The LISP Implementation for the PDP-1 Computer

L. Peter Deutsch and Edmund C. Berkeley

March 1964

LISP
Appendix 5

The LISP Implementation for the PDP-1 Computer

L. Peter Deutsch and Edmund C. Berkeley

TABLE OF CONTENTS

Part I

1. Introduction
2. Functions and Properties Included in Basic PDP-1 LISP
3. Use of these Functions and Suggested Test Sequences
4. Auxiliary Functions which May Be Defined with LISP Expressions
5. Some Additional Functions for Basic PDP-1 LISP
6. Input and Output
7. Operation of the System
8. Error Diagnostics
9. Some Remarks

Part II

1. Macro Symbolic Program for Basic PDP-1 LISP
2. Alphabetic Listing of Defined Macro Symbols
3. Numeric Listing of the Defined Macro Symbols
4. Mnemonic Key or Derivation of Symbols
Part I

1. Introduction

In October 1963 a system for implementing LISP on the PDP-1 computer was finished by L. Peter Deutsch. This system was further improved in March 1964 by adding:

- variable length of push-down list;
- variable quantity of combined storage;
- optional machine language subroutines;

and is here called Basic PDP-1 LISP. It uses a minimum of some 2000 (decimal) registers out of 4096 registers in a one-core PDP-1 computer; it may use 16,361 registers in a four-core PDP-1 computer.

Basic PDP-1 LISP is presented in considerable detail in this appendix for the following reasons:

- the structure of a system for programming LISP on any computer is thereby revealed;
- if changes are to be implemented, they can be easily linked with the existing system.

In a one-core PDP-1 computer with 4096 registers, as many as 4070 registers may be assigned to regular LISP, and only 23 reserved for the read-in routine (namely, from 7751 to 7777, octal).

With the system described here, additional LISP functions can be defined and included in the system and later used when desired. Or if desired, additional functions can be programmed in machine language and these can be inserted compatibly with the system.

Punched tapes for placing this LISP system on the PDP-1 computer are available through DECUS, the Digital Equipment Corporation Users Organization, Maynard, Mass.

In the following, it is assumed that the reader has a fairly good working knowledge of: (1) LISP (which may be obtained from the "LISP 1.5 Programmer's Manual," 1962); (2) the machine language codes for the PDP-1 computer (which may be obtained from the computer manual supplied by Digital Equipment Corporation); and (3) the program assembly language MACRO, in which the sym-
bolic tapes are written (a description may be obtained in two manuals published by Digital Equipment Corporation).

2. Functions and Properties included in Basic PDP-1 LISP

The functions and properties included in Basic PDP-1 LISP are shown in Table 1. These functions and properties together constitute a basic subset of the functions and properties of the LISP interpreter for the IBM 7090, as stated in the LISP 1.5 Programmer's Manual.

In order to obtain other LISP functions and properties as may be desired for any particular purpose, see Sections 4 and 5 below.

Table 1
FUNCTIONS AND PROPERTIES OF BASIC PDP-1 LISP

A. Functions Identical with the Corresponding IBM 7090 LISP Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>IBM 7090 LISP Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATOM</td>
<td>LIST</td>
</tr>
<tr>
<td>CAR</td>
<td>LOGAND</td>
</tr>
<tr>
<td>CDR</td>
<td>LOGOR</td>
</tr>
<tr>
<td>COND</td>
<td>MINUS</td>
</tr>
<tr>
<td>CONS</td>
<td>NULL</td>
</tr>
<tr>
<td>EVAL</td>
<td>NUMBERP</td>
</tr>
<tr>
<td>GENSYM</td>
<td>PLUS</td>
</tr>
<tr>
<td>GO</td>
<td>PRINT</td>
</tr>
<tr>
<td></td>
<td>PROG</td>
</tr>
<tr>
<td></td>
<td>QUOTE</td>
</tr>
<tr>
<td></td>
<td>READ</td>
</tr>
<tr>
<td></td>
<td>RETURN</td>
</tr>
<tr>
<td></td>
<td>RPLACA</td>
</tr>
<tr>
<td></td>
<td>RPLACD</td>
</tr>
<tr>
<td></td>
<td>SASSOC</td>
</tr>
<tr>
<td></td>
<td>SETQ</td>
</tr>
<tr>
<td></td>
<td>TERPRI</td>
</tr>
</tbody>
</table>

B. Functions Somewhat Different from the Corresponding 7090 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>This works both on atoms and on numbers</td>
</tr>
<tr>
<td>GREATERP</td>
<td>This tests for X greater than Y, not for X greater than or equal to Y.</td>
</tr>
<tr>
<td>STOP</td>
<td>This is equivalent to PAUSE in 7090 LISP. It takes a numerical argument which appears in the accumulator when the computer halts.</td>
</tr>
<tr>
<td>PRIN1 X</td>
<td>This prints the atom X without the extra space at the end. Its value is NIL.</td>
</tr>
</tbody>
</table>
C. Functions Which Have No Analog in 7090 Functions

XEQ This provides for putting into storage a named machine language subroutine, which can be referred to and used by the PDP-1 LISP interpreter. It also provides for executing single specified machine language instructions.

The SUEF (XEQ C A I) executes the machine language instruction C, with A in the accumulator and I in the in-out register; and returns a value in the form of (a i P) where a is the new value of the accumulator after execution, i is the new value of the in-out register after execution, and P is T if the instruction skipped, and NIL if the instruction did not skip.

LOC X This gives the machine register in which the atom or list X begins; its value is the location.

Of the foregoing functions, COND, LIST, PROG, SETQ, PLUS, TIMES, LOGAND, LOGOR, and QUOTE are FSUBRs and the remainder are SUFRs.

D. The following special form is available and is identical with the corresponding form in 7090 LISP:

LAMBDAs

E. The following permanent objects exist in the Basic PDP-1 LISP system:

OBLIST the current list of atomic symbols
NIL F has been replaced by NIL
T
EXPR
SUBR
EXPR
F:SUBR
AFVAL

F. Miscellaneous

The print names of atomic symbols are not part of property lists. A quick examination of listings of the system will show exactly where the print names are.

Doing a CDR of an atom is permissible and will get the atom's property list. Doing a CAR of an atom may very easily wreck the system.

QUOTE should be used in place of 7090 FUNCTION. This may re-
quire a bit of extra care in defining functions with functional arguments.

It is advisable to use PROG to avoid recursion wherever possible, even though it may take more space.

3. Use of these Functions and Suggested Test Sequences

How to use these functions is briefly explained here.

As soon as the basic PDP-1 LISP system is read into the computer, control stops at register 4. Turn up sense switch 5 for typewriter input; press CONTINUE; and the system enters a waiting loop which causes lamps to light in the program counter, looking like 1335. At this point, the LISP system is ready for manual typewriter input. As soon as the operator types, for example:

(CAR (QUOTE (A B C D)))

together with a final space at the end of the last right parenthesis, the computer takes control of the typewriter, impulses a carriage return, and then types out:

A

which of course is the correct answer. Similarly, for the other suggested test sequences in Table 2 below.

Table 2

<table>
<thead>
<tr>
<th>SUGGESTED TEST SEQUENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
<tr>
<td>(CAR (QUOTE (A B C D)))</td>
</tr>
<tr>
<td>(CDR (QUOTE (A B C D)))</td>
</tr>
<tr>
<td>OBLIST</td>
</tr>
<tr>
<td>(LIST (QUOTE (A B C D)))</td>
</tr>
</tbody>
</table>
NIL

(CDR NIL)

(CAR (QUOTE (T.NIL)))

(CONS (ATOM (CDR T)) (LIST (GENSYM) (GENSYM)))

(COND (EQ T NIL) (STOP 1))
   (T (EQ (PLUS 1 1) 2)))

(PROG (U) (PRINT NIL) (TERPRI)
   (PRINT T) (SETQ U T)
   (RETURN U))

(RPLACED (QUOTE CAAR) (QUOTE (EXPR (LAMHDA (X) (CAR (CAR X))))))

(CAAR (QUOTE ((A))))

(STOP 2)

(PRIN1 (QUOTE CAR))

(PRINT X)

(TERPRI)

(LOC NIL)

(LOC (QUOTE COND))

(LOGAND 6 7 3)

(LOGOR 12 3 15)
Suppose the computer contains DDT — DDT is short for "Digital Equipment Corp. Debugging Tape"; its starting register is 6000, and in one of its customary forms it uses registers 5540 to 7750. Then, if the highest storage register of LISP is below 5540, the instruction:

\[(\text{XEQ } 606000 \ 0 \ 0)\]

transfers control to DDT, and puts zero in the accumulator and in the in-out register.

If there is the following subroutine stored in the computer:

<table>
<thead>
<tr>
<th>Register</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>5500</td>
<td>dzm 5507</td>
</tr>
<tr>
<td>5501</td>
<td>idx 5507</td>
</tr>
<tr>
<td>5502</td>
<td>lac 5507</td>
</tr>
<tr>
<td>5503</td>
<td>dpv'</td>
</tr>
<tr>
<td>5504</td>
<td>sma</td>
</tr>
<tr>
<td>5505</td>
<td>jmp 5501</td>
</tr>
<tr>
<td>5506</td>
<td>jmp 2241</td>
</tr>
<tr>
<td>5507</td>
<td>(being used for storage)</td>
</tr>
</tbody>
</table>

and LISP is below 5500, then:

\[(\text{XEQ } 605500 \ 0 \ 0)\]

Will cause a horizontal line to be drawn on the scope from the origin to the x-axis positive limit, and then control will be returned to LISP. NIL will be typed out. 2241 is the register called "prx" in the macro symbolic.

4. Auxiliary Functions Which May Be Defined with LISP Expressions

Any of the functions listed below in Table 3 can be put into the system at will, as follows: Prepare a punched tape listing of it. Insert tape into the reader. Turn on the reader. Turn down Sense Switch 5. Thereupon the computer will read in the
tape. The typewriter, when the reading in is accomplished, will type back the name of the inserted function.

Many other functions besides those listed in Table 3 may be inserted.

