EXPNQ ')' ;
0005900-IDENT = ID (
0006000-      PARAMSQ .[*2, -[*1]- ] /
0006100-      (':[' EXPNQ ']'
0006200-      (':=' EXPNQ .[DEFLIST,
0006300-      .[LIST, .[LIST, *2, *1]], .[QUOTE, *1]] /
0006400-      .EMPTY .[GET, *1, .[QUOTE, *1]] )) /
0006500-      ':=' EXPNQ .[SETQ, *2, *1] /
0006600-      (SETQ[ID-V, ID-V] .[QUOTE, *1] / .EMPTY )) ;
0006700-LIST-SEQ = '-[' EXPNQ ']-'
0006800-      (',' LIST-SEQ .[APPEND, *2, *1] / .EMPTY)/
0006900-      EXPQ (',' LIST-SEQ / .EMPTY .[] )
0007000-      .[CONS, *2, *1] ;
0007100-LISTX = '.[' (LIST-SEQ / .EMPTY .[] ) ']' ;
0007200-EXPQ = WHERE [ FUNCTION[EXPX], T] ;
0007300-EXPNQ = WHERE [ FUNCTION[EXPX], F] ;
0007400-LISTEXP = ELEMENT $('.1' .[CAR, *1] /
0007500-      '.2' .[CADR, *1] / '.3' .[CADDR, *1] /
0007600-      '.4' .[CADDDR, *1] / ':2' .[CDR, *1] /
0007700-      ':3' .[CDDR, *1] / ':4' .[CDDDR, *1] ) ;
0007800-BASIC = LISTEXP ( '*' BASIC .[CONS, *2, *1] /
0007900-      .EMPTY);
0008000-RELATION = BASIC ('=' BASIC .[EQUAL, *2, *1] /
0008100-      '-=' BASIC .[NOT, .[EQUAL, *2, *1]] /
0008200-      .EMPTY) ;
0008300-NEGATION = '-' RELATION .[NOT, *1] /
0008400-      RELATION;
0008500-FACTOR = NEGATION ('.A.' FACTOR
0008600-      .[AND, *2, *1] / .EMPTY) ;
0008700-EXPX = FACTOR ( '.V.' EXPX
0008800-      .[OR, *2, *1] / .EMPTY) ;
0008900-LOOPST = '.LOOP' 'UNTIL' GEN1
0009000-      (EXPNQ \ ERRORX $(-'.BEGIN' DELETEX))
0009100-      '.BEGIN' .[COND, .[*1, .[GO, GN2[] ]]]
0009200-      $(ST\ -'.END' GOBBLE)
0009300-      '.END' .[GO, GN1[] ] GEN2 ;
0009400-IFST = '.IF' (EXPNQ \ ERRORX $(-'.BEGIN' DELETEX))
0009500-      '.BEGIN' ('.THEN' / .EMPTY)
0009600-      .[COND, .[ .[NOT, *1], .[GO, GN1[] ]]]
0009700-      $ ST
0009800-      ('.ELSE' .[GO, GN2[] ] GEN1
0009900-      $(ST \ -'.END' GOBBLE)
0010000-      '.END' GEN2 / '.END' GEN1) ;
0010100-PRINTST = '.PRINT' $ OUTPUT ';' ;
0010200-ST = EXPNQ ';' / LOOPST / IFST / PRINTST ;
0010300-IDSEQ = '[' < (FORMAL $( ',' FORMAL) / .EMPTY) ']' > ;
0010400-FORMAL = ID / '.LOC' ID ;
0010500-PROCEDURE = '.PROCEDURE' ID IDSEQ ';'
0010600-      ('.LOCAL' IDSEQ ';' / .EMPTY .[] )
0010700-      < $(ST \ -'.RETURN' GOBBLE)
0010800-      '.RETURN' ( '[' EXPNQ .[RETURN, *1] ']' / .EMPTY) ';' >
0010900-      .[DEFINE, .[ .[ .[ *4,
0011000-      .[LAMBDA, *3, .[PROG, *2, -[*1]- ]]] ]]]  COMPILE ;
0011100-LISP-DIVISION = '.LISPX' $ PROCEDURE '.FINISH' ;
0011200-FLUID-DECLARATION = '.DECLARE' '['
0011300-      FLUID1 $(',' FLUID1) ']' ';' ;
0011400-FLUID1 = ID .[CSET, .[*1, NIL]] COMPILE;
0011500-PROGRAM = $(SYNTAX / LISP-DIVISION /
0011600-      FLUID-DECLARATION) '.STOP' ;
0011700-.FINISH
0011800-.STOP
```

The definition of META and 215 PA written in 215P/L

(Translated from version written in META)

```
(1 DEFINE (((META (LAMBDA (X Y)
     (COMPLETE (INITIALIZE X Y) (PROGRAM))))))
(1 DEFINE (((SYNTAX (LAMBDA NIL (LEAVE (ENTER)
       (AND (CMPR (QUOTE ('. 'M 'E 'T 'A)))
        (OR (AND (CR (AND (CMPR (QUOTE ('())))
             (CR (ID) (SETQ OK F))
             (CR (ID) (SETQ OK F))
             (CR (CMPR (QUOTE (')))) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE DEFINE)
              (CONS (CONS (CONS (CONS (STAR2)
                (CONS (CONS (QUOTE LAMBDA)
                 (CONS (CONS (QUOTE X) (CONS (QUOTE Y) NIL))
                  (CONS (CONS (QUOTE COMPLETE)
                   (CONS (CONS (QUOTE INITIALIZE)
                    (CONS (QUOTE X) (CONS (QUOTE Y) NIL)))
                   (CONS (CONS (STAR1) NIL) NIL))) NIL))) NIL))
                NIL) NIL) NIL))) (SETQ OK F))
          (OR (COMPILE) (SETQ OK F))) (NOT OK) TRUE) OK)
         (SETQ OK F))
        (OR (PROG NIL M00001 (COND ((RESTORE (SAVER)
         (AND (OR (BACKUP (RULE))
          (AND (NCOMP (QUOTE ('. 'F 'I 'N 'I 'S 'H)))
           (OR (GOBBLE) (SETQ OK F)))) OK)) (GO M00001)))
       (RETURN OK)) (SETQ OK F))
      (OR (CMPR (QUOTE ('. 'F 'I 'N 'I 'S 'H))) (SETQ OK F))))))))))
(1 DEFINE (((RULE (LAMBDA NIL (LEAVE (ENTER)
      (AND (ID)
       (OR (AND (CR (AND (CMPR (QUOTE ('=)))
           (OR (EXPR) (SETQ OK F)))
          (NOT OK)
          (AND (CMPR (QUOTE ('. '.)))
           (OR (EXPR) (SETQ OK F))
           (OR (LOAD (CONS (QUOTE TOKEN)
             (CONS (CONS (QUOTE MARK) NIL) (CONS (STAR1) NIL))))
            (SETQ OK F)))) OK) (SETQ OK F))
       (OR (CMPR (QUOTE (' ))) (SETQ OK F))
       (OR (LOAD (CONS (QUOTE DEFINE)
          (CONS (CONS (CONS (CONS (STAR2)
             (CONS (CONS (QUOTE LAMBDA)
              (CONS (QUOTE NIL)
               (CONS (CONS (QUOTE LEAVE)
                (CONS (CONS (QUOTE ENTER) NIL)
                 (CONS (STAR1) NIL))) NIL))) NIL)) NIL) NIL) NIL))
        ) (SETQ OK F)) (OR (COMPILE) (SETQ OK F))))))))
(1 DEFINE (((EXPR (LAMBDA NIL (LEAVE (ENTER)
      (AND (SUBEXP)
       (OR (AND (CR (AND (EXPR1)
           (OR (SEQ (FLAGS)
             (PROG NIL M00002 (COND ((EXPR1) (GO M00002)))
              (RETURN OK))) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE AND)
              (CONS (CONS (QUOTE OR)
               (CONS (STAR4)
                (CONS (STAR3) (CONS (STAR2) (STAR1)))))
               (CONS (QUOTE OK) NIL)))) (SETQ OK F)))
          (NOT OK)
          (AND (CMPR (QUOTE (' )))
           (OR (LOAD (CONS (QUOTE BACKUP) (CONS (STAR1) NIL)))
            (SETQ OK F))
           (OR (SEQ (FLAGS)
             (AND (SUBEXP)
              (OR (PROG NIL M00003 (COND ((EXPR2) (GO M00003)))
```

