DLU 78/4

L I S P F 3

USERS GUIDE
by
Mats Nordstrom

June 1978

Mailing address:
Datalogilaboratoriet
Sturegatan 2B
S-752 23 UPPSALA
S W E D E N

This work has been supported by the Swedish Board for Technical Development (STU) no 76-4253.

TABLE OF CONTENTS

PREFACE

LISP F3 is a LISP system written in FORTRAN 4. The first version (LISP F1) was written 1970-1971 and has by now been delivered to about 100 different computer installations around the world.

LISP F1 was a LISP 1.5 system (with some extensions) but has now been almost completely rewritten into INTERLISP standard. LISP F3 is (almost) a subset of INTERLISP as defined in Te 74 or Ha 75. In addition it is about 3 - 10 times more efficient than LISP F1 and is easier to implement (as it is coded in a more structured style).

As a user's manual, Ha 75 is referred to (and delivered together with the system) and in this guide only differences from INTERLISP are reported.

Some of the functions in LISP F3 are coded in LISP and in the following it is assumed, that all those LISP-packages are included in your system. (Check with your installation manager!).

CHAPTER 1

PRIMARY DATATYPES.

small integers	range -n,n where n is implementation dependent. If x is the maximum positive integer in a full word, n is x - size of CAR,CDR.
large integers	range depending on the size of a full word.
alfanum atom.	Max. nr of characters = size of IO-buff/2 (default = 80).
strings	given as "THIS IS A STRING".
floating numbers	DO NOT EXIST.
lists	given as (A B (C D)) etc.

CHAPTER 2

INTERNAL REPRESENTATIONS.

A more complete description of the internal representations is given in the implementation guide. Here we only give the information needed for a complete knowlegde and usage of LISP F3 from the user's point of view.

a) The ADDRESS SPACE.

The address space is shown by the following figure:

```
1                                                                   MAX
I----------------------I-------------I----------------I---------------I
NIL   alphanum atoms         lists        pointers to    pointers to
      and strings                         big integers   small inte-
                                                         gers.
```

MAX = the largest positive integer in a halfword (or full word) in your computer.

b) alphanumerical ATOMS and STRINGS.

ATOMS are represented as a "two-pointer record"

```
          CAR          CDR
       I----------I----------I
       I          I           ----> property list
       I---I------I----------I
           V
      global value
```

The global value of an atom is stored in CAR(atom). (EVAL checks for a bound value BEFORE a global value - as in INTERLISP but in contradiction to LISP 1.5). A global value may be set either by SET/SETQ at the top level, or directly by RPLACA. If a global value has not been assigned, car(atom) points to the atom NOBIND.

STRINGS are represented exactly as atoms except for
- car(string) points to the atom STRING.
- Two different strings may have the same printname.
- Strings always have themselves as the value.

SUBSTRINGS are like strings, but instead of having a print name
- car(substring) = SUBSTR
- cdr(substring) = (sourcestring start . length)

PRINTNAMES are accessed through the pointer representing the atom and hidden from the user in a special area.

FUNCTION DEFINITIONS.
In INTERLISP each atom-record also has a "function field" called function cell (Ha 75 page 4). In LISP F3 user defined functions are stored as LAMBDA or NLAMBDA expressions under the property FNCELL. A SUBR or FSUBR is recognized by the atom pointer itself but in order to simulate the facility of making use of "free function indicators", GETD is defined to return (SUBR . FOO) if FOO is a FORTRAN coded SUBR, and (FSUBR . FOO) if it is a FSUBR.

The forms (SUBR . FOO) and (FSUBR . FOO) are simulated function indicators and legal function arguments to apply.

Ex.:

```
        (DE KAR(X) ((SUBR . CAR) X>
```

This definition of KAR causes KAR to behave exactly as CAR independently of whether CAR has been redefined to something else.

c) The SYMBOL TABLE can be looked upon.

The function
```
        (OBLIST x)
```
creates a new list of atoms, with the last atom created as the first member of the list, and the atomic argument x as the last one. As NIL is the very first atom created, (OBLIST) gives you all atoms and as T is the last one defined by a "clean" system, (OBLIST T) gives you all but SUBR's and FSUBR's.

Variable bindings are stored in an association list (as in LISP 1.5) but this list simulates a push down stack (as in INTERLISP) and is implicitly given to EVAL, APPLY and EVLIS.

The function
```
        (ALIST)
```
returns the actual association list.
If evaluation is to be performed in some special variable environment use

```
        (EVALA s ass)    - as (EVAL s) but uses
                           ass as the push down stack

        (APPLYA fn l ass) as (APPLY fn l)       -"-
```

Ex.: A "safe" definition of GETPQ may look like

```
        (DF GETPQ(A IND)(GETP (EVALA A (CDDR (ALIST))) IND>
```

I.e. the rebinding of A and IND, here done by GETPQ, is not seen inside the evaluation of A.

e) LISTS are represented as two pointer records with CAR and CDR fields.

f) NUMBERS are implemented as high valued pointers.