Table 3

AUXILIARY LISP FUNCTIONS

**ABSOLUTE VALUE**

\[
(RPLACD (QUOTE ABSVAL) (QUOTE (EXPR (LAMBDA (X) (COND ((GREATERP O X) (MINUS X)) (T X)))))
\]

**AND**

\[
(RPLACD (QUOTE AND) (QUOTE (FEXPR (LAMBDA (X A) (PROG NIL N (COND ((NULL X) (RETURN T)) ((NULL (EVAL (CAR X) A)) (RETURN NIL))) (SETQ X (CDR X)) (GO N))))))
\]

**ASSOC**

\[
(RPLACD (QUOTE ASSOC) (QUOTE (EXPR (LAMBDA (X Y) (COND ((EQUAL (CAAR Y) X) (CAR Y)) (T (ASSOC X (CDR Y)))))))
\]

**CAAR**

\[
(RPLACD (QUOTE CAAR) (QUOTE (EXPR (LAMBDA (X) (CAR (CAR X))))))
\]

**CDR**

\[
(RPLACD (QUOTE CDR) (QUOTE (EXPR (LAMBDA (X) (CAR (CDR X))))))
\]

**CDAR**

\[
(RPLACD (QUOTE CDAR) (QUOTE (EXPR (LAMBDA (X) (CDR (CAR X))))))
\]

**CDDR**

\[
(RPLACD (QUOTE CDDR) (QUOTE (EXPR (LAMBDA (X) (CDR (CDR X))))))
\]

**CSET**

\[
(RPLACD (QUOTE CSET) (QUOTE (EXPR (LAMBDA (X Y) (RPLACD X (LIST (QUOTE APROV) Y))))))
\]
CSETQ
(RPLACD (QUOTE CSETQ) (QUOTE (FEXPR (LAMBDA (X A) (CSET (CAR X) (EVAL (CADR X) A))))))

DEX
(RPLACD (QUOTE DEX) (QUOTE (FEXPR (LAMBDA (X A) (RPLACD (CAR X) (CONS (QUOTE FEXPR) (CDR X))))))

DFX
(RPLACD (QUOTE DFX) (QUOTE (FEXPR (LAMBDA (X A) (RPLACD (CAR X) (CONS (QUOTE FEXPR) (CDR X))))))

DIFFLIST
(RPLACD (QUOTE DIFFLIST) (QUOTE (EXPR (LAMBDA (A X) (COND ((NULL X) NIL) ((EQUAL A (CAR X)) (DIFFLIST A (CDR X))) (T (CONS (CAR X) (DIFFLIST A (CDR X))))))))

DOUBLE
(RPLACD (QUOTE DOUBLE) (QUOTE (EXPR (LAMBDA (X) (PLUS X X)))))

EQUAL
(RPLACD (QUOTE EQUAL) (QUOTE (EXPR (LAMBDA (X Y) (COND ((ATOM X) (EQ X Y)) ((ATOM Y) NIL) ((EQUAL (CAR X) (CAR Y)) (EQUAL (CDR X) (CDR Y))) (T NIL))))))

GREATEST COMMON DIVISOR
(RPLACD (QUOTE GCD) (QUOTE (EXPR (LAMBDA (X Y) (COND ((GREATERP X Y) (GCD Y X)) ((ZEROP (REM Y X)) X) (T (GCD (REM Y X) X))))))

LAST
(RPLACD (QUOTE LAST) (QUOTE (EXPR (LAMBDA (L) (COND ((NULL L) NIL) ((NULL (CDR L)) (CAR L)) (T (LAST (CDR L))))))))

LENGTH using Program Feature
(RPLACD (QUOTE LENGTH) (QUOTE (EXPR (LAMBDA (L) (PROG (U V) (SETQ V 0) (SETQ U L) A (COND ((NULL U) (RETURN V))) (SETQ U (CDR U)) (SETQ V (PLUS 1 V)) (GO A))))))

LENGTH using Recursion
(RPLACD (QUOTE LENGTH) (QUOTE (EXPR (LAMBDA (L) (COND ((NULL L) 0) (T (PLUS 1 (LENGTH (CDR L))))))))

MAPLIST using Recursion
(RPLACD (QUOTE MAPLIST) (QUOTE (EXPR (LAMBDA (X A) (COND ((NULL X) NIL) (T (CONS (A X) (MAPLIST (CDR X) A)))))))

MAPLIST using Program Feature
(RPLACD (QUOTE MAPLIST) (QUOTE (FEXPR (LAMBDA (X A) (PROG (V M R)))))

9
(SETQ R (SETQ M (LIST (EVAL (CADR X) A)))) (SETQ V (EVAL (CAR X) A)) P (COND ((NULL V) (RETURN (CDR R)))) (SETQ M (CDR (RPLACD M (LIST (EVAL (LIST (CAR R) (LIST (QUOTE QUOTE) V)) A)))) (SETQ V (CDR V)) (GO P)))))

MEMBER
(RPLACD (QUOTE MEMBER) (QUOTE (EXPR (LAMBDA (A X) (COND ((NULL X) NIL) ((EQ A (CAR X)) T) (T (MEMBER A (CDR X)))))))))

MINIMUM
(RPLACD (QUOTE MIN) (QUOTE (EXPR (LAMBDA (L) (COND ((NULL L) NIL) ((NULL (CDR L)) (CAR L)) (T (SMALLER (CAR L) (MIN (CDR L)))))))

NOT
(RPLACD (QUOTE NOT) (QUOTE (EXPR NULL)))

OR
(RPLACD (QUOTE OR) (QUOTE (FEXPR (LAMBDA (X A) (PROG NIL N (COND ((NULL X) (RETURN NIL)) ((EVAL (CAR X) A) (RETURN T))) (SETQ X (CDR X)) (GO N))))))

PAIR
(RPLACD (QUOTE PAIR) (QUOTE (EXPR (LAMBDA (X Y) (PROG (U V M) (SETQ U X) (SETQ V Y) (SETQ M NIL) K (COND ((NULL U) (COND ((NULL V) (RETURN M)))) (SETQ M (CONS (CONS (CAR U) (CAR V)) M)) (SETQ U (CDR U)) (SETQ V (CDR V)) (GO K))))))

PAIRLIS
(RPLACD (QUOTE PAIRLIS) (QUOTE (EXPR (LAMBDA (X Y A) (COND ((NULL X) A) (T (CONS (CONS (CAR X) (CAR Y)) (PAIRLIS (CDR X) (CDR Y) A))))))

PDEF (Print and Punch Definition)
(RPLACD (QUOTE PDEF) (QUOTE (EXPR (LAMBDA (X A) (LIST (QUOTE (RPLACD) (LIST (QUOTE QUOTE (CAR X)) (LIST (QUOTE QUOTE (CAR (CAR X)))))))))

QUOTIENT using Program Feature
(RPLACD (QUOTE QUOTIENT) (QUOTE (EXPR (LAMBDA (Q D) (PROG (U V) (SETQ V 0) (SETQ U Q) A (COND ((GREATERP D U) (RETURN V))) (SETQ U (PLUS U (MINUS D))) (SETQ V (PLUS V 1)) (GO A)))))

QUOTIENT using Recursion
(RPLACD (QUOTE QUOTIENTR) (QUOTE (EXPR (LAMBDA (Y X) (COND ((GREATERP X Y) 0) ((EQ X Y) 1) ((GREATERP Y X) (PLUS 1 (QUOTIENTR (PLUS Y (MINUS X) X))))))))
RENDER
(REPLACE (~REDEF ~REPLACE (QUOTE REPLACEMENT) (QUOTE (EXPR (LAMBDA (X) (COND ((EQUAL X 0) ((GREATERP X Y) Y) (T (REPLACEMENT (PLUS X (MINUS X) X)))))))

REVERSE (Defined Recursively with Auxiliary Function)
(REPLACE (QUOTE REVERSE) (QUOTE (EXPR (LAMBDA (M) (COND ((NULL M) M) (T (REVERSE (CONS (CAR M) M) (CDR M)))))))

(REPLACE (QUOTE REVERSE) (QUOTE (EXPR (LAMBDA (L) (REVERSE NIL L))))

REVERSE using Program Feature
(REPLACE (QUOTE REVERSE) (QUOTE (EXPR (LAMBDA (M) (PROG (U V) (SETQ U M) K (COND ((NULL U) (RETURN V)) (SETQ V (CONS (CAR U) V)) (SETQ U (CDR U)) (GO K))))))

SEQUENCE
(REPLACE (QUOTE SEQUENCE) (QUOTE (EXPR (LAMBDA (L) (PROG (U V W) (SETQ U L) (SETQ V (MIN L)) (SETQ W NIL) A (COND ((NULL U) (RETURN W))) (SETQ V (MIN U)) (SETQ U (DIFFLIST V U)) (SETQ W (APPEND W (LIST V)) (GO A))))))

SMALLER
(REPLACE (QUOTE SMALLER) (QUOTE (EXPR (LAMBDA (X Y) (COND ((GREATERP X Y) Y) (T X))))))

SUB2
(REPLACE (QUOTE SUB2) (QUOTE (EXPR (LAMBDA (A Z) (COND ((NULL A) Z) ((EQ (CAAR A) Z) (CDAR A)) (T (SUB2 (CDR A) Z))))))

SUBLIS
(REPLACE (QUOTE SUBLIS) (QUOTE (EXPR (LAMBDA (A Y) (COND ((ATOM Y) (SUB2 A Y)) (T (CONS (SUBLIS A (CAR Y)) (SUBLIS A (CDR Y)))))))))

SUBST
(REPLACE (QUOTE SUBST) (QUOTE (EXPR (LAMBDA (X Y Z) (COND ((EQUAL Y Z) X) ((ATOM Z) Z) (T (CONS (SUBST X Y (CAR Z)) (SUBST X Y (CDR Z))))))))

TIMES using Recursion
(REPLACE (QUOTE TIMES) (QUOTE (EXPR (LAMBDA (N M) (COND ((EQUAL N 1) M) (T (PLUS M (TIMES M (PLUS N (MINUS 1)))))))))))

11
TIMES using Program Feature
(RPLACD (QUOTE TIMES) (QUOTE (EXPR (LAMBDA (X N) (PROG (U V)
               (SETQ V 0) (SETQ U 0) A (COND ((EQ V N) (RETURN U))) (SETQ U
               (PLUS X U)) (SETQ V (PLUS V 1)) (GO A))))))

UNION
(RPLACD (QUOTE UNION) (QUOTE (EXPR (LAMBDA (X Y) (COND ((NULL X)
               Y) ((MEMBER (CAR X) Y) (UNION (CDR X) Y)) T (CONS (CAR X)
               (UNION (CDR X) Y))))))

ZEROP
(RPLACD (QUOTE ZEROP) (QUOTE (EXPR (LAMBDA (X) (COND ((EQUAL X
               0) T) (T NIL))))))

5. Some Additional Functions for Basic PDP-1 LISP

In order to remove symbols from the OELIST, and reuse the
storage capacity that they previously occupied, we use:

(RPLACD (QUOTE XSY) (QUOTE (EXPR (LAMBDA (X) (PROG (Y) (SETQ Y
               OELIST) A (COND ((NULL (CDR Y)) (RETURN NIL)) ((EQ X (CAR
               (CDR Y))) (RETURN (RPLACD Y (CDR (CDR Y)))))) (SETQ Y (CDR Y))
               (GO A))))))

(RPLACD (QUOTE REMOVE) (QUOTE (EXPR (LAMBDA (X Y) (PROG NIL A
               (COND ((NULL X) (RETURN OELIST))) (XSY (CAR X)) (SETQ X
               (CDR X)) (GO A))))))

XSY stands for "expunge symbol".