```
                (RETURN OK)) (SETQ OK F)))) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE RESTORE)
                (CONS (CONS (QUOTE SAVER) NIL)
                 (CONS (CONS (QUOTE AND)
                   (CONS (CONS (QUOTE OR) (CONS (STAR2) (STAR1)))
                    (CONS (QUOTE OK) NIL))) NIL)))) (SETQ OK F)))
          (NOT OK) TRUE) OK) (SETQ OK F)))))))))
(1 DEFINE (((EXPR1 (LAMBDA NIL (LEAVE (ENTER)
       (AND (CMPR (QUOTE ('/)))
        (OR (LOAD (CONS (QUOTE NOT) (CONS (QUOTE OK) NIL)))
        (SETQ OK F)) (OR (SUBEXP) (SETQ OK F)))))))))
(1 DEFINE (((EXPR2 (LAMBDA NIL (LEAVE (ENTER)
       (AND (CMPR (QUOTE (' )))
        (OR (LOAD (CONS (QUOTE BACKUP) (CONS (STAR1) NIL)))
        (SETQ OK F)) (OR (SUBEXP) (SETQ OK F)))))))))
(1 DEFINE (((SUBEXP (LAMBDA NIL (LEAVE (ENTER)
       (AND (TESTS)
        (OR (AND (CR (AND (BACKTEST)
            (OR (SEQ (FLAGS)
              (PROG NIL M00004 (COND ((BACKTEST) (GO M00004)))
               (RETURN OK))) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE AND)
              (CONS (STAR3) (CONS (STAR2) (STAR1))))) (SETQ OK F)))
          (NOT OK) TRUE) OK) (SETQ OK F)))))))))
(1 DEFINE (((BACKTEST (LAMBDA NIL (LEAVE (ENTER)
       (AND (TESTS)
        (OR (LOAD (CONS (QUOTE OR)
           (CONS (STAR1)
            (CONS (CONS (QUOTE SETQ)
              (CONS (QUOTE OK) (CONS (QUOTE F) NIL))) NIL))))
        (SETQ OK F)))))))))
(1 DEFINE (((TESTS (LAMBDA NIL (LEAVE (ENTER)
       (AND (OR (AND (ID)
          (OR (AND (OR (AND (PARAMSQ)
              (OR (LOAD (CONS (STAR2) (STAR1))) (SETQ OK F)))
             (NOT CK)
             (AND TRUE (OR (LOAD (CONS (STAR1) NIL))
              (SETQ OK F)))) OK) (SETQ OK F)))
         (NOT CK)
         (AND (STRING1)
          (OR (LOAD (CONS (QUOTE CMPR) (CONS (STAR1) NIL)))
           (SETQ OK F)))
         (NOT CK)
         (AND (CMPR (QUOTE ('+)))
          (OR (STRING1) (SETQ OK F))
          (OR (LOAD (CONS (QUOTE CCMPS) (CONS (STAR1) NIL)))
           (SETQ OK F)))
         (NOT CK)
         (AND (CMPR (QUOTE ('-)))
          (OR (STRING1) (SETQ OK F))
          (OR (LOAD (CONS (QUOTE NCOMP) (CONS (STAR1) NIL)))
           (SETQ OK F)))
         (NOT CK)
         (AND (CMPR (QUOTE (' )))
          (OR (AND (OR (AND (ID)
              (OR (LOAD (CONS (QUOTE LOAD)
                 (CONS (CONS (QUOTE QUOTE)
                  (CONS (STAR1) NIL)) NIL))) (SETQ OK F)))
             (NOT CK)
             (AND (STRING1)
              (OR (LOAD (CONS (QUOTE CMPR2) (CONS (STAR1) NIL)))
               (SETQ OK F)))) OK) (SETQ OK F)))
```

```
          (NOT CK)
          (AND (CMPR (QUOTE (' )))
           (OR (EXPR) (SETQ OK F))
           (OR (CMPR (QUOTE (' ))) (SETQ OK F))
           (OR (LOAD (CONS (QUOTE SEQ)
              (CONS (CONS (QUOTE FLAGS) NIL) (CONS (STAR1) NIL))))
            (SETQ OK F)))
          (NOT CK)
          (AND (CMPR (QUOTE ('())))
            (OR (EXPR) (SETQ OK F))
            (OR (CMPR (QUOTE (')))) (SETQ OK F)))
          (NOT CK)
          (AND (CMPR (QUOTE ('. 'E 'M 'P 'T 'Y)))
           (OR (LOAD (QUOTE TRUE)) (SETQ OK F)))
          (NOT CK)
          (AND (CMPR (QUOTE (',)))
           (OR (STRING1) (SETQ OK F))
           (OR (LOAD (CONS (QUOTE INSERT) (CONS (STAR1) NIL)))
            (SETQ OK F)))
          (NOT CK)
          (AND (LISTX)
           (OR (LOAD (CONS (QUOTE LOAD) (CONS (STAR1) NIL)))
            (SETQ OK F)))
          (NOT CK)
          (AND (CMPR (QUOTE ('. '())))
            (OR (SEQ (FLAGS)
              (PROG NIL M00005 (COND ((OUTPUT) (GO M00005)))
               (RETURN OK))) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE PROG)
               (CONS (QUOTE NIL)
                (APPEND (STAR1)
                 (CONS (CONS (QUOTE RETURN)
                   (CONS (QUOTE T) NIL)) NIL))))) (SETQ OK F))
            (OR (CMPR (QUOTE (')))) (SETQ OK F)))
          (NOT CK) (REPEAT)) OK)))))))
 (1 DEFINE (((OUTPUT (LAMBDA NIL (LEAVE (ENTER)
        (AND (OR (AND (STRING1)
           (OR (LOAD (CONS (QUOTE PRINST) (CONS (STAR1) NIL)))
            (SETQ OK F)))
          (NOT CK)
          (AND (ID)
           (OR (AND (OR (AND (PARAMSQ)
              (OR (LOAD (CONS (QUOTE PRINO)
                 (CONS (CONS (STAR2) (STAR1)) NIL))) (SETQ OK F)))
             (NOT CK)
             (AND TRUE (OR (LOAD (CONS (QUOTE PRINO)
                (CONS (STAR1) NIL))) (SETQ OK F)))) OK)
            (SETQ OK F)))
          (NOT CK)
          (AND (STACK)
           (OR (LOAD (CONS (QUOTE PRINO) (CONS (STAR1) NIL)))
            (SETQ OK F)))
          (NOT CK)
          (AND (CMPR (QUOTE ('/)))
           (OR (LOAD (CONS (QUOTE TERPRI) NIL))
            (SETQ OK F)))) OK)))))))
 (1 DEFINE (((REPEAT (LAMBDA NIL (LEAVE (ENTER)
        (AND (CMPR (QUOTE ('$)))
         (OR (TESTS) (SETQ OK F))
         (OR (LOAD (CONS (QUOTE PROG)
           (CONS (QUOTE NIL)
            (CONS (GN1)
```

```
                (CONS (CONS (QUOTE COND)
                  (CONS (CONS (STAR1)
                    (CONS (CONS (QUOTE GO)
                      (CONS (GN1) NIL)) NIL)) NIL))
                  (CONS (CONS (QUOTE RETURN)
                    (CONS (QUOTE OK) NIL)) NIL)))))) (SETQ OK F)))))))))
(1 DEFINE (((STACK (LAMBDA NIL (LEAVE (ENTER)
        (AND (OR (AND (CMPR (QUOTE ('* '1)))
             (OR (LOAD (CONS (QUOTE STAR1) NIL)) (SETQ OK F)))
            (NOT OK)
            (AND (CMPR (QUOTE ('* '2)))
             (OR (LOAD (CONS (QUOTE STAR2) NIL)) (SETQ OK F)))
            (NOT OK)
            (AND (CMPR (QUOTE ('* '3)))
             (OR (LOAD (CONS (QUOTE STAR3) NIL)) (SETQ OK F)))
            (NOT OK)
            (AND (CMPR (QUOTE ('* '4)))
             (OR (LOAD (CONS (QUOTE STAR4) NIL)) (SETQ OK F)))
            (NOT OK)
            (AND (CMPR (QUOTE ('+ '* '1)))
             (OR (LOAD (CONS (QUOTE CAR) (CONS (QUOTE STAR) NIL)))
              (SETQ OK F)))
            (NOT OK)
            (AND (CMPR (QUOTE ('+ '* '2)))
             (OR (LOAD (CONS (QUOTE CADR) (CONS (QUOTE STAR) NIL)))
              (SETQ OK F)))
            (NOT OK)
            (AND (CMPR (QUOTE ('+ '* '3)))
             (OR (LOAD (CONS (QUOTE CADDR) (CONS (QUOTE STAR) NIL)))
              (SETQ OK F)))) OK))))))
(1 DEFINE (((PARAMSQ (LAMBDA NIL (LEAVE (ENTER)
        (AND (CMPR (QUOTE (' )))
         (OR (SEQ (FLAGS)
           (AND (OR (AND (EXPNQ)
              (CR (PROG NIL M00006 (COND ((AND (CMPR (QUOTE (',)))
                   (OR (EXPNQ) (SETQ OK F))) (GO M00006)))
                (RETURN OK)) (SETQ OK F))) (NOT OK) TRUE) OK))
         (SETQ OK F)) (OR (CMPR (QUOTE (' ))) (SETQ OK F)))))))))
(1 DEFINE (((GCBBLE (LAMBDA NIL (LEAVE (ENTER)
        (AND (ERRORX)
         (OR (PROG NIL M00007 (COND ((AND (NCOMP (QUOTE (' )))
              (CR (DELETEX) (SETQ OK F))) (GO M00007))) (RETURN OK))
         (SETQ OK F)) (OR (CMPR (QUOTE (' ))) (SETQ OK F)))))))))
(1 DEFINE (((STRING1 (LAMBDA NIL (LEAVE (ENTER)
        (TOKEN (MARK)
         (AND (CMPR (QUOTE (''')))
          (CR (PROG NIL M00008 (COND ((AND (OR (AND (NCOMP (QUOTE (''')))
               ) (OR (ANY) (SETQ OK F)))
              (NOT OK)
              (AND (CMPR (QUOTE (''' ''')))
               (OR (INSERT (QUOTE (''')))) (SETQ OK F)))) OK)
             (GO MC0008))) (RETURN OK)) (SETQ OK F))
          (CR (CMPR (QUOTE (''')))) (SETQ OK F))
          (CR (LOAD (CONS (QUOTE QUOTE) (CONS (STAR1) NIL)))
           (SETQ OK F)))))))))
(1 DEFINE (((ID (LAMBDA NIL (LEAVE (ENTER)
        (TOKEN (MARK)
         (AND (LET)
          (CR (PROG NIL M00009 (COND ((AND (OR (LET)
               (NOT OK)
               (DGT)
               (NOT OK)
```

```
                (AND (CMPR (QUOTE ('-)))
                 (OR (INSERT (QUOTE ('*))) (SETQ OK F)))) OK)
              (GO MOOOO9))) (RETURN OK)) (SETQ OK F))
          (CR (MAKEATOM) (SETQ OK F))))))))))
 (1 DEFINE (((NUM (LAMBDA NIL (LEAVE (ENTER)
        (TOKEN (MARK)
         (AND (DGT)
          (CR (PROG NIL MOOO10 (COND ((DGT) (GO MOOO10)))
            (RETURN OK)) (SETQ OK F))
          (CR (MAKENUMBER) (SETQ OK F)))))))))
 (1 DEFINE (((LET (LAMBDA NIL (LEAVE (ENTER) (ISIT LETTER T)))))))
 (1 DEFINE (((DGT (LAMBDA NIL (LEAVE (ENTER) (ISIT DIGIT T)))))))
 (1 DEFINE (((CHARACTER (LAMBDA NIL (LEAVE (ENTER)
        (TOKEN (MARK)
         (AND (CMPR (QUOTE (' )))
          (CR (ANY) (SETQ OK F)) (OR (MAKECHR) (SETQ OK F)))))))))
 (1 DEFINE (((ELEMENT (LAMBDA NIL (LEAVE (ENTER)
        (AND (OR (NUM)
          (NOT CK)
          (STRING1)
          (NOT CK)
          (CHARACTER)
          (NOT CK)
          (IDENT)
          (NOT CK)
          (STACK)
          (NOT CK)
          (LISTX)
          (NOT CK)
          (AND (CMPR (QUOTE ('())))
            (OR (EXPNQ) (SETQ OK F))
            (OR (CMPR (QUOTE (')))) (SETQ OK F)))) OK))))))
 (1 DEFINE (((IDENT (LAMBDA NIL (LEAVE (ENTER)
        (AND (IC)
         (OR (AND (CR (AND (PARAMSQ)
             (OR (LOAD (CONS (STAR2) (STAR1))) (SETQ OK F)))
            (NOT OK)
            (AND (CMPR (QUOTE ('  ' )))
             (OR (EXPNQ) (SETQ OK F))
             (OR (CMPR (QUOTE (' ))) (SETQ OK F))
             (OR (AND (OR (AND (CMPR (QUOTE ('  '=)))
                 (OR (EXPNQ) (SETQ OK F))
                 (OR (LOAD (CONS (QUOTE DEFLIST)
                    (CONS (CONS (QUOTE LIST)
                     (CONS (CONS (QUOTE LIST)
                      (CONS (STAR2) (CONS (STAR1) NIL))) NIL))
                     (CONS (CONS (QUOTE QUOTE)
                      (CONS (STAR1) NIL)) NIL)))) (SETQ OK F)))
               (NOT OK)
               (AND TRUE (OR (LOAD (CONS (QUOTE GET)
                    (CONS (STAR1)
                     (CONS (CONS (QUOTE QUOTE)
                      (CONS (STAR1) NIL)) NIL)))) (SETQ OK F)))) OK)
              (SETQ OK F)))
            (NOT OK)
            (AND (CMPR (QUOTE ('  '=)))
             (OR (EXPNQ) (SETQ OK F))
             (OR (LOAD (CONS (QUOTE SETQ)
              (CONS (STAR2) (CONS (STAR1) NIL)))) (SETQ OK F)))
            (NOT OK)
            (AND (OR (AND (SETQ ID*V ID*V)
              (OR (LOAD (CONS (QUOTE QUOTE) (CONS (STAR1) NIL)))
```

```
                (SETQ OK F))) (NOT OK) TRUE) OK)) OK)
          (SETQ OK F)))))))))
 (1 DEFINE (((LIST*SEQ (LAMBDA NIL (LEAVE (ENTER)
        (AND (OR (AND (CMPR (QUOTE ('- ' )))
           (OR (EXPNQ) (SETQ OK F))
           (OR (CMPR (QUOTE ('  '-))) (SETQ OK F))
           (OR (AND (OR (AND (CMPR (QUOTE (',)))
               (OR (LIST*SEQ) (SETQ OK F))
               (OR (LOAD (CONS (QUOTE APPEND)
                  (CONS (STAR2) (CONS (STAR1) NIL)))) (SETQ OK F)))
             (NOT OK) TRUE) OK) (SETQ OK F)))
          (NOT OK)
          (AND (EXPQ)
           (OR (AND (OR (AND (CMPR (QUOTE (',)))
               (OR (LIST*SEQ) (SETQ OK F)))
              (NOT OK) (AND TRUE (OR (LOAD NIL) (SETQ OK F)))) OK)
            (SETQ OK F))
           (OR (LOAD (CONS (QUOTE CONS)
              (CONS (STAR2) (CONS (STAR1) NIL))))
            (SETQ OK F)))) OK)))))))
 (1 DEFINE (((LISTX (LAMBDA NIL (LEAVE (ENTER)
        (AND (CMPR (QUOTE ('. ' )))
         (OR (AND (CR (LIST*SEQ)
            (NOT OK) (AND TRUE (OR (LOAD NIL) (SETQ OK F)))) OK)
          (SETQ OK F)) (OR (CMPR (QUOTE (' ))) (SETQ OK F)))))))))
 (1 DEFINE (((EXPQ (LAMBDA NIL (LEAVE (ENTER)
        (WHERE (FUNCTION EXPX) T)))))))
 (1 DEFINE (((EXPNQ (LAMBDA NIL (LEAVE (ENTER)
        (WHERE (FUNCTION EXPX) F)))))))
 (1 DEFINE (((LISTEXP (LAMBDA NIL (LEAVE (ENTER)
        (AND (ELEMENT)
         (OR (PROG NIL M00011 (COND ((AND (OR (AND (CMPR (QUOTE ('. '1)
                 ))
                (OR (LOAD (CONS (QUOTE CAR) (CONS (STAR1) NIL)))
                 (SETQ OK F)))
               (NOT OK)
               (AND (CMPR (QUOTE ('. '2)))
                (OR (LOAD (CONS (QUOTE CADR) (CONS (STAR1) NIL)))
                 (SETQ OK F)))
               (NOT OK)
               (AND (CMPR (QUOTE ('. '3)))
                (OR (LOAD (CONS (QUOTE CADDR) (CONS (STAR1) NIL)))
                 (SETQ OK F)))
               (NOT OK)
               (AND (CMPR (QUOTE ('. '4)))
                (OR (LOAD (CONS (QUOTE CADDDR) (CONS (STAR1) NIL)))
                 (SETQ OK F)))
               (NOT OK)
               (AND (CMPR (QUOTE ('  '2)))
                (OR (LOAD (CONS (QUOTE CDR) (CONS (STAR1) NIL)))
                 (SETQ OK F)))
               (NOT OK)
               (AND (CMPR (QUOTE ('  '3)))
                (OR (LOAD (CONS (QUOTE CDDR) (CONS (STAR1) NIL)))
                 (SETQ OK F)))
               (NOT OK)
               (AND (CMPR (QUOTE ('  '4)))
                (OR (LOAD (CONS (QUOTE CDDDR) (CONS (STAR1) NIL)))
                 (SETQ OK F)))) OK) (GO M00011))) (RETURN OK))
          (SETQ OK F)))))))))
 (1 DEFINE (((BASIC (LAMBDA NIL (LEAVE (ENTER)
        (AND (LISTEXP)
```