REMOVE is used as follows: Suppose we have a case where the
OELIST starts for example with G F OBLITT Y X ATOM CAR CDR COND
CONS ..... and we wish to delete F OBLITT Y. We put in: REMOVE
OBLITT F Y), and the computer response is:

G X ATOM CAR CDR CONS ..... 

In this way, both accidentally mistyped expressions and sym-
bols no longer needed in the LISP system can be removed from stor-
age, and from any recollection within the LISP system. (Note:
REMOVE will not operate on the first expression in the OELIST,
but only on the second and later expressions.)

In order to put in machine-language subroutines, outside of
the storage used by LISP, name them, use them, and return from
them to LISP, we use:
The **EDR** (DEFOSIT X A) deposits the list of numbers \( X \) starting at location \( A \); its value is the first register beyond the list.

The **EXPR** (PUTSUBR \( N \times X \times A \)) performs (DEFOSIT \( X \times A \)), and then sets up \( N \) (name) as a **SUBR** starting at \( A \).

An example (if LISP storage stops at 5477) is:

```
(PUTSUBR (QUOTE SHOWLINE) (LIST 345507 445507 205507 730007 640400 605501 602241) 5500)
```

This inserts the line-display program mentioned above into the computer starting at register 5500 and makes it accessible to LISP with the name **SHOWLINE**.

The **EXPR** (DEFSUBR \( N \times X \)) accepts an existing, inserted, machine-language subroutine starting at register \( X \), gives it the name \( N \), and makes it accessible to LISP with the name \( N \). For example, the line-display program mentioned above, if already in the computer, could be named and called with:

```
(DEFSUBR (QUOTE SHOWLINE) 5500)
```

The last command in the subroutine, instead of 602241, should be either 600004, if LISP is to return to the starting address 4, or 600005, if LISP is to continue to the waiting loop.

If the **A-LIST** is wanted, establish **GETALIST** with:

```
(RPLACD (QUOTE GETALIST) (QUOTE (FEXPR (LAMDA (X Y) Y)))))
```

and then use:

```
(PRINT (GETALIST))
```
6. Input and Output

Input comes from the typewriter if sense switch 5 is up and from the tape reader otherwise. Output is normally on the typewriter; however, SS 3 up causes punching (with correct parity) and SS 6 up independently suppresses typeout.

Each S-expression typed in will be evaluated and its value printed out. Unlike 7090 LISP, arguments of functions are also evaluated on the top level; for example, to evaluate

```
cons [A;B]
```

it is necessary to write

```
(CONS (QUOTE A) (QUOTE B))
```

In preparing input:

- Tab, space, and comma are equivalent;
- Carriage return is ignored;
- Backspace causes deletion of everything typed since the last control character (parenthesis, space/tab/comma, or period);
- An extra space must be typed to terminate the entire expression;
- Upper and lower case shifts will be noted but not necessarily inserted into the symbol at that point (for example, the sequence u.c., l.c., u.c., A, space, produces a symbol with print name u.c., A, l.c.);
- Alphabetic characters should regularly and generally be in lower case; and basic functions, (such as CAR, CDR, ...), contrary to their representation throughout this report, are in PDP-1 LISP actually stored in lower-case symbols (such as car, cdr ...); and then taken in to the system and put out by the system as lower-case symbols;
- It is very advisable to stick to "printout" format for all input since the READ routine is not guaranteed to work on any other form, although it may;
- Hyphen, "-", is a letter and does not negate a following number;
- All numbers are octal integers; to input the number -1 it is necessary to type 777776;
- There is no limit on the length of a print name;
- The character overbar "-" or vertical bar "|" will cause the next character to be inserted in the print name and considered a letter, regardless of what it actually is (the "-" or "|" itself does not appear
in the print name): thus atoms may be generated for output formatting purposes with names such as "tab" or "space".

In producing the output:

A carriage return is automatically generated after any 100 (octal) characters not containing a carriage return;
Unlike the 7090 LISP output, no spaces are provided before and after the ",," of concatenation (since there are no floating-point numbers to be concerned with).

7. Operation of the System

First, zero core, to avoid unnecessary difficulties.

Second, put the binary tape in the reader, and press READIN. Do nothing until the tape stops. Almost all of the tape will read in; and the machine will come to a halt. If you wish 7701 to be the highest register of free storage, and 300 to be the length of the push-down list, press READIN once more. The machine will stop at address 4. Turn up Sense Switch 5 (to control from the typewriter). Press CONTINUE.

If you wish to select the highest register of free storage, when the machine stops for the first time, with memory address at 0004, put the number of the highest register of free storage (recommended, 5000 to 7750; possible but not recommended, 4000 to 4777) in the Test Word switches and press CONTINUE. Then put the length of the push down list (suggested 200 to 400) in the Test Word switches, and press CONTINUE. The machine will go to address 4. Turn up Sense Switch 5, and press CONTINUE. The LISP system should be ready for use.

If the tape stops at an improper place, pull the tape back a block, check for missing holes, and CONTINUE. When the tape stops at 4, CONTINUEing begins the READ-EVAL-PRINT cycle. STARTing at 4 at any time and CONTINUEing is safe; indeed, it is the only way to annul most typing errors.

If the system "drops dead", the normal recourse is to start over.

Following is the assignment of the sense switches and the program flags:
8. Error Diagnostics

Error halts cause identification of the error and typing of the error code in red on the typewriter, regardless of the settings of Sense Switches 3 and 6; an error usually sends the system to address 4. The list of error indications follows:

- icd Illegal \texttt{COND}; returns value \texttt{NIL} and continues.
- uss Unbound symbol in \texttt{SEQQ}; returns \texttt{NIL} and continues.
- tma Too many arguments for a \texttt{SUBR} (more than 3); ignores extra arguments and proceeds.
- uas Unbound atomic symbol (followed by the form currently being evaluated).
- ilp Illegal parity; halts with character in accumulator. \texttt{CONTINUE} ignores character, but SS 5 may be turned up, and typing used to provide a replacement if desired.
- lts \texttt{LAMBDA} variable list too short.
- ats Argument list (paired with \texttt{LAMBDA} list) too short.
- sce Storage capacity exceeded. \texttt{CONTINUEing} is not advisable, as it will probably call the same error again in short order, unless one promptly deletes several atoms having lengthy definitions from the OELIST.
- pce Pushdown capacity exceeded.
- nma Non-numeric argument for arithmetic, followed by the argument in question; returns value zero and proceeds.
- ana Argument not atom (for \texttt{PRINT}); returns \texttt{NIL} as usual and proceeds.
- ovf Division overflow; returns zero and proceeds.
9. Some Remarks

In general, each character in each LISP expression is recognized by the computer as 2 octal digits called concise code. The pairs of octal digits are packed 3 pairs at a time into the 6-octal-digit registers of the PDP-1. If a LISP atom has a number of characters which is not a multiple of three, there will be spaces left over, which are filled arbitrarily with a filler character, 76 (octal). For example, a LISP word with 7 characters such as SMALLER will be packed into three computer registers, S M A in one, L L E in a second, and R along with two filler characters in the third.

These three registers are linked by list structure. An example of a hypothetical list structure which might store SMALLER if introduced as a defined function into the LISP system would be as shown in Table 4:

<table>
<thead>
<tr>
<th>PDP-1 Register</th>
<th>Contents</th>
<th>Meaning</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>5763</td>
<td>405765</td>
<td>pointer to 5765</td>
<td>5765 is the start of the print name of the atom SMALLER</td>
</tr>
<tr>
<td>5764</td>
<td>005773</td>
<td>pointer to property list</td>
<td>5773 is the start of the property list</td>
</tr>
<tr>
<td>5765</td>
<td>224461</td>
<td>S M A</td>
<td></td>
</tr>
<tr>
<td>5766</td>
<td>005767</td>
<td>pointer</td>
<td>5767 holds continuation of the list</td>
</tr>
<tr>
<td>5767</td>
<td>434365</td>
<td>L L E</td>
<td></td>
</tr>
<tr>
<td>5770</td>
<td>005771</td>
<td>pointer</td>
<td>5771 holds continuation of the list</td>
</tr>
<tr>
<td>5771</td>
<td>767651</td>
<td>- - R</td>
<td></td>
</tr>
<tr>
<td>5772</td>
<td>003011</td>
<td>nil</td>
<td></td>
</tr>
</tbody>
</table>

Table 4

Concise code
5767 holds continuation of the list
Concise code
5771 holds continuation of the list
Concise code and 2 filler characters
Terminator of list
If SMALLER were defined by the expression:

\[
\text{(RPLACD (QUOTE SMALLER) (QUOTE (EXPR (LAMBDA (X Y)
\text{(COND ((GREATERRP Y X) X) (T Y)))))))}
\]

then the property list of SMALLER would be (hypothetically) as shown in Table 5:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>5773</td>
<td>003271</td>
<td>&quot;EXPR&quot;</td>
</tr>
<tr>
<td>5774</td>
<td>005775</td>
<td>pointer</td>
</tr>
<tr>
<td>5775</td>
<td>005777</td>
<td>pointer</td>
</tr>
<tr>
<td>5776</td>
<td>002651</td>
<td>&quot;NIL&quot;</td>
</tr>
<tr>
<td>5777</td>
<td>003255</td>
<td>&quot;LAMBDATA&quot;</td>
</tr>
<tr>
<td>6000</td>
<td>006001</td>
<td>pointer to forking</td>
</tr>
<tr>
<td>6001</td>
<td>006003</td>
<td>pointer to (X Y)</td>
</tr>
<tr>
<td>6002</td>
<td>006007</td>
<td>pointer to (COND ....</td>
</tr>
<tr>
<td>6003</td>
<td>007701</td>
<td>&quot;X&quot;</td>
</tr>
<tr>
<td>6004</td>
<td>006005</td>
<td>pointer</td>
</tr>
<tr>
<td>6005</td>
<td>007711</td>
<td>&quot;Y&quot;</td>
</tr>
<tr>
<td>6006</td>
<td>002651</td>
<td>&quot;NIL&quot;</td>
</tr>
<tr>
<td>6007</td>
<td>002725</td>
<td>&quot;COND&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>etc.</td>
</tr>
</tbody>
</table>

An accepted LISP expression L is identified within the machine by the address of the list structure in storage which represents L.