```
        (OR (AND (CR (AND (CMPR (QUOTE ('*)))
            (OR (BASIC) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE CONS)
               (CONS (STAR2) (CONS (STAR1) NIL)))) (SETQ OK F)))
          (NOT OK) TRUE) OK) (SETQ OK F)))))))))
(1 DEFINE (((RELATION (LAMBDA NIL (LEAVE (ENTER)
       (AND (BASIC)
        (OR (AND (CR (AND (CMPR (QUOTE ('=)))
            (OR (BASIC) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE EQUAL)
               (CONS (STAR2) (CONS (STAR1) NIL)))) (SETQ OK F)))
          (NOT OK)
          (AND (CMPR (QUOTE ('- '=)))
           (OR (BASIC) (SETQ OK F))
           (OR (LOAD (CONS (QUOTE NOT)
              (CONS (CONS (QUOTE EQUAL)
                 (CONS (STAR2) (CONS (STAR1) NIL))) NIL)))
            (SETQ OK F))) (NOT OK) TRUE) OK) (SETQ OK F)))))))))
(1 DEFINE (((NEGATION (LAMBDA NIL (LEAVE (ENTER)
       (AND (OR (AND (CMPR (QUOTE ('-)))
          (OR (RELATION) (SETQ OK F))
          (OR (LOAD (CONS (QUOTE NOT) (CONS (STAR1) NIL)))
           (SETQ OK F))) (NOT OK) (RELATION)) OK))))))
(1 DEFINE (((FACTOR (LAMBDA NIL (LEAVE (ENTER)
       (AND (NEGATION)
        (OR (AND (CR (AND (CMPR (QUOTE ('. 'A '.)))
            (OR (FACTOR) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE AND)
               (CONS (STAR2) (CONS (STAR1) NIL)))) (SETQ OK F)))
          (NOT OK) TRUE) OK) (SETQ OK F)))))))))
(1 DEFINE (((EXPX (LAMBDA NIL (LEAVE (ENTER)
       (AND (FACTOR)
        (OR (AND (CR (AND (CMPR (QUOTE ('. 'V '.)))
            (OR (EXPX) (SETQ OK F))
            (OR (LOAD (CONS (QUOTE OR)
               (CONS (STAR2) (CONS (STAR1) NIL)))) (SETQ OK F)))
          (NOT OK) TRUE) OK) (SETQ OK F)))))))))
(1 DEFINE (((LOOPST (LAMBDA NIL (LEAVE (ENTER)
       (AND (CMPR (QUOTE ('. 'L 'O 'O 'P)))
        (OR (CMPR (QUOTE ('U 'N 'T 'I 'L))) (SETQ OK F))
        (OR (GEN1) (SETQ OK F))
        (OR (RESTORE (SAVER)
          (AND (OR (BACKUP (EXPNQ))
           (AND (ERRORX)
            (OR (PROG NIL M00012 (COND ((AND (NCOMP (QUOTE ('. 'B 'E
                 'G 'I 'N))) (OR (DELETEX) (SETQ OK F)))
                (GO M00012))) (RETURN OK)) (SETQ OK F)))) OK))
         (SETQ OK F))
        (OR (CMPR (QUOTE ('. 'B 'E 'G 'I 'N))) (SETQ OK F))
        (OR (LOAD (CONS (QUOTE COND)
          (CONS (CONS (STAR1)
           (CONS (CONS (QUOTE GO) (CONS (GN2) NIL)) NIL)) NIL)))
         (SETQ OK F))
        (OR (PROG NIL M00013 (COND ((RESTORE (SAVER)
            (AND (OR (BACKUP (ST))
              (AND (NCOMP (QUOTE ('. 'E 'N 'D)))
               (OR (GOBBLE) (SETQ OK F)))) OK)) (GO M00013)))
          (RETURN OK)) (SETQ OK F))
        (OR (CMPR (QUOTE ('. 'E 'N 'D))) (SETQ OK F))
        (OR (LOAD (CONS (QUOTE GO) (CONS (GN1) NIL))) (SETQ OK F))
        (OR (GEN2) (SETQ OK F)))))))))
(1 DEFINE (((IFST (LAMBDA NIL (LEAVE (ENTER)
```

```
        (AND (CMPR (QUOTE (". 'I 'F)))
         (OR (RESTORE (SAVER)
           (AND (OR (BACKUP (EXPNQ))
             (AND (ERRORX)
              (CR (PROG NIL M00014 (COND ((AND (NCOMP (QUOTE (". 'B 'E
                    'G 'I 'N))) (OR (DELETEX) (SETQ OK F)))
                  (GO M00014))) (RETURN OK)) (SETQ OK F)))) OK))
            (SETQ OK F))
           (OR (CMPR (QUOTE (". 'B 'E 'G 'I 'N))) (SETQ OK F))
           (OR (AND (CR (CMPR (QUOTE (". 'T 'H 'E 'N)))
             (NOT OK) TRUE) OK) (SETQ OK F))
           (OR (LOAD (CONS (QUOTE COND)
             (CONS (CONS (CONS (QUOTE NOT) (CONS (STAR1) NIL))
              (CONS (CONS (QUOTE GO) (CONS (GN1) NIL)) NIL)) NIL)))
            (SETQ OK F))
           (OR (PROG NIL M00015 (COND ((ST) (GO M00015))) (RETURN OK))
            (SETQ OK F))
           (OR (AND (CR (AND (CMPR (QUOTE (". 'E 'L 'S 'E)))
              (OR (LOAD (CONS (QUOTE GO) (CONS (GN2) NIL)))
               (SETQ OK F))
              (OR (GEN1) (SETQ OK F))
              (OR (PROG NIL M00016 (COND ((RESTORE (SAVER)
                   (AND (OR (BACKUP (ST))
                     (AND (NCOMP (QUOTE (". 'E 'N 'D)))
                      (OR (GOBBLE) (SETQ OK F)))) OK)) (GO M00016)))
                (RETURN OK)) (SETQ OK F))
              (OR (CMPR (QUOTE (". 'E 'N 'D))) (SETQ OK F))
              (OR (GEN2) (SETQ OK F)))
             (NOT OK)
             (AND (CMPR (QUOTE (". 'E 'N 'D)))
              (OR (GEN1) (SETQ OK F)))) OK) (SETQ OK F)))))))))
(1 DEFINE (((PRINTST (LAMBDA NIL (LEAVE (ENTER)
       (AND (CMPR (QUOTE (". 'P 'R 'I 'N 'T)))
        (OR (PROG NIL M00017 (COND ((OUTPUT) (GO M00017)))
          (RETURN OK)) (SETQ OK F))
        (OR (CMPR (QUOTE (" ))) (SETQ OK F)))))))))
(1 DEFINE (((ST (LAMBDA NIL (LEAVE (ENTER)
       (AND (OR (AND (EXPNQ) (OR (CMPR (QUOTE (" ))) (SETQ OK F)))
         (NOT OK)
         (LOOPST) (NOT OK) (IFST) (NOT OK) (PRINTST)) OK))))))
(1 DEFINE (((IDSEQ (LAMBDA NIL (LEAVE (ENTER)
       (AND (CMPR (QUOTE (" )))
        (OR (SEQ (FLAGS)
          (AND (AND (OR (AND (FORMAL)
            (OR (PROG NIL M00018 (COND ((AND (CMPR (QUOTE (",)))
                 (OR (FORMAL) (SETQ OK F))) (GO M00018)))
              (RETURN OK)) (SETQ OK F))) (NOT OK) TRUE) OK)
           (OR (CMPR (QUOTE (" ))) (SETQ OK F))) (SETQ OK F)))))))))
(1 DEFINE (((FORMAL (LAMBDA NIL (LEAVE (ENTER)
       (AND (OR (ID)
         (NOT OK)
         (AND (CMPR (QUOTE (". 'L 'O 'C)))
          (OR (ID) (SETQ OK F)))) OK))))))
(1 DEFINE (((PROCEDURE (LAMBDA NIL (LEAVE (ENTER)
       (AND (CMPR (QUOTE (". 'P 'R 'O 'C 'E 'D 'U 'R 'E)))
        (OR (ID) (SETQ OK F))
        (OR (IDSEQ) (SETQ OK F))
        (OR (CMPR (QUOTE (" ))) (SETQ OK F))
        (OR (AND (CR (AND (CMPR (QUOTE (". 'L 'O 'C 'A 'L)))
            (OR (IDSEQ) (SETQ OK F))
            (OR (CMPR (QUOTE (" ))) (SETQ OK F)))
           (NOT OK) (AND TRUE (OR (LOAD NIL) (SETQ OK F)))) OK)
```