The computer evaluates expressions using either machine subroutines (SUBRs and FSUBRs) or LISP subroutines (EXPRs or FEXPRs).

The computer converts the resulting value into concise codes, and presents the value for output to the computer-associated typewriter or the punch.

Basic PDP-1 LISP is very flexible:

1. The number of registers on the push-down list can be reasonably varied between 200 and 400 octal. The number chosen can vary according to the amount of recursion it is desired to provide for.
2. The number of registers of storage (there is only one kind of storage) can be varied from under 1000 octal to over 4000 octal in a one-core machine. In the smallest extreme case, LISP system can occupy only the registers up to about 4000 octal; in the other extreme case LISP can occupy all the registers up to 7750 octal, leaving 7751 to 7777 for the read-in subroutine.

3. Machine subroutines may be located in core, and referred to and used. These machine subroutines should be located above the highest register in free storage.

4. DDT (the Digital Debugging Tape) may be loaded in registers 5500 up and LISP may be loaded below, so that the facilities of DDT are available for modifying LISP.

5. A core dump routine may be loaded into 400 (octal) registers above free storage and used upon LISP.
Part II
1. Macro Symbolic Program for Basic PDP-1 LISP

lisp 3-23-64 : 1 field
define
  terminate
define
  load A,B
  law B
  dac A
decline
define
  init A,B
  law B
  dap A
define
  index A,B,C
  idx A
  sas B
  jmp C
define
  step A,B,C
  index A,(B,C)
define
  setup A,B
  law i B
  dac A
define
  exit
  jmp R
define
  move A,B
  lac A
  dac B
load A,B
move (B,A)
define
  terminate
define
  1sp A
  jmp B
define
  test K,P
  sad (K
  jmp P
define
  undeck A
  law 1 1
  add A
  dac A
define
  swap
  rcl 9s
define
  move A,B
  start
  smi=spi 1
  sza=sma-szf
  spq=szm 1
  xy=0
  xx=halt
  clo=spa sma szo 1-szf-szf
  mul=540000
  div=560000
Lisp interpreter 3-20-64, part 1

4/

go,

hlr+cla+cl1+7-oer-opr
stf 6
extend
dzm 77
law 77
dap avx

beg,

law pdo-1
dac pdl
lac n
dac ar2
cal rin
cal evo
cal ont
jmp beg
t0, 0
t1, 0
go, 0
g1, 0
hi, 0
csi, 72
cso, 72
ffi, 0
gai, 0
0 1si, 1si-1
gst, repeat 5,20
ao, 0
ai, 0
a2, 0

/append word to pdl

pwl, 0
dap pwx
idx pdl
sad bfw
jmp qg2
lac pdl
dac 1 pdl

pwx, exit

/retrieve word from pdl

uw, 0
dap uwx
llo i pdl
undex pdl

uwx, exit
lac 1 100
rcl 8s
rcl 8s
jmp x

/get two values
vad,
dio a1
cal vag
dac a0
lac a1
cal vag
dac a1
jmp x

/pack character onto end of buffer
oc,
rar 6s
lio 1 isi
rcl 6s
sad (76
jmp oc1
lac 100
ior (76;7600
cal cf
lio t0
idx t0
idx isi
dac a1
dio isi
lac 1 a1
dac 1 to
dio 1 a1
jmp x

oc1,
dio 1 isi
jmp x

/output routine
out,
lio 100
ssz 36
ppa
ssz 1 66
ty0
jmp x

/error printout
err,
cif 6
dap exr
lac 1 exr
dac ern
law ern
cal pra
stf 6
idx exr
exr,
exit
erm,
357776

ern,
0
.r1
347776
n,fro,
nil
define error F
csp err
F
termin

/garbage collector, non-compacting
gc,
dap gcx
dio ga1
dio gfr
lac gfr
sar 2s
sza
jsp gfr+1
lac f1
sza i
jmp gco
lac 100
jda gfr

gco,
lac 1 i0b
jda gfr
lac isi
sas (isi-1
jmp gci
law pd1+1
dac g1

gcp,
lac 1 g1
jda gfr
idx g1
sub (1
sad pd1
jmp g2e
jmp gcp

/mark one list
gfr,
0
dap gfx
lac gfr
ral 1s
spq
jmp gfx
lac pd1
jda pd1

gfn,
lio 1 gfr
idx gfr
lac 1 gfr
spa
jmp gfu
lor (add
dac 1 gfr
spi
jmp gfd
jda pd1
/SASSOC

aso, cal asc
jmp x
jmp asc
lac a2
lac na
cal cns-1
jmp evo

ase, lio ar2
asc, dio ar1
lacr ar1

as1, sad n
jmp x
lac i ar1
dac to
lac i to
sad 100
jmpl sac2
idx ar1
lac i ar1
dac ar1
jmpl sac1

as2, idx i pdl
lac to
jmpl x

/garbage collector, linear sweep phase

/garbage collector, linear sweep phase

/SASSOC

aso, cal asc
jmp x
jmp asc
lac a2
lac na
cal cns-1
jmp evo

ase, lio ar2
asc, dio ar1
lacr ar1

as1, sad n
jmp x
lac i ar1
dac to
lac i to
sad 100
jmpl sac2
idx ar1
lac i ar1
dac ar1
jmpl sac1

as2, idx i pdl
lac to
jmpl x

/program feature

/PROG

pgm, lac pa3
jda pdl
lac pa4
jda pdl
dzm pa4
dac ar3
lio ar2
lio 1 100
idx 100
lac 1 100
dac pa3
dac ar1

/pg5, sad n
jmpl pg6
lac i ar1
cal cns-1
lio ar2
cal cns
dac ar2
idx ar1
lac i ar1
dac ar1
jmpl pg5

/lproc

/lproc

/lproc

/lproc
/expand go-list (on a-list)

pg6,  lac pa3

pg7,  dac ar1
      sad n
      jmp pg6
      lac i ar1
      cal car
      sma
      jmp pg9
      lac ar1
      lio ar2
      cal cns
      dac ar2

pg9,  idx ar1
      lac i ar1
      jmp pg7

/process program

pg0,  lac pa3

pg1,  sad n
      jmp pg2
      lac i pa3
      cal car
      spa
      jmp pg3
      lac ar2
      jda pw1
      lac 100
      cal evo
      jsp uw1
      dio ar2
      cla
      sas pa4
      jmp pg4

pg3,  idx pa3
      lac i pa3
      dac pa3
      jmp pg1

/terminate program

pg4,  lac pa4
      jda uw
      dio pa4
      jsp uw1
      dio pa3
      lac uw
      jmp x

pg2,  lac pa4
      jda uw
      dio pa4
      jsp uw1
      dio pa3
      lac uw
      jmp x

ret,  dac pa4
      jmp x

/go

go,  lio 100
      lac n
      cal cns
      dac pa3
      jmp prx

/SETQ

stq,  dac ar1
      dio t1
      lac i ar1
      cal asc
      jmp qa4
      jda pw1
      lac ar1
      cal cdr
      cal car
      lio t1
      cal evl
      jda uw
      dio t0
      idx t0
      lac uw
      dac i t0
      jmp x

/CAR

cdr,  idx 100

/ATOM

atm,  lac i 100
      sma
      jmp fa1

tru,  lac tr
      jmp x

/NULL

nul,  lio n

/eq

eqq,  dio a1
      sad a1
      jmp tru
      lac i a1
      and i 100
      and (jmp
      sas (jmp
      jmp fa1
lac 100
cal vad
sad a0
jmp tru
jmp fal

/RPLACD
rdc, idx 100
sub (1

/RPLACA
rda, dio i 100
jmp x

/create atom
mka, ior (add
dac 100
lio n

/CONS
cns, idx ffl

cnc, lac fre
sad n
jmp gcs

cna, dac to
lac 100
dac i fre
idx fre
lac i fre
dio i fre
dac fre
lac to
jmp x

/PLUS
pls, cal elc
lio (add a0
plz, dzm a0
pl1, dio plo
pl2, sad n
jmp ple
dac a1
lac 1 a1
cal vag
plo, 0
dac a0
lac a1
cal cdr
jmp pl2
ple, lac a0
jmp crn

/logand, logor, times
lga, cal elc
lio (-0
dio a0
lio (and a0
jmp pl1

lgo, cal elc
lio (ior a0
jmp pl2

tim, cal elc
lio (1
dio a0
lio (jmp tic
jmp pl1

tic, mul a0
scr is
dio 100
add 100
jmp pl0+1

gcs, jsp gc
lac fre
sas n
jmp cna
jmp qgl

/TERPRI
tpr, law 77
ical pc
jmp prx

/prin1
pr1, lac 1 100
sma
jmp qpl1
sub (lac
spa
jmp prn
and (-jmp

pra, sad n
jmp x

dac a0
lac 1 a0
ral 6s
cal pc
lac 1 a0
xor 6s

cal pc
lac 1 a0

cal pc
idx a0
lac 1 a0
jmp pra
prn, lac 100
cal vag
dac t1
clf 2
setup to,6

prv, lio t1
sad (-1
stf 2
cla
rcl 3s
dio t1
sza i
law 20
sad (20
szt 2
cal pc
isp t0
jmp prv
jmp prx

/gensym

rmp, lac 1100
and (jmp
sad (jmp
jmp tru
jmp fal

/do a CONS into full word space

cf, lio n
cpf, dzm flf1
jmp cnc

/minus

min, cal vag
 cma
 jmp crn

/xeq

exeq, cal vad
 lac tr
dac t1
 lac a2
cal vag
 lio a0
dio xei
 lac a1
 lio uw
 xei, 0
 jmp xen
dio a2
 xer, cal crn
dac ar1
 lac a2
cal crn
 dac ar2

lac t1
 cal cns-1
 lio ar2
 cal efc
 lio ar1
 dac 100
 jmp efc
 lio n
dio t1
 jmp xer

law

gsm, law gst
dac t0
gsi, idx 1 to
sad (12
jmp gan
sad (21
law 1
dac 1 to

gsp, lac gst+2
ral 6s
ior gst+1
ral 6s
ior gst
cal cf
law 6700
ior gst+4
ral 6s
ior gst+3
 lio t0
cal cpf
cal mka
 jmp x

gsn, law 20
dac 1 to
idx t0
sas (gst+5
 jmp gsi
 jmp gsp

/qotient

qot, cal vad
 lio a0
 cla
 spi
 clc
 rcl 1a
 div a1
 jmp q14
 jmp crn

/cond

cnd, dio ar2
cd1, dac ar1
sad n
jmp qa3
jda pw1
lac ar2
jda pw1
lac 1 ar1
cal car
cal evo
jda uw
dio ar2
jsp uw1
dio ar1
lac uw
sas n
jmp cdy
idx ar1
lac 1 ar1
jmp cd1
cdy,
lac 1 ar1
cal cdr
cal car
jmp evo

/STOP

stp, cal vag
hl +cli-opr
jmp prx

/GREATERP

grp, cal vad
clo
sub a0
szo
lac a1
sma
jmp fal
jmp tru

/get a character

ava, szs 50
jmp avi
c1i

avx, lac 77
sza 1
jmp avr
rcl 9s
dio 1 avx
ral 2s
spq
jmp ava
ral 7s
ior (rar
dac avc
law 525

avc, xx
sma
jmp qc3

avt, law 77
and avc
sas (72
sad (74
dac csi
sad csi
jmp ava
jmp x

avr, index avx, ave, avx
init avx, buf
dap avs

avn, rpa
rcr 9s
rpa
rcl 9s

avs, dio xy
step avs, dio 100, avn
jmp ava

avi, szf 1 1
jmp ava
tyi
cfr 1
dio avc
jmp avt

/terminate print name

mkn, law 72
sas cs0
cal oc
idx isi
dac t0
lio n
dio isi
lac t0
dio t0
jmp x

/pack character into print name

pak, dap pk1
lac csi
sad cso
jmp pk1
dac cs0
cal oc

pk1, law
dac 100
jmp oc

start
Lisp interpreter 3-20-64, part 2

/PRINT

pnt, dac a0
dac a1
cal tpr

pn1, lio 1 a0
spi
jmp pn2
law 57

pn5, cal pc
lac a0
cal cdr
jda pwl
lio 1 a0
dio a0
jmp pn1

pn2, lac a0
cal pr1

pn6, jsp uw1
cla
dio a0
spi
jmp pn7
lio 1 a0
spi 1
jmp pn5
lac a0
sad n
jmp pn3
law 73
cal pc
lac a0
cal pr1

pn3, law 55
cal pc
jmp pn6

pn7, cal pc
lac a1
jmp a0

/READ

r18, 0
r19, 0
rin, lac rx
dac ar1
dzm r19

ris, jsp rhe
sza 1
jmp rie
sad (57)
jmp ria
sad (55)
jmp r1b

r1d, spi
jmp r12

r1q, idx ar1
lac 1 ar1
dio 1 ar1
dac r19
jsp rhe
jmp rix

r13, dac r19
jmp r13-2

r12, lac (jmp r13
jda pwl
law r1c

/read symbol and terminator

rhe, dap rhx
clf 5
dzm t1
law 1s1-1
dac 1s1
dzm 1s1-1
law 72
dac cso

rhn, cal ava
dac 100
lio cs1
rir 3s
spi
jmp rhb
sad (33)
cla
sas (57)
sad (55)
jmp rye
sad (73)
jmp rye

rhb, sad (56)
jmp ryo
sad (77)
jmp rhn
sad (36)
cla
sza 1
jmp rye
sad (75)
jmp rye+1
law 1 7
and 100
sza 1
jmp ryn
lac 100
sad (20)
jmp ryn
ryn, stf 5
cal pak
jmp rhn

ryj, lac t1
cal crn
jmp rhr

ryo, cal ava
jmp ryp

/symbol lookup

rye, dac r18
cal mkn
dac a0
sad n
jmp ryy
szf 1 5
jmp ryj
lac i lob

rys, dac t0
sad n
jmp ryc
lac i t0
dac t1
lac i t1
dac t1
lac a0

ryw, dac a1
sas n
jmp ryt
sad t1
jmp rhh

ryd, idx t0
lac i t0
jmp rys

ryt, lac t1
sad n
jmp ryd
lac i a1
sas i t1
jmp ryd
idx t1
lac i t1
dac t1
idx a1
lac i a1
jmp rwy

ryc, lac a0
cal mka
lio i lob
cal cn5
dac i lob

rhh, lac i t0
jmp rhr

ryn, lio 100
lac ti
rri 3s
rcl 3s
dac t1
lac 100
jmp ryp+1

ryy, cli
lio (1s1-1
dio tsi

rhr, dac t0
lac r19
lio r18
dio r19
lio t0

rhx, exit

/, space tab

ric, lac ar1
spi
jmp ris
spa
jmp r14

rio, dio t0
cal cdr
lio t0

rie, swap
cal cn5
idx ar1
lac to
dac i ar1
dac ar1
jmp ris

r14, lac t0
jmp ar1

/

ria, dio t0
lac ar1
jda pwl
lac to
spa
jmp riz

riy, cal cn5-1
dac ar1
lio ar1
cal rdc
jmp ris

riz, dzm ar1
jmp ris
/x is atomic: search a-list, then p-list

e1,  ral 1s
    spa
    jmp  e1
    lac ar1
    cal  ar2
    jmp  t0
    lac i t0
    spa
    jmp  e2

ev5,  lac ar1

ev4,  cal  cdr
    sad n
    jmp  qa8
    dac t0
    lac i t0
    sad  iap
    jmp  ev5
    idx t0
    lac i t0
    jmp  ev4

ev6,  idx t0
    lac i t0
    cal  car
    jmp  ex

ev2,  lac ar1
    szs 10
    cal  pnt
    lac i ar1
    spa
    jmp  e1
    dac t0
    lac i t0
    spa
    jmp  e2

/exit from EVAL

ex,  szs 10
    jmp  pnt
    jmp  x

e2,  lac t0

ev8,  cal  cdr
    sad n
    jmp  ev3
    lac i uw
    sad  ifs
    jmp  efs
    sad  isb
    jmp  esb
    sad  lxp

/ril,  idx ar1
    lac  i ar1
    lio  n
    dio i ar1

evc,  idx ar1
    lio i ar1
    lac uw
    dzm ar1
    cal  cns
    jmp  evo

/evaluate function name and try again

ev3,  lac i ar1
    cal  ar2
    jmp  qa8
    cal  cdr

/evaluate current expression

evo,  dac ar1

/car[x] not atomic

en1,  lac ar1

/car[x] is atomic: search its p-list

ev,  szs 10
    jmp  pnt
    jmp  x

e2,  lac t0

ev8,  cal  cdr
    sad n
    jmp  ev3
    lac i uw
    sad  ifs
    jmp  efs
    sad  isb
    jmp  esb
    sad  lxp

/eval

eval,  dio ar2

eval,  dlo ar1

eval,  rlo 1s
    lac  uw
    sp1
    jmp  ar1
    lio  uw
    lac ar1
    sza
    jmp  rlo
    lac  uw
    jmp  rlo
\texttt{jmp exp}
\texttt{sad ifx}
\texttt{jmp efx}
\texttt{idx t1}
\texttt{lac i t1}
\texttt{jmp ev8}

\texttt{/function is PSUBR}
\texttt{efs, idx uw}
\texttt{lac i uw}
\texttt{cal car}
\texttt{cal vag}
\texttt{dac exx}
\texttt{idx ar1}
\texttt{lac i ar1}
\texttt{llo ar2}
\texttt{exy, dac 100}
\texttt{daz ar1}
\texttt{exx, 0}
\texttt{jmp ex}

\texttt{/function is FEXPR}
\texttt{efx, idx uw}
\texttt{lac i uw}
\texttt{cal car}
\texttt{jda pw1}
\texttt{lac ar1}
\texttt{cal cdr}
\texttt{cal efx}
\texttt{jda pw1}
\texttt{lac ar2}
\texttt{cal efx}
\texttt{cal cns-1}
\texttt{jsp uw1}
\texttt{cal efx}
\texttt{jsp uw1}
\texttt{cal efx}
\texttt{dac ar1}
\texttt{jmp ev2}
\texttt{efq, cal cns-1}
\texttt{llo to}
\texttt{lac 1qu}
\texttt{dac 100}
\texttt{jmp cns}
\texttt{efc, dio 100}
\texttt{llo to}
\texttt{jmp cns}

\texttt{/function is EXPR}
\texttt{exp, idx uw}
\texttt{lac i uw}
\texttt{dac a1}
\texttt{idx ar1}
\texttt{llo i ar1}
\texttt{daz ar1}
\texttt{lac i a1}
\texttt{cal cns}
\texttt{jmp evo}

\texttt{/function is SUBR}
\texttt{esb, idx uw}
\texttt{lac i uw}
\texttt{cal car}
\texttt{jda pw1}
\texttt{lac ar1}
\texttt{cal cdr}
\texttt{llo ar2}
\texttt{cal elc}
\texttt{jmp els}