```
          (SETQ OK F))
         (OR (SEQ (FLAGS)
          (AND (PROG NIL M00019 (COND ((RESTORE (SAVER)
               (AND (OR (BACKUP (ST))
                 (AND (NCOMP (QUOTE ('. 'R 'E 'T 'U 'R 'N)))
                  (OR (GOBBLE) (SETQ OK F)))) OK)) (GO M00019)))
           (RETURN OK))
          (OR (CMPR (QUOTE ('. 'R 'E 'T 'U 'R 'N))) (SETQ OK F))
          (OR (AND (OR (AND (CMPR (QUOTE (' )))
               (OR (EXPNQ) (SETQ OK F))
               (OR (LOAD (CONS (QUOTE RETURN) (CONS (STAR1) NIL)))
                (SETQ OK F)) (OR (CMPR (QUOTE (' ))) (SETQ OK F)))
             (NOT OK) TRUE) OK) (SETQ OK F))
          (OR (CMPR (QUOTE (' ))) (SETQ OK F)))) (SETQ OK F))
        (OR (LOAD (CONS (QUOTE DEFINE)
          (CONS (CONS (CONS (CONS (STAR4)
               (CONS (CONS (QUOTE LAMBDA)
                 (CONS (STAR3)
                  (CONS (CONS (QUOTE PROG)
                   (CONS (STAR2)
                    (STAR1))) NIL))) NIL)) NIL) NIL) NIL)))
         (SETQ OK F)) (OR (COMPILE) (SETQ OK F)))))))))
(1 DEFINE (((LISP*DIVISION (LAMBDA NIL (LEAVE (ENTER)
        (AND (CMPR (QUOTE ('. 'L 'I 'S 'P 'X)))
         (OR (PROG NIL M00020 (COND ((PROCEDURE) (GO M00020)))
          (RETURN OK)) (SETQ OK F))
         (OR (CMPR (QUOTE ('. 'F 'I 'N 'I 'S 'H))) (SETQ OK F)))))))))
(1 DEFINE (((FLUID*DECLARATION (LAMBDA NIL (LEAVE (ENTER)
        (AND (CMPR (QUOTE ('. 'D 'E 'C 'L 'A 'R 'E)))
         (OR (CMPR (QUOTE (' ))) (SETQ OK F))
         (OR (FLUID1) (SETQ OK F))
         (OR (PROG NIL M00021 (COND ((AND (CMPR (QUOTE (',)))
              (OR (FLUID1) (SETQ OK F))) (GO M00021))) (RETURN OK))
          (SETQ OK F))
         (OR (CMPR (QUOTE (' ))) (SETQ OK F))
         (OR (CMPR (QUOTE (' ))) (SETQ OK F)))))))))
(1 DEFINE (((FLUID1 (LAMBDA NIL (LEAVE (ENTER)
        (AND (ID)
         (OR (LOAD (CONS (QUOTE CSET)
           (CONS (CONS (STAR1) (CONS (QUOTE NIL) NIL)) NIL)))
          (SETQ OK F)) (OR (COMPILE) (SETQ OK F)))))))))
(1 DEFINE (((PROGRAM (LAMBDA NIL (LEAVE (ENTER)
        (AND (PROG NIL M00022 (COND ((AND (OR (SYNTAX)
             (NOT OK)
             (LISP*DIVISION) (NOT OK) (FLUID*DECLARATION)) OK)
            (GO M00022))) (RETURN OK))
         (OR (CMPR (QUOTE ('. 'S 'T 'O 'P))) (SETQ OK F)))))))))

          ECF CARD  ALL PROGRAM COMPATIBLE
T              ENDINPUT
```

Library of META written in LISPX

MLIBR1

```
0000100-.DECLARE [STAR, GEN, SKIP-BLANKS, LINE,
0000200-      INPUT, COLUMN, BACK, CHR, FLAGX, LETTER, DIGIT,
0000300-      ODGT, ID-V, COUNT, MAXIMUM, OK, TRUE] ;
0000400-.LISPX
0000500-.PROCEDURE INITIALIZE [A, B];
0000600-.IF A = QUOTE[TTY]
0000700-      .BEGIN
0000800-      .THEN RDS[NIL];
0000900-      .ELSE OPEN[A, QUOTE[DISC], QUOTE[PERM] ];
0001000-      RDS[A];
0001100-      .END
0001200-.IF B = QUOTE[TTY]
0001300-      .BEGIN
0001400-      .THEN WRS[NIL];
0001500-      .ELSE  .IF  B = QUOTE[CORE]
0001600-           .BEGIN
0001700-           .THEN OPEN[ QUOTE[SDCSDC], QUOTE[DISC] ];
0001800-           WRS[QUOTE[SDCSDC] ];
0001900-           .ELSE  OPEN[ B, QUOTE[DISC] ];  WRS[B];
0002000-           .END
0002100-      .END
0002200-STAR := NIL;  GEN := NIL;  SKIP-BLANKS := T;
0002300-COLUMN := 0; LINE := READLN[]; INPUT := LINE; CHR := INPUT.1;
0002400-BACK := NIL;  COUNT := 1;  MAXIMUM := 1;  OK := T;
0002500-FLAGX := NIL; LETTER := 2; DIGIT := 12; ODGT := 8;
0002600-GENCH1[]; TRUE := T;
0002700-.RETURN;
0002800-.PROCEDURE COMPLETE[Z, A];  .LOCAL[X];
0002900-.IF -A
0003000-      .BEGIN
0003100-      .THEN  ERRORX[] ;
0003300-      .END
0003400-X := RDS[NIL];
0003500-.IF X -= QUOTE[TTY]
0003600-      :BEGIN
0003700-      .THEN  SHUT[X] ;
0003800-      .END
0003900-X := WRS[NIL];
0004000-.IF X = QUOTE[SDCSDC] .A. A
0004100-      .BEGIN
0004200-      .THEN  POSITION[X, QUOTE[WEOF]] ;
0004300-      POSITION[X, QUOTE[REWIND] ];
0004400-      .LOOP UNTIL  LOADEXP[X] = QUOTE[EOF]
0004500-           .BEGIN
0004600-           .END
0004700-      SHUT[X, QUOTE[DELETE]];
0004800-      .ELSE .IF X -= QUOTE[TTY]
0004900-           .BEGIN
0005000-           .THEN  POSITION[X, QUOTE[WEOF]];
0005100-           SHUT[X];
0005200-           .END
0005300-      .END
0005400-.RETURN[A];
```

MLIBR/

```
0005500-.PROCEDURE READLN[];  .LOCAL[X];
0005600-X := READCH[];
0005700-.IF NULL[X]
0005800-      .BEGIN
0005900-      .THEN RETURN[NIL];
0006000-      .ELSE RETURN[X*READLN[]];
0006100-      .END
0006200-.RETURN;
0006300-.PROCEDURE PRINTLN[U];   .LOCAL[X];
0006400-X := PRINTLN1[U];
0006500-.LOOP UNTIL NULL[X]
0006600-      .BEGIN
0006700-      PRINTCH[X.1];   X := X:2;
0006800-      .END
0006900-TERPRI[];
0007000-.RETURN;
0007100-.PROCEDURE NXTCHR[];
0007200-.IF NULL[INPUT:2]
0007300-      .BEGIN
0007400-      .THEN COLUMN := COUNT;   LINE := READLN[];
0007500-      RPLACD[INPUT, LINE];
0007600-      .END
0007700-INPUT := INPUT:2;   CHR := INPUT.1;
0007800-COUNT := ADD1[COUNT];   MAXIMUM := MAX[COUNT, MAXIMUM];
0007900-.RETURN;
0008000-.PROCEDURE LOAD[X];
0008100-STAR := X * STAR ;
0008200-.RETURN[T];
0008300-.PROCEDURE COMPARE[S, N];   .LOCAL[U, V, W, SIGNAL];
0008400-.IF SKIP-BLANKS
0008500-      .BEGIN
0008600-      .THEN
0008700-      .LOOP UNTIL CHR -= #
0008800-            .BEGIN
0008900-            NXTCHR[];
0009000-            .END
0009100-      .END
0009200-U := S;   V := INPUT;   W := COUNT;
0009300-.LOOP UNTIL (NULL[U] .V. U.1 -= CHR)
0009400-      .BEGIN
0009500-      U := U:2;   NXTCHR[];
0009600-      .END
0009700-SIGNAL := NULL[U];
0009800-.IF N = 3
0009900-      .BEGIN
0010000-      .THEN   SIGNAL := -SIGNAL ;   INPUT := V;
0010100-      CHR := INPUT.1;   COUNT := W;
0010200-      .ELSE
0010300-      .IF SIGNAL
0010400-            .BEGIN
0010500-            .THEN
0010600-            .IF N = 2
0010700-                  .BEGIN
0010800-                  .THEN STAR :=
0010900-                  APPEND[STAR.1, S] * STAR:2 ;
0011000-                  .END
0011100-            .ELSE  INPUT := V;   CHR := INPUT.1;   COUNT := W;
0011200-            .END
0011300-      .END
0011400-.RETURN[SIGNAL];
```

MLIB21