\texttt{/evaluate argument list: also LIST}
\texttt{elc, sad n}
\texttt{jmp x}
\texttt{dac ar1}
\texttt{dio ar2}
\texttt{lac ar2}
\texttt{jda pw1}
\texttt{lac ar1}
\texttt{dzm ar1}
\texttt{ele, llo 1 pdl}
\texttt{dac to}
\texttt{jda pw1}
\texttt{lac ar1}
\texttt{jda pw1}
\texttt{lac 1 to}
\texttt{cal ev1}
\texttt{cal cns-1}
\texttt{jsp uw1}
\texttt{dio ar1}
\texttt{llo to}
\texttt{lac ar1}
\texttt{sza 1}
\texttt{dio ar1}
\texttt{idx ar1}
\texttt{sub (1}
\texttt{sas to}
\texttt{llo i ar1}
\texttt{lac to}
\texttt{dac 1 ar1}
\texttt{dac ar1}
\texttt{idx to}
\texttt{dio 1 to}
\texttt{jsp uw1}
\texttt{swap}
\texttt{cal cdr}
\texttt{sas n}
\texttt{jmp ele}
\texttt{jsp uw1}
\texttt{dio ar2}
\texttt{idx ar1}
\texttt{lac i ar1}
\texttt{llo n}
\texttt{dio 1 ar1}
\texttt{dac ar1}
\texttt{szs 10}
\texttt{cal pnt}
\texttt{lac ar1}
\texttt{jmp x}
els, dac ar1
jsp uwI
swap
cal vag
dac exx
init esa,a0-1

/store arguments for subroutine

eda, sad n
jmp exs
idx esa
sad (dac a2+1
jmp qa7
lac i ar1

esa, dac xy
idx ar1
lac i ar1
dac ar1
jmp eda

exs, lac a0
lio a1
jmp exy

/caar[x] = LAMBDA

e3, lac ar1
jda pwl
lac ar2
jda pwl
lac i ar1
cal cdr
cal car
jda pwl
lac ar1

cal cdr
lio ar2
cal elc
dac ar1
jsp uw1
dio a0
jsp uw1
dio ar2

ep1, lac a0
sad n
jmp ep2
lac ar1
sad n
jmp qf3
lac i a0
lio i ar1
cal cns
lio ar2
cal cns
dac ar2
idx a0
lac i a0
dac a0
idx ar1
dac ar1
dac ar1
jmp ep1

ep2, sas ar1
jmp qf2
jsp uw1
dio ar1
lac i ar1
cal cdr
cal cdr
cal car
jmp evo
/error halt entries

qa3,  lac n
  sas pa3
  jmp x
error flex 1cd
  lac n
  jmp x

qa4,  error flex uss
  jmp prx

qa7,  error flex tma
  jmp exs

qa8,  error flex uas
  clf 6
  lac ar1
  cal pnt
  cal tpr
  jmp go

qc3,  error flex 1lp
  law 377
  and avc
  hlt+cli-opr+1
  jmp ava

qf2,  error flex lts
  jmp go

qf3,  error flex ats
  jmp go

qg2,  error flex pce
  jmp go

qg1,  error flex sce
  jmp go

q13,  lac 100
  dac a2
  error flex nna
  clf 6
  lac a2
  cal pnt
  cal tpr
  jmp qix

q14,  error flex ovf

qix,  cla 16
    jmp crn

qp1,  error flex ana

prx,fal,  lac n
    jmp x

start
lisp storage 3-23-64

constants

/special symbols

ssy,

1qu,  quo
1la,  lam
1ap,  apv
1ob,  obl
1sb,  sbr
1fs,  fsb
1xp,  xpr
1fx,  fwp
fre,  nil
bfw,  frs-4
tr,  t

pdl,  pdo-1

ari,  nil
ar2,  nil
pa3,  nil
pa4,  0

pdo,

/load storage parameters

lio mz
clc+hlt-opr
lat+cli-opr
and ad
dac h11
hlt
lat
and ad
dac lp1
law l end
add h11
spa
jmp pdo
law l frs-pdo
add lp1
spa
jmp pdo
law l pdo+end-frs
add h11
sub lp1
spa
jmp pdo

/set up registers

stu,  law pdo
add lp1
dac frs
lio h11

rcr 1s
r1l 1s
d1o h1
law end
dac t0

/relocate storage

rrs,  law l l
add t0
dac t0
law l 4
add l t0
sma
jsp rrl
jsp mvs
law l i
add t0
dac t0
sub frl
spa
jsp rrl
jsp mvs
lac t0
sas ofs
jmp rrs
law ssy
dac t0

/relocate special registers

rss,  jsp rrl
idx t0
sas esy
jmp rss
lac l 1ob
jda gfr
law go
dap gcx
jmp g2e

/relocate 1 word, move 1 word

rrl,  dap rrx
lac l t0
and ad
sub dfs
spa
jmp rrx
lac l t0
add fro
sub dfs
dac l t0
rrx,  jmp .

mvs,  dap mvx
lac t0
add fro
sub dfs
dac t1
lac l t0
dac l t1
mvx, jmp .

/ constants etc. 

ad, 177777
lpl, 0
hll, 0
mz, -0
ofs, frs
frl, fws
esy, pdo
ofr, pdo

define item X
    .+2 .+3
    add X nil
    termin

define next A
    A .+1
    termin

define subr F
    .+2 .+7
    add F+2 .+1
    sb r .+1
    F nil
    termin

define fsubr F
    .+2 .+7
    add F+2 .+1
    f sb .+1
    F nil
    termin

define apval A
    apv .+1
    A nil
    termin

frs,
nil, add f38 kz

kt, apval nil
kt, apval t

obj, add fb0 .+1
apv .+1

obl, ols nil

/object list

ols, subr f2
subr f3
subr f4

fsubr f6
subr f7
subr f8
subr f12
subr f13
fsubr f14
subr f18
subr f21
subr f24
subr f26
subr f27
subr f32
subr f33
subr f34
fsubr f50
subr f51
subr f52
fsubr f53
subr f54
fsubr f60
fsubr f61
fsubr f62
fsubr f63
subr f00
subr f01
subr fa3

next t
next obj
subr fb2
subr fb3
subr fb4

f ws,

define opr A
    loca A
    0
    termin

define nam1 X
    nnil
    termin

define nam2 X,Y
    .+1
    termin

define nam3 X,Y,Z
    .+1
    termin
/SUBRs and FSUBRs

f2, loca atm nam2 flex ato,767644
f3, loca car nam1 flex car
f4, loca cdr nam1 flex cdr
f6, loca cnd nam2 flex con,767664
f7, loca cns nam2 flex con,767622
f8, loca eqq nam1 766550
f12, loca gsm nam2 flex gen,flex sym
f13, loca grp nam3 flex gre,flex ate,765147
f14, loca elc nam2 flex lls,767623
f18, loca min nam2 flex min,762422
f21, loca mnp nam3 flex num,flex ber,767647
f24, loca stp nam2 flex sto,767647
f26, loca pri nam2 flex pri,764501
f27, loca qot nam3 flex quo,flex tie,764523
f32, loca rda nam2 flex rpl,flex aca
f33, loca rdc nam2 flex rpl,flex acd
f00, loca xeq nam1 flex xeq
f01, loca crn nam1 flex loc
f34, loca tpr nam2 flex ter,flex pri
f50, loca pgm nam2 flex pro,767667
f51, loca ret nam2 flex ret,flex urn
f52, loca goe nam1 766746
f53, loca stq nam2 flex set,767650
f54, loca aso nam2 flex sas,flex soc
fb2, loca rin nam2 flex rea,767664
fb3, loca evl nam2 flex eva,767643
fb4, loca pnt nam2 flex pri,764523
fb5, loca car nam2 flex quo,762365
fa3, loca nul nam2 flex nul,767643
f60, loca pls nam2 flex plu,767622
f61, loca tim nam2 flex tim,766522
f62, loca lga nam2 flex log,flex and
f63, loca lgo nam2 flex log,764651

/miscellany
f38, nam1 flex nil
f40, nam2 flex lam,flex bda
f42, nam2 flex apv,766143
f43, nam2 flex sub,767651
f44, nam2 flex exp,767651
f45, nam2 flex fsu,766251
f46, nam2 flex fex,764751
f40, nam2 flex obl,flex 1st
f37, nam1 767623