```
0011500-.PROCEDURE CMPR2[X];
0011600-.IF CMPR[X]
0011700-      .BEGIN
0011800-      .THEN  STAR := COMPRESS[X] * STAR ;  RETURN[T];
0011900-      .END
0012000-.RETURN[F];
0012100-.PROCEDURE CMPR[S];
0012200-.RETURN[COMPARE[S,1]];
0012300-.PROCEDURE COMPS[S];
0012400-.RETURN[COMPARE[S, 2]];
0012500-.PROCEDURE NCOMP[S];
0012600-.RETURN[COMPARE[S, 3]];
0012700-.PROCEDURE ERRORX[];  .LOCAL[X];
0012800-X := WRS[NIL];
0012900-PRINTLN[LINE];  BLANKS[ SUBI[ DIFFER [MAXIMUM, COLUMN]]] ;
0013000-PRINTCH[#↑];  TERPRI[];
0013100-WRS[X];  OK := T;
0013200-.RETURN[T];
0013300-.PROCEDURE MARK[];
0013400-.LOOP UNTIL CHR -= #
0013500-      .BEGIN
0013600-      NXTCHR[];
0013700-      .END
0013800-SKIP-BLANKS := F;
0013900-STAR := NIL * STAR ;
0014000-.RETURN;
0014100-.PROCEDURE TOKEN[W, X];
0014200-SKIP-BLANKS := T ;
0014300-.IF -X
0014400-      .BEGIN
0014500-      .THEN STAR := STAR:2 ;
0014600-      .END
0014700-.RETURN[X];
0014800-.PROCEDURE INSERT[S];
0014900-STAR := APPEND[STAR.1, S] * STAR:2 ;
0015000-.RETURN[T];
0015100-.PROCEDURE STAR1[];  .LOCAL[X];
0015200-X := STAR.1;  STAR := STAR:2;
0015300-.RETURN[X];
0015400-.PROCEDURE STAR2[];  .LOCAL[X];
0015500-X:= STAR.2;  STAR:= STAR.1 * STAR:3;
0015600-.RETURN[X];
0015700-.PROCEDURE STAR3[];  .LOCAL[X];
0015800-X := STAR.3;  STAR:= STAR.1 * STAR.2 * STAR:4;
0015900-.RETURN[X];
0016000-.PROCEDURE STAR4[]; .LOCAL[X];
0016100-X := STAR.4; STAR := STAR.1 * STAR.2 * STAR.3 * STAR:4:2;
0016200-.RETURN[X];
0016300-.PROCEDURE FLAGS[];
0016400-FLAGX := STAR * FLAGX;  STAR := NIL;
0016500-.RETURN;
0016600-.PROCEDURE SEQ[W, X];
0016700-.IF X
0016800-      .BEGIN
0016900-      .THEN  STAR := REVERSE[STAR] * FLAGX.1 ;
0016910-      .ELSE STAR := FLAGX.1;
0017000-      .END
0017100-FLAGX := FLAGX:2 ;
0017200-.RETURN[X];
```

MLIBR/

```
0017300-.PROCEDURE GN1[];
0017400-.IF NULL[GEN.1]
0017500-      .BEGIN
0017600-      .THEN GEN := GENSYM[] * GEN:2 ;
0017700-      .END
0017800-.RETURN[GEN.1];
0017900-.PROCEDURE GN2[];
0018000-.IF NULL[GEN.2]
0018100-      .BEGIN
0018200-      .THEN GEN := GEN.1 * GENSYM[] * GEN:3 ;
0018300-      .END
0018400-.RETURN[GEN.2];
0018500-.PROCEDURE GEN1[];
0018600-STAR := GN1[] * STAR;
0018700-.RETURN[T];
0018800-.PROCEDURE GEN2[];
0018900-STAR := GN2[] * STAR;
0019000-.RETURN[T];
0019100-.PROCEDURE ANY[];
0019200-STAR := APPEND[STAR.1, .[(CHR)]] * STAR:2;
0019300-NXTCHR[];
0019400-.RETURN[T];
0019500-.PROCEDURE DELETEX[];
0019600-NXTCHR[];
0019700-.RETURN[T];
0019800-.PROCEDURE MAKEATOM[];
0019900-STAR := COMPRESS[STAR.1] * STAR:2 ;
0020000-.RETURN[T];
0020100-.PROCEDURE MAKENUMBER[];  .LOCAL[S, N];
0020200-S := STAR.1;  N := 0;
0020300-.LOOP UNTIL NULL[S]
0020400-      .BEGIN
0020500-      N := PLUS[TIMES[N, 10], CHR2OCT[S.1]];  S := S:2;
0020600-      .END
0020700-STAR := N * STAR:2;
0020800-.RETURN[T];
0020900-.PROCEDURE COMPILE[];
0021000-PRINT[1 * STAR.1]; STAR := STAR:2;
0021100-.RETURN[T];
0021200-.PROCEDURE ISIT[X, Y];  .LOCAL[SIGNAL];
0021300-SIGNAL := NOT[ZEROP[LOGAND[X, CONVERT[CHR]]]];
0021400-.IF SIGNAL
0021500-      .BEGIN
0021600-      .THEN .IF Y
0021700-            .BEGIN
0021800-            .THEN  STAR := APPEND[STAR.1, .[(CHR)]] * STAR:2 ;
0021900-            .END
0022000-      NXTCHR[];
0022100-      .END
0022200-.RETURN[SIGNAL];
0022300-.PROCEDURE MAKECHR[];
0022400-STAR := STAR.1.1 * STAR:2;
0022500-.RETURN[T];
0022600-.PROCEDURE  WHERE[X, Y];  .LOCAL[A, B];
0022700-A := ID-V;  ID-V := Y;  B := X[];  ID-V := A;
0022800-.RETURN[B];
```

MLIBR1

```
0022900-.PROCEDURE CHECK[A, B]; .LOCAL[X];
0023000-OPEN[A, QUOTE[DISC], QUOTE[PERM] ] ;
0023100-OPEN[B, QUOTE[DISC], QUOTE[PERM] ] ;
0023200-.LOOP UNTIL (X := PROG2[RDS[A],READ[]] ) = QUOTE[EOF]
0023300-      .BEGIN
0023400-      .IF X -= PROG2 [ RDS[B], READ[] ]
0023500-            .BEGIN
0023600-            .THEN  PRINT[QUOTE[BAD]];
0023700-            .END
0023800-      .IF X.2 = QUOTE[DEFINE]
0023900-            .BEGIN
0024000-            .THEN  PRINT [ X.3.1.1.1 ] ;
0024100-            .END
0024200-      .END
0024300-RDS[NIL]; SHUT[A]; SHUT[B];
0024400-.RETURN;
0024500-.PROCEDURE PRINTLN1[U]; .LOCAL[X];
0024600-.IF U = NIL
0024700-      .BEGIN
0024800-      .THEN  RETURN[NIL];
0024900-      .END
0025000-X := PRINTLN1[U:2] ;
0025100-.IF X = NIL .A. U.1 = #
0025200-      .BEGIN
0025300-      .THEN  RETURN[NIL];
0025400-      .END
0025500-.RETURN[U.1 * X] ;
0025600-.PROCEDURE SAVER[];
0025700-BACK := .[(INPUT),(STAR),(FLAGX),(COUNT)] * BACK;
0025800-.RETURN;
0025900-.PROCEDURE RESTORE[W, X];
0026000-BACK := BACK:2 ;
0026100-.RETURN[X];
0026200-.PROCEDURE BACKUP[X]; .LOCAL[A];
0026300-A := X .A. OK;
0026400-.IF -OK
0026500-      .BEGIN
0026600-      .THEN  INPUT := BACK.1.1 ; STAR := BACK.1.2 ;
0026700-      FLAGX := BACK.1.3; COUNT := BACK.1.4;
0026800-      CHR := INPUT.1; OK := T;
0026900-      .END
0027000-.RETURN[A];
0027100-.PROCEDURE ENTER[];
0027200-GEN := NIL*NIL*GEN;
0027300-.RETURN;
0027400-.PROCEDURE LEAVE[X, Y];
0027500-GEN := GEN:3;
0027600-.RETURN[Y];
0027700-.PROCEDURE PRINST[X]; .LOCAL[Z];
0027800-Z := X;
0027900-.LOOP UNTIL Z = NIL
0028000-      .BEGIN
0028100-      PRINTCH[Z.1]; Z := Z:2;
0028200-      .END
0028300-.RETURN;
0028400-.FINISH
0028500-.STOP

READY=
```

```
(SESAME
LAP(
    ((CHR2OCT SUBR1)(BAX($ 2)1)(O(E CHR2OCT)1)
     (LDA $A 0 370Q)(SUB 1Q4 0 100370Q)(LDM $A)
     (BSX *MKNO 2 5)(BUC 0 4))
())
LAP(
    ((CONVERT SUBR 1)(BAX($ 2)1)(O(E CONVERT)1)
     (SUB 1Q4 0 100370Q)(LDA KONVERT 17Q)(BSX *MKNO 2 5)
     (BUC 0 4)
   KONVERT(11Q)(11Q)(11Q)(11Q)(11Q)(11Q)(11Q)(11Q)
           ( 5Q)( 5Q)(41Q)( 1Q)( 1Q)(41Q)(41Q)(41Q)
           ( 1Q)( 3Q)( 3Q)( 3Q)( 3Q)( 3Q)( 3Q)( 3Q)
           ( 3Q)( 3Q)(41Q)( 1Q)( 1Q)(41Q)(41Q)(41Q)
           ( 1Q)( 3Q)( 3Q)( 3Q)( 3Q)( 3Q)( 3Q)( 3Q)
           ( 3Q)( 3Q)(41Q)( 1Q)( 1Q)(41Q)(41Q)(41Q)
           (21Q)( 1Q)( 3Q)( 3Q)( 3Q)( 3Q)( 3Q)( 3Q)
           ( 3Q)( 3Q)(41Q)( 1Q)( 1Q)(41Q)(41Q)(41Q)
)(()
DEFINE(( (GENCH1(LAMBDA() (PROG()
      (*PLANT (LCGOR 201Q13 777744Q) 41242Q)
      (*PLANT 0 41251Q)))) ))
DEFINE(( (DUMP(LAMBDA(L)(MAPCAR L (FUNCTION EVAL1))))))
DEFINE((
      (MAGIC(LAMBDA(X)(PROG(W Y)
      (OPEN X (QUOTE DISC)(QUOTE PERM))
      (SETQ W (RDS X))
      A1 (COND((NOT(EQ(LOADEXP X)(QUOTE EOF)))(GO A1)))
      A2 (RDS W)(SHUT X)
      (RETURN Y ) ))) )) )
```

ECF CARD ALL PROGRAM COMPATIBLE

Library Routines written directly in Lisp 1.5

Library of MBTA in LISP 1.5 (Translated from version written in LISPX)

```
(1 CSET (STAR NIL))
(1 CSET (GEN NIL))
(1 CSET (SKIP*BLANKS NIL))
(1 CSET (LINE NIL))
(1 CSET (INPUT NIL))
(1 CSET (CCLUMN NIL))
(1 CSET (BACK NIL))
(1 CSET (CHR NIL))
(1 CSET (FLAGX NIL))
(1 CSET (LETTER NIL))
(1 CSET (DIGIT NIL))
(1 CSET (ODGT NIL))
(1 CSET (IC*V NIL))
(1 CSET (CCUNT NIL))
(1 CSET (MAXIMUM NIL))
(1 CSET (OK NIL))
(1 CSET (TRUE NIL))
(1 DEFINE (((INITIALIZE (LAMBDA (A B)
       (PROG NIL (COND ((NOT (EQUAL A (QUOTE TTY))) (GO M00001)))
        (RDS NIL)
        (GO M00002)
        M00001 (OPEN A (QUOTE DISC) (QUOTE PERM))
        (RDS A)
        M00002 (COND ((NOT (EQUAL B (QUOTE TTY))) (GO M00003)))
        (WRS NIL)
        (GO M00004)
        M00003 (COND ((NOT (EQUAL B (QUOTE CORE))) (GO M00005)))
        (OPEN (QUOTE SDCSDC) (QUOTE DISC))
        (WRS (QUOTE SDCSDC))
        (GO M00006)
        M00005 (OPEN B (QUOTE DISC))
        (WRS B)
        M00006 M00004 (SETQ STAR NIL)
        (SETQ GEN NIL)
        (SETQ SKIP*BLANKS T)
        (SETQ CCLUMN 0)
        (SETQ LINE (READLN))
        (SETQ INPUT LINE)
        (SETQ CHR (CAR INPUT))
        (SETQ BACK NIL)
        (SETQ CCUNT 1)
        (SETQ MAXIMUM 1)
        (SETQ OK T)
        (SETQ FLAGX NIL)
        (SETQ LETTER 2)
        (SETQ DIGIT 12) (SETQ ODGT 8) (GENCH1) (SETQ TRUE T))))))
(1 DEFINE (((COMPLETE (LAMBDA (Z A)
      (PROG (X)
        (COND ((NOT (NOT A)) (GO M00007)))
        (ERRORX)
        M00007 (SETQ X (RDS NIL))
        (COND ((NOT (NOT (EQUAL X (QUOTE TTY)))) (GO M00008)))
        (SHUT X)
        M00008 (SETQ X (WRS NIL))
        (COND ((NOT (AND (EQUAL X (QUOTE SDCSDC)) A)) (GO M00009)))
        (POSITION X (QUOTE WEOF))
        (POSITION X (QUOTE REWIND))
        M00010 (COND ((EQUAL (LOADEXP X) (QUOTE EOF)) (GO M00011)))
        (GO M00010)
        M00011 (SHUT X (QUOTE DELETE))
        (GO M00012)
        M00009 (COND ((NOT (NOT (EQUAL X (QUOTE TTY)))) (GO M00013)))
```

```
        (POSITION X (QUOTE WEOF))
        (SHIFT X) M00013 M00012 (RETURN A)))))))
(1 DEFINE (((READLN (LAMBDA NIL (PROG (X)
        (SETQ X (READCH))
        (COND ((NOT (NULL X)) (GO M00014)))
        (RETURN NIL)
        (GO M00015) M00014 (RETURN (CONS X (READLN))) M00015))))))
(1 DEFINE (((PRINTLN (LAMBDA (U)
      (PROG (X)
        (SETQ X (PRINTLN1 U))
        M00016 (COND ((NULL X) (GO M00017)))
        (PRINTCH (CAR X))
        (SETQ X (CDR X)) (GO M00016) M00017 (TERPRI))))))
(1 DEFINE (((NXTCHR (LAMBDA NIL (PROG NIL (COND ((NOT (NULL (CDR
          INPUT))) (GO M00018)))
        (SETQ COLUMN COUNT)
        (SETQ LINE (READLN))
        (RPLACD INPUT LINE)
        M00018 (SETQ INPUT (CDR INPUT))
        (SETQ CHR (CAR INPUT))
        (SETQ COUNT (ADD1 COUNT))
        (SETQ MAXIMUM (MAX COUNT MAXIMUM)))))))
(1 DEFINE (((LOAD (LAMBDA (X)
      (PROG NIL (SETQ STAR (CONS X STAR)) (RETURN T)))))))
(1 DEFINE (((COMPARE (LAMBDA (S N)
      (PROG (U V W SIGNAL)
        (COND ((NOT SKIP*BLANKS) (GO M00019)))
        M00020 (COND ((NOT (EQUAL CHR ' )) (GO M00021)))
        (NXTCHR)
        (GO M00020)
        M00021 M00019 (SETQ U S)
        (SETQ V INPUT)
        (SETQ W COUNT)
        M00022 (COND ((OR (NULL U) (NOT (EQUAL (CAR U) CHR)))
          (GO M00023)))
        (SETQ U (CDR U))
        (NXTCHR)
        (GO M00022)
        M00023 (SETQ SIGNAL (NULL U))
        (COND ((NOT (EQUAL N 3)) (GO M00024)))
        (SETQ SIGNAL (NOT SIGNAL))
        (SETQ INPUT V)
        (SETQ CHR (CAR INPUT))
        (SETQ COUNT W)
        (GO M00025)
        M00024 (COND ((NOT SIGNAL) (GO M00026)))
        (COND ((NOT (EQUAL N 2)) (GO M00027)))
        (SETQ STAR (CONS (APPEND (CAR STAR) S) (CDR STAR)))
        M00027 (GO M00028)
        M00026 (SETQ INPUT V)
        (SETQ CHR (CAR INPUT))
        (SETQ COUNT W) M00028 M00025 (RETURN SIGNAL))))))
(1 DEFINE (((CMPR2 (LAMBDA (X)
      (PROG NIL (COND ((NOT (CMPR X)) (GO M00029)))
        (SETQ STAR (CONS (COMPRESS X) STAR))
        (RETURN T) M00029 (RETURN F)))))))
(1 DEFINE (((CMPR (LAMBDA (S) (PROG NIL (RETURN (COMPARE S 1)))))))
(1 DEFINE (((COMPS (LAMBDA (S) (PROG NIL (RETURN (COMPARE S 2)))))))
(1 DEFINE (((NCOMP (LAMBDA (S) (PROG NIL (RETURN (COMPARE S 3))))))))
(1 DEFINE (((ERRORX (LAMBDA NIL (PROG (X)
        (SETQ X (WRS NIL))
        (PRINTLN LINE)
```