end,

start pdo
2. Alphabetic Listing of Defined Macro Symbols

Following is an alphabetic listing of the defined symbols used in the macro symbolic program for Basic PDP-1 LISP. The listing shows either the numeric meaning of the instruction or the numeric register (octal) in which the subroutine commences. For the mnemonic derivation or significance of the symbols, see Section 4 below.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>lap</td>
<td>2333</td>
<td>clo</td>
</tr>
<tr>
<td>lfs</td>
<td>2336</td>
<td>cna</td>
</tr>
<tr>
<td>lfx</td>
<td>2340</td>
<td>cnc</td>
</tr>
<tr>
<td>lla</td>
<td>2332</td>
<td>cnd</td>
</tr>
<tr>
<td>lob</td>
<td>2334</td>
<td>cns</td>
</tr>
<tr>
<td>lqu</td>
<td>2331</td>
<td>cpf</td>
</tr>
<tr>
<td>lsb</td>
<td>2335</td>
<td>crn</td>
</tr>
<tr>
<td>1xp</td>
<td>2337</td>
<td>csi</td>
</tr>
<tr>
<td>a1</td>
<td>42</td>
<td>csb</td>
</tr>
<tr>
<td>a2</td>
<td>43</td>
<td>dba</td>
</tr>
<tr>
<td>ad</td>
<td>2470</td>
<td>ddc</td>
</tr>
<tr>
<td>apv</td>
<td>3110</td>
<td>dia</td>
</tr>
<tr>
<td>ar1</td>
<td>2345</td>
<td>div</td>
</tr>
<tr>
<td>ar2</td>
<td>2346</td>
<td>dra</td>
</tr>
<tr>
<td>a0</td>
<td>41</td>
<td>e1</td>
</tr>
<tr>
<td>as1</td>
<td>405</td>
<td>e2</td>
</tr>
<tr>
<td>as2</td>
<td>420</td>
<td>e3</td>
</tr>
<tr>
<td>asc</td>
<td>403</td>
<td>eda</td>
</tr>
<tr>
<td>asd</td>
<td>377</td>
<td>efc</td>
</tr>
<tr>
<td>aso</td>
<td>374</td>
<td>efq</td>
</tr>
<tr>
<td>asr</td>
<td>402</td>
<td>efs</td>
</tr>
<tr>
<td>atm</td>
<td>562</td>
<td>efx</td>
</tr>
<tr>
<td>ava</td>
<td>1133</td>
<td>e1c</td>
</tr>
<tr>
<td>avc</td>
<td>1152</td>
<td>e1e</td>
</tr>
<tr>
<td>ave</td>
<td>110</td>
<td>eis</td>
</tr>
<tr>
<td>avi</td>
<td>1204</td>
<td>en1</td>
</tr>
<tr>
<td>avn</td>
<td>1173</td>
<td>end</td>
</tr>
<tr>
<td>avr</td>
<td>1165</td>
<td>ep1</td>
</tr>
<tr>
<td>avs</td>
<td>1177</td>
<td>ep2</td>
</tr>
<tr>
<td>avt</td>
<td>1155</td>
<td>eqq</td>
</tr>
<tr>
<td>avx</td>
<td>1136</td>
<td>erm</td>
</tr>
<tr>
<td>beg</td>
<td>11</td>
<td>ern</td>
</tr>
<tr>
<td>bfw</td>
<td>2342</td>
<td>err</td>
</tr>
<tr>
<td>buf</td>
<td>63</td>
<td>exr</td>
</tr>
<tr>
<td>car</td>
<td>555</td>
<td>esa</td>
</tr>
<tr>
<td>cd1</td>
<td>1071</td>
<td>esb</td>
</tr>
<tr>
<td>cdr</td>
<td>554</td>
<td>esy</td>
</tr>
<tr>
<td>cdy</td>
<td>1114</td>
<td>ev2</td>
</tr>
<tr>
<td>cf</td>
<td>761</td>
<td>ev3</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ev4</td>
<td>1637</td>
<td>fb3</td>
</tr>
<tr>
<td>ev5</td>
<td>1636</td>
<td>fb4</td>
</tr>
<tr>
<td>ev6</td>
<td>1651</td>
<td>fb5</td>
</tr>
<tr>
<td>ev8</td>
<td>1662</td>
<td>fb0</td>
</tr>
<tr>
<td>evc</td>
<td>1620</td>
<td>ff1</td>
</tr>
<tr>
<td>ev1</td>
<td>1563</td>
<td>fre</td>
</tr>
<tr>
<td>evo</td>
<td>1564</td>
<td>frl</td>
</tr>
<tr>
<td>ex</td>
<td>1656</td>
<td>fro</td>
</tr>
<tr>
<td>exp</td>
<td>1746</td>
<td>frs</td>
</tr>
<tr>
<td>exs</td>
<td>2074</td>
<td>f01</td>
</tr>
<tr>
<td>exx</td>
<td>1713</td>
<td>f00</td>
</tr>
<tr>
<td>exy</td>
<td>1711</td>
<td>f0b</td>
</tr>
<tr>
<td>f12</td>
<td>3226</td>
<td>fws</td>
</tr>
<tr>
<td>f13</td>
<td>3234</td>
<td>fxp</td>
</tr>
<tr>
<td>f14</td>
<td>3244</td>
<td>g1</td>
</tr>
<tr>
<td>f18</td>
<td>3252</td>
<td>g2a</td>
</tr>
<tr>
<td>f2</td>
<td>3170</td>
<td>g2e</td>
</tr>
<tr>
<td>f21</td>
<td>3260</td>
<td>g2f</td>
</tr>
<tr>
<td>f24</td>
<td>3270</td>
<td>g2n</td>
</tr>
<tr>
<td>f26</td>
<td>3276</td>
<td>g2x</td>
</tr>
<tr>
<td>f27</td>
<td>3304</td>
<td>ga1</td>
</tr>
<tr>
<td>f3</td>
<td>3176</td>
<td>ge</td>
</tr>
<tr>
<td>f32</td>
<td>3314</td>
<td>gci</td>
</tr>
<tr>
<td>f33</td>
<td>3322</td>
<td>gco</td>
</tr>
<tr>
<td>f34</td>
<td>3340</td>
<td>gep</td>
</tr>
<tr>
<td>f37</td>
<td>3526</td>
<td>gcs</td>
</tr>
<tr>
<td>f38</td>
<td>3470</td>
<td>gex</td>
</tr>
<tr>
<td>f4</td>
<td>3202</td>
<td>gfa</td>
</tr>
<tr>
<td>f42</td>
<td>3476</td>
<td>grd</td>
</tr>
<tr>
<td>f43</td>
<td>3502</td>
<td>gfl</td>
</tr>
<tr>
<td>f44</td>
<td>3506</td>
<td>gfn</td>
</tr>
<tr>
<td>f45</td>
<td>3512</td>
<td>grf</td>
</tr>
<tr>
<td>f46</td>
<td>3516</td>
<td>grfu</td>
</tr>
<tr>
<td>f40</td>
<td>3472</td>
<td>gfx</td>
</tr>
<tr>
<td>f51</td>
<td>3354</td>
<td>go</td>
</tr>
<tr>
<td>f52</td>
<td>3362</td>
<td>goe</td>
</tr>
<tr>
<td>f53</td>
<td>3366</td>
<td>grp</td>
</tr>
<tr>
<td>f54</td>
<td>3374</td>
<td>g0</td>
</tr>
<tr>
<td>f50</td>
<td>3346</td>
<td>gsi</td>
</tr>
<tr>
<td>f6</td>
<td>3206</td>
<td>gsm</td>
</tr>
<tr>
<td>f61</td>
<td>3446</td>
<td>gsn</td>
</tr>
<tr>
<td>f62</td>
<td>3454</td>
<td>gsp</td>
</tr>
<tr>
<td>f63</td>
<td>3462</td>
<td>gat</td>
</tr>
<tr>
<td>f60</td>
<td>3440</td>
<td>hi</td>
</tr>
<tr>
<td>f7</td>
<td>3214</td>
<td>hi1</td>
</tr>
<tr>
<td>f8</td>
<td>3222</td>
<td>ioh</td>
</tr>
<tr>
<td>fa3</td>
<td>3432</td>
<td>isi</td>
</tr>
<tr>
<td>fa1</td>
<td>2241</td>
<td>ktt</td>
</tr>
<tr>
<td>fb2</td>
<td>3402</td>
<td>kz</td>
</tr>
</tbody>
</table>

39
<p>| lai  | 760040 | plz  | 633  |
| lam  | 3104   | pn1  | 1240 |
| lga  | 651    | pn2  | 1253 |
| lgo  | 656    | pn3  | 1274 |
| lia  | 760020 | pn5  | 1244 |
| lp1  | 2471   | pn6  | 1255 |
| min  | 764    | pn7  | 1277 |
| mka  | 611    | pnt  | 1235 |
| mkn  | 1242   | prt  | 703  |
| mul  | 540000 | pra  | 712  |
| mvs  | 2460   | prn  | 730  |
| mvx  | 2467   | prv  | 736  |
| mz   | 2473   | prx  | 2244 |
| n    | 234    | pwl  | 44   |
| nil  | 2500   | pwx  | 53   |
| nmp  | 754    | qa3  | 2154 |
| nul  | 567    | qa4  | 2163 |
| obj  | 2514   | qa7  | 2166 |
| ob1  | 2520   | qa8  | 2171 |
| oc   | 165    | qc3  | 2200 |
| oc1  | 206    | qf2  | 2206 |
| ofr  | 2477   | qf3  | 2241 |
| ofs  | 2474   | qg1  | 2247 |
| ols  | 2522   | qg2  | 2244 |
| out  | 210    | qi3  | 2222 |
| pa3  | 2347   | qj4  | 2233 |
| pa4  | 2350   | qlx  | 2235 |
| pak  | 1224   | qot  | 1057 |
| pc   | 117    | qpl  | 2237 |
| pcc  | 135    | quo  | 3074 |
| pch  | 143    | rda  | 607  |
| pd1  | 2344   | rdc  | 605  |
| pdo  | 2351   | ret  | 524  |
| pg1  | 472    | rbb  | 1360 |
| pg2  | 516    | rhe  | 1333 |
| pg3  | 511    | rhe  | 1462 |
| pg4  | 515    | rhn  | 1343 |
| pg5  | 437    | rhr  | 1476 |
| pg6  | 452    | rhx  | 1503 |
| pg7  | 453    | ri2  | 1330 |
| pg9  | 466    | ri3  | 1326 |
| pgm  | 423    | ri4  | 1524 |
| pg0  | 471    | ri8  | 1302 |
| plk1 | 1232   | ri9  | 1303 |
| pl1  | 634    | ria  | 1526 |
| pl2  | 635    | rib  | 1543 |
| ple  | 647    | ric  | 1504 |
| plo  | 642    | rid  | 1346 |
| plx  | 631    | rie  | 1514 |</p>
<table>
<thead>
<tr>
<th>Word</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>rin</td>
<td>1304</td>
<td>spq</td>
<td>650500</td>
</tr>
<tr>
<td>rio</td>
<td>1511</td>
<td>ssy</td>
<td>2331</td>
</tr>
<tr>
<td>riq</td>
<td>1320</td>
<td>stop</td>
<td>1120</td>
</tr>
<tr>
<td>ris</td>
<td>1307</td>
<td>stq</td>
<td>533</td>
</tr>
<tr>
<td>rix</td>
<td>1547</td>
<td>stu</td>
<td>2377</td>
</tr>
<tr>
<td>riy</td>
<td>1534</td>
<td>swp</td>
<td>(600000)</td>
</tr>
<tr>
<td>riz</td>
<td>1541</td>
<td>szm</td>
<td>640500</td>
</tr>
<tr>
<td>rrl</td>
<td>2445</td>
<td>t</td>
<td>2502</td>
</tr>
<tr>
<td>rss</td>
<td>2410</td>
<td>t1</td>
<td>22</td>
</tr>
<tr>
<td>rrx</td>
<td>2457</td>
<td>tic</td>
<td>666</td>
</tr>
<tr>
<td>rss</td>
<td>2434</td>
<td>tim</td>
<td>661</td>
</tr>
<tr>
<td>rx</td>
<td>561</td>
<td>tpr</td>
<td>700</td>
</tr>
<tr>
<td>ryc</td>
<td>1455</td>
<td>tr</td>
<td>2343</td>
</tr>
<tr>
<td>ryd</td>
<td>1436</td>
<td>tru</td>
<td>565</td>
</tr>
<tr>
<td>rye</td>
<td>1411</td>
<td>t0</td>
<td>21</td>
</tr>
<tr>
<td>ryj</td>
<td>1404</td>
<td>uw</td>
<td>54</td>
</tr>
<tr>
<td>ryn</td>
<td>1464</td>
<td>uwl</td>
<td>55</td>
</tr>
<tr>
<td>roo</td>
<td>1407</td>
<td>uwx</td>
<td>62</td>
</tr>
<tr>
<td>ryp</td>
<td>1401</td>
<td>vad</td>
<td>156</td>
</tr>
<tr>
<td>rys</td>
<td>1421</td>
<td>vag</td>
<td>144</td>
</tr>
<tr>
<td>ryt</td>
<td>1441</td>
<td>x</td>
<td>556</td>
</tr>
<tr>
<td>ryw</td>
<td>1431</td>
<td>xei</td>
<td>1000</td>
</tr>
<tr>
<td>ryy</td>
<td>1473</td>
<td>xen</td>
<td>1017</td>
</tr>
<tr>
<td>sbr</td>
<td>3114</td>
<td>xeq</td>
<td>767</td>
</tr>
<tr>
<td>sft</td>
<td>660000</td>
<td>xer</td>
<td>1003</td>
</tr>
<tr>
<td>smi</td>
<td>652000</td>
<td>xpr</td>
<td>3120</td>
</tr>
<tr>
<td>sni</td>
<td>644000</td>
<td>xy</td>
<td>0</td>
</tr>
</tbody>
</table>
3. Numeric Listing of the Defined Macro Symbols