```
        (BLANKS (SUB1 (DIFFER MAXIMUM COLUMN)))
        (PRINTCH ' ) (TERPRI) (WRS X) (SETQ OK T) (RETURN T))))))
(1 DEFINE (((MARK (LAMBDA NIL (PROG NIL M00030 (COND ((NOT (EQUAL CHR
          ' )) (GO M00031)))
        (NXTCHR)
        (GO M00030)
        M00031 (SETQ SKIP*BLANKS F) (SETQ STAR (CONS NIL STAR)))))))
(1 DEFINE (((TCKEN (LAMBDA (W X)
       (PROG NIL (SETQ SKIP*BLANKS T)
        (COND ((NOT (NOT X)) (GO M00032)))
        (SETQ STAR (CDR STAR)) M00032 (RETURN X)))))))
(1 DEFINE (((INSERT (LAMBDA (S)
       (PROG NIL (SETQ STAR (CONS (APPEND (CAR STAR) S) (CDR STAR)))
        (RETURN T)))))))
(1 DEFINE (((STAR1 (LAMBDA NIL (PROG (X)
        (SETQ X (CAR STAR)) (SETQ STAR (CDR STAR)) (RETURN X)))))))
(1 DEFINE (((STAR2 (LAMBDA NIL (PROG (X)
        (SETQ X (CADR STAR))
        (SETQ STAR (CONS (CAR STAR) (CDDR STAR))) (RETURN X)))))))
(1 DEFINE (((STAR3 (LAMBDA NIL (PROG (X)
        (SETQ X (CADDR STAR))
        (SETQ STAR (CONS (CAR STAR) (CONS (CADR STAR) (CDDDR STAR))))
        (RETURN X)))))))
(1 DEFINE (((STAR4 (LAMBDA NIL (PROG (X)
        (SETQ X (CADDDR STAR))
        (SETQ STAR (CONS (CAR STAR)
          (CONS (CADR STAR) (CONS (CADDR STAR) (CDR (CDDDR STAR))))))
        (RETURN X)))))))
(1 DEFINE (((FLAGS (LAMBDA NIL (PROG NIL (SETQ FLAGX (CONS STAR FLAGX)
        ) (SETQ STAR NIL)))))))
(1 DEFINE (((SEQ (LAMBDA (W X)
       (PROG NIL (COND ((NOT X) (GO M00033)))
        (SETQ STAR (CONS (REVERSE STAR) (CAR FLAGX)))
        (GO M00034)
        M00033 (SETQ STAR (CAR FLAGX))
        M00034 (SETQ FLAGX (CDR FLAGX)) (RETURN X)))))))
(1 DEFINE (((GN1 (LAMBDA NIL (PROG NIL (COND ((NOT (NULL (CAR GEN)))
          (GO M00035)))
        (SETQ GEN (CONS (GENSYM) (CDR GEN)))
        M00035 (RETURN (CAR GEN)))))))
(1 DEFINE (((GN2 (LAMBDA NIL (PROG NIL (COND ((NOT (NULL (CADR GEN)))
          (GO M00036)))
        (SETQ GEN (CONS (CAR GEN) (CONS (GENSYM) (CDDR GEN))))
        M00036 (RETURN (CADR GEN)))))))
(1 DEFINE (((GEN1 (LAMBDA NIL (PROG NIL (SETQ STAR (CONS (GN1) STAR))
        (RETURN T)))))))
(1 DEFINE (((GEN2 (LAMBDA NIL (PROG NIL (SETQ STAR (CONS (GN2) STAR))
        (RETURN T)))))))
(1 DEFINE (((ANY (LAMBDA NIL (PROG NIL (SETQ STAR (CONS (APPEND (CAR
          STAR) (CONS CHR NIL)) (CDR STAR))) (NXTCHR) (RETURN T)))))))
(1 DEFINE (((DELETEX (LAMBDA NIL (PROG NIL (NXTCHR) (RETURN T)))))))
(1 DEFINE (((MAKEATOM (LAMBDA NIL (PROG NIL (SETQ STAR (CONS
         (COMPRESS (CAR STAR)) (CDR STAR))) (RETURN T)))))))
(1 DEFINE (((MAKENUMBER (LAMBDA NIL (PROG (S N)
        (SETQ S (CAR STAR))
        (SETQ N 0)
        M00037 (COND ((NULL S) (GO M00038)))
        (SETQ N (PLUS (TIMES N 10) (CHR2OCT (CAR S))))
        (SETQ S (CDR S))
        (GO M00037)
        M00038 (SETQ STAR (CONS N (CDR STAR))) (RETURN T)))))))
(1 DEFINE (((COMPILE (LAMBDA NIL (PROG NIL (PRINT (CONS 1 (CAR STAR)))
```

```
        (SETQ STAR (CDR STAR)) (RETURN T))))))
(1 DEFINE (((ISIT (LAMBDA (X Y)
      (PROG (SIGNAL)
        (SETQ SIGNAL (NOT (ZEROP (LOGAND X (CONVERT CHR)))))
        (COND ((NOT SIGNAL) (GO M00039)))
        (COND ((NOT Y) (GO M00040)))
        (SETQ STAR (CONS (APPEND (CAR STAR) (CONS CHR NIL))
          (CDR STAR))) M00040 (NXTCHR) M00039 (RETURN SIGNAL))))))
(1 DEFINE (((MAKECHR (LAMBDA NIL (PROG NIL (SETQ STAR (CONS (CAR (CAR
          STAR)) (CDR STAR))) (RETURN T))))))
(1 DEFINE (((WHERE (LAMBDA (X Y)
      (PROG (A B)
        (SETQ A ID*V)
        (SETQ ID*V Y) (SETQ B (X)) (SETQ ID*V A) (RETURN B))))))
(1 DEFINE (((CHECK (LAMBDA (A B)
      (PROG (X)
        (OPEN A (QUOTE DISC) (QUOTE PERM))
        (OPEN B (QUOTE DISC) (QUOTE PERM))
        M00041 (COND ((EQUAL (SETQ X (PROG2 (RDS A) (READ)))
          (QUOTE ECF)) (GO M00042)))
        (COND ((NOT (NOT (EQUAL X (PROG2 (RDS B) (READ)))))
          (GO M00043)))
        (PRINT (QUOTE BAD))
        M00043 (COND ((NOT (EQUAL (CADR X) (QUOTE DEFINE)))
          (GO M00044)))
        (PRINT (CAR (CAR (CAR (CADDR X)))))
        M00044 (GO M00041) M00042 (RDS NIL) (SHUT A) (SHUT B))))))
(1 DEFINE (((PRINTLN1 (LAMBDA (U)
      (PROG (X)
        (COND ((NOT (EQUAL U NIL)) (GO M00045)))
        (RETURN NIL)
        M00045 (SETQ X (PRINTLN1 (CDR U)))
        (COND ((NOT (AND (EQUAL X NIL) (EQUAL (CAR U) ' )))
          (GO M00046)))
        (RETURN NIL) M00046 (RETURN (CONS (CAR U) X)))))))
(1 DEFINE (((SAVER (LAMBDA NIL (PROG NIL (SETQ BACK (CONS (CONS INPUT
          (CONS STAR (CONS FLAGX (CONS COUNT NIL)))) BACK)))))))
(1 DEFINE (((RESTORE (LAMBDA (W X)
      (PROG NIL (SETQ BACK (CDR BACK)) (RETURN X))))))
(1 DEFINE (((BACKUP (LAMBDA (X)
      (PROG (A)
        (SETQ A (AND X OK))
        (COND ((NOT (NOT OK)) (GO M00047)))
        (SETQ INPUT (CAR (CAR BACK)))
        (SETQ STAR (CADR (CAR BACK)))
        (SETQ FLAGX (CADDR (CAR BACK)))
        (SETQ COUNT (CADDDR (CAR BACK)))
        (SETQ CHR (CAR INPUT)) (SETQ OK T) M00047 (RETURN A))))))
(1 DEFINE (((ENTER (LAMBDA NIL (PROG NIL (SETQ GEN (CONS NIL (CONS
          NIL GEN)))))))))
(1 DEFINE (((LEAVE (LAMBDA (X Y)
      (PROG NIL (SETQ GEN (CDDR GEN)) (RETURN Y))))))
(1 DEFINE (((PRINST (LAMBDA (X)
      (PROG (Z)
        (SETQ Z X)
        M00048 (COND ((EQUAL Z NIL) (GO M00049)))
        (PRINTCH (CAR Z)) (SETQ Z (CDR Z)) (GO M00048) M00049))))))
```

ECF CARD ALL PROGRAM COMPATIBLE