Following is a listing in numerical order by register number or other meaning of the defined symbols in the macro symbolic program for Basic PDP-1 LISP.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xy</td>
<td>0</td>
<td>gfr</td>
</tr>
<tr>
<td>go</td>
<td>4</td>
<td>gfn</td>
</tr>
<tr>
<td>beg</td>
<td>11</td>
<td>gfd</td>
</tr>
<tr>
<td>t0</td>
<td>21</td>
<td>gfu</td>
</tr>
<tr>
<td>t1</td>
<td>22</td>
<td>gfx</td>
</tr>
<tr>
<td>g0</td>
<td>23</td>
<td>gra</td>
</tr>
<tr>
<td>gl</td>
<td>24</td>
<td>glf</td>
</tr>
<tr>
<td>hi</td>
<td>25</td>
<td>g2e</td>
</tr>
<tr>
<td>csi</td>
<td>26</td>
<td>g2n</td>
</tr>
<tr>
<td>cso</td>
<td>27</td>
<td>g2a</td>
</tr>
<tr>
<td>ffl</td>
<td>30</td>
<td>g2x</td>
</tr>
<tr>
<td>gax</td>
<td>31</td>
<td>gcx</td>
</tr>
<tr>
<td>isi</td>
<td>33</td>
<td>g2f</td>
</tr>
<tr>
<td>gst</td>
<td>34</td>
<td>gc1</td>
</tr>
<tr>
<td>a0</td>
<td>41</td>
<td>aso</td>
</tr>
<tr>
<td>a1</td>
<td>42</td>
<td>ase</td>
</tr>
<tr>
<td>a2</td>
<td>43</td>
<td>asr</td>
</tr>
<tr>
<td>pwk</td>
<td>44</td>
<td>asc</td>
</tr>
<tr>
<td>pwx</td>
<td>53</td>
<td>as1</td>
</tr>
<tr>
<td>uw</td>
<td>54</td>
<td>as2</td>
</tr>
<tr>
<td>uwk</td>
<td>55</td>
<td>pgm</td>
</tr>
<tr>
<td>uwx</td>
<td>62</td>
<td>pg5</td>
</tr>
<tr>
<td>buf</td>
<td>63</td>
<td>pg6</td>
</tr>
<tr>
<td>ave</td>
<td>110</td>
<td>pg7</td>
</tr>
<tr>
<td>crn</td>
<td>112</td>
<td>pg9</td>
</tr>
<tr>
<td>pc</td>
<td>117</td>
<td>pg0</td>
</tr>
<tr>
<td>pcc</td>
<td>135</td>
<td>pg1</td>
</tr>
<tr>
<td>pch</td>
<td>143</td>
<td>pg3</td>
</tr>
<tr>
<td>vag</td>
<td>144</td>
<td>pg4</td>
</tr>
<tr>
<td>vad</td>
<td>156</td>
<td>pg2</td>
</tr>
<tr>
<td>oc</td>
<td>165</td>
<td>ret</td>
</tr>
<tr>
<td>oc1</td>
<td>206</td>
<td>goe</td>
</tr>
<tr>
<td>out</td>
<td>210</td>
<td>stq</td>
</tr>
<tr>
<td>err</td>
<td>216</td>
<td>cdr</td>
</tr>
<tr>
<td>erx</td>
<td>226</td>
<td>car</td>
</tr>
<tr>
<td>erm</td>
<td>227</td>
<td>x</td>
</tr>
<tr>
<td>ern</td>
<td>231</td>
<td>rx</td>
</tr>
<tr>
<td>fro</td>
<td>234</td>
<td>atm</td>
</tr>
<tr>
<td>n</td>
<td>234</td>
<td>tru</td>
</tr>
<tr>
<td>gc</td>
<td>235</td>
<td>nul</td>
</tr>
<tr>
<td>gco</td>
<td>251</td>
<td>eqq</td>
</tr>
<tr>
<td>gcp</td>
<td>260</td>
<td>rdc</td>
</tr>
<tr>
<td>rda</td>
<td>607</td>
<td>pld</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>mka</td>
<td>611</td>
<td>pnt</td>
</tr>
<tr>
<td>cns</td>
<td>614</td>
<td>pn1</td>
</tr>
<tr>
<td>cnc</td>
<td>615</td>
<td>pn5</td>
</tr>
<tr>
<td>cna</td>
<td>620</td>
<td>pn2</td>
</tr>
<tr>
<td>pls</td>
<td>631</td>
<td>pn6</td>
</tr>
<tr>
<td>plz</td>
<td>633</td>
<td>pn3</td>
</tr>
<tr>
<td>pl1</td>
<td>634</td>
<td>pn7</td>
</tr>
<tr>
<td>pl2</td>
<td>635</td>
<td>r18</td>
</tr>
<tr>
<td>plo</td>
<td>642</td>
<td>r19</td>
</tr>
<tr>
<td>ple</td>
<td>647</td>
<td>r1n</td>
</tr>
<tr>
<td>lga</td>
<td>651</td>
<td>ris</td>
</tr>
<tr>
<td>lgo</td>
<td>656</td>
<td>rid</td>
</tr>
<tr>
<td>tim</td>
<td>661</td>
<td>riq</td>
</tr>
<tr>
<td>tic</td>
<td>666</td>
<td>ri3</td>
</tr>
<tr>
<td>gcs</td>
<td>673</td>
<td>ri2</td>
</tr>
<tr>
<td>tpr</td>
<td>700</td>
<td>rhe</td>
</tr>
<tr>
<td>pr1</td>
<td>703</td>
<td>rhn</td>
</tr>
<tr>
<td>pra</td>
<td>712</td>
<td>rhb</td>
</tr>
<tr>
<td>prn</td>
<td>730</td>
<td>ryp</td>
</tr>
<tr>
<td>prv</td>
<td>736</td>
<td>ryj</td>
</tr>
<tr>
<td>npm</td>
<td>754</td>
<td>ryo</td>
</tr>
<tr>
<td>cf</td>
<td>761</td>
<td>rye</td>
</tr>
<tr>
<td>cpf</td>
<td>762</td>
<td>rys</td>
</tr>
<tr>
<td>min</td>
<td>764</td>
<td>ryw</td>
</tr>
<tr>
<td>xeq</td>
<td>767</td>
<td>ryd</td>
</tr>
<tr>
<td>xei</td>
<td>1000</td>
<td>ryt</td>
</tr>
<tr>
<td>xer</td>
<td>1003</td>
<td>ryc</td>
</tr>
<tr>
<td>xen</td>
<td>1017</td>
<td>rhh</td>
</tr>
<tr>
<td>gsm</td>
<td>1023</td>
<td>ryn</td>
</tr>
<tr>
<td>gsi</td>
<td>1025</td>
<td>ryy</td>
</tr>
<tr>
<td>gsp</td>
<td>1033</td>
<td>rhr</td>
</tr>
<tr>
<td>gsn</td>
<td>1051</td>
<td>rhx</td>
</tr>
<tr>
<td>qot</td>
<td>1057</td>
<td>ric</td>
</tr>
<tr>
<td>cnd</td>
<td>1070</td>
<td>rio</td>
</tr>
<tr>
<td>cd1</td>
<td>1071</td>
<td>rie</td>
</tr>
<tr>
<td>cdy</td>
<td>1114</td>
<td>r14</td>
</tr>
<tr>
<td>stp</td>
<td>1120</td>
<td>ria</td>
</tr>
<tr>
<td>grp</td>
<td>1123</td>
<td>riy</td>
</tr>
<tr>
<td>ava</td>
<td>1133</td>
<td>riz</td>
</tr>
<tr>
<td>avx</td>
<td>1136</td>
<td>rib</td>
</tr>
<tr>
<td>avc</td>
<td>1152</td>
<td>rix</td>
</tr>
<tr>
<td>avt</td>
<td>1155</td>
<td>evl</td>
</tr>
<tr>
<td>avr</td>
<td>1165</td>
<td>evo</td>
</tr>
<tr>
<td>avn</td>
<td>1173</td>
<td>ev2</td>
</tr>
<tr>
<td>avs</td>
<td>1177</td>
<td>ev3</td>
</tr>
<tr>
<td>avl</td>
<td>1204</td>
<td>evc</td>
</tr>
<tr>
<td>mkn</td>
<td>1212</td>
<td>e1</td>
</tr>
<tr>
<td>pak</td>
<td>1224</td>
<td>ev5</td>
</tr>
</tbody>
</table>