The value of a small integer is the value of the pointer subtracted by a proper offset. The value of a big integer is stored in a full word hidden from the user (but found through its pointer value).

CHAPTER 3

ATOMS OF PREDESIGNED MEANING.

Here is a list of those atoms which may be of interest for the LISP F3 user.

NIL,T	These atoms can not be destroyed by any functions such as RPLACA etc.
NOBIND	is stored in car of undefined atoms.
STRING	is stored in car of strings.
SUBSTR	is stored in car of substrings.
ADVISEDFNS	List of advised functions.
*BACKTRACEFLG	if true, eval-apply will store forms under execution. This is needed to perform the command BT (backtrace) inside a break.
*BACKTRACE	List of forms under execution if *BACKTRACEFLG = T.
BROKENFNS	List of broken functions.
CURFNS	List of those functions which have been defined before the first time (CURFILE file) was performed.
CURFILE	Name of the current file (used by the MAKEFILE package).
*PRINTLEVEL	The printlevel used by TRACE.
CURLIBS	List of current files. Updated by the function CURFILE.

CHAPTER 4

I/O HANDLING.

Though LISP F3 was designed to be as true a subset of INTERLISP as possible, there do exist some minor differences. Most of them have to do with I/O.

a) <u>Input characters of special meaning.</u>

.	for dotted pairs. Must be separated by blanks! A '.' which can not be interpreted as 'a dotted pair' is read as an atom.
%	escape character
'	QUOTE character
" "	string character
<>	super brackets

All those characters works the same as in INTERLISP.

~	'rescue character'. When this character is seen by the read routine, LISP F3 will enter BREAK. (Useful for infinite read loops for example).

b) <u>Changing the meaning of special characters.</u>

The "meaning" of all characters are stored in a table which is accessible by the function

(CHTAB x)	Read the type of x.
(CHTAB x n)	Change the type of x. Returns old type.

CHTAB uses the first character of the atom x.

The following character table is standard.

type	means
1	space
2	(
3	)
4	<
5	>
6	"
7	'
8	user break
9	.
10	alphanumerical
11	+
12	-
13-22	0-9
23	%
24	rescue character

Ex.: If you want to have $ as a super bracket, and > as an ordinary letter do:

```
        (SETQ TYPE (CHTAB '%> (CHTAB 'A>
        (CHTAB '$ TYPE)
```

and if you want to have * as a break character do

```
        (CHTAB '* 8)
```

after which A*B will be read as the three atoms A * B separately.

c) <u>Changing logical units etc.</u>

All I/O functions refer to a table with the following meaning:

NR	Means:
1	FORTRAN logical input nr
2	current read position
3	left margin - input
4	right margin - input
5	FORTRAN logical output nr
6	current print position
7	left margin - output
8	right margin - output
9	the print length
10	the print depth

The table is accessible by the function

(IOTAB i)	read position i in the table
(IOTAB i val)	put val in position i. Returns old value

If val is T and i is 1 or 5, the default value (= standard Input/Output) is put in position i.

A number of basic functions coded in LISP such as READPOS _INUNIT etc are defined by using IOTAB, so in practice you rarely use IOTAB yourself.

d) Changing standard behavior of LISP F3.

The function

```
        (SYSFLAG i)      Read flag i.
        (SYSFLAG i x)    Change flag i to x (=T or NIL).
                         Returns old value.
```

is used to read/write flags with the following meaning:

flag nr NIL (which is default) means:

1	no GBC message
2	output is fast printed
3	(QUOTE s) not printed as 's
5	no % or " at output
6	no automatic change to large integer at overflow
7	during pretty print, do not begin a new line if the current expression will fit on line. T means: Print sublists on separate lines, unless it is the first (or sometimes second) sub-expression.

e) Basic I/O functions.

The following functions work as in INTERLISP except that they do not have a file argument.

```
(READ)             (RATOM)            (READC)
(PRINT x)          (PRIN1 x)          (PRIN2 x)          (TERPRI)
(EJECT)            (SPACES n)
```

(One minor difference for READ and RATOM though: If a number is too large to be handled as an integer, an alphanumerical atom is returned.)

In addition the following functions are defined using IOTAB. If n is NIL they return the current value, otherwise a new value n is set and the old value is returned.

(INUNIT n)	logical input nr
(OUTUNIT n)	logical output nr
(READPOS n)	the current read position
(PRINTPOS n)	the current print position
(PRINTLEVEL n)	the max depth of printing. (Lists below this level will be printed as ...)
(PRINTLENGTH n)	the max length of printing. (Elements beyond this length will be indicated as ---)

As a matter of fact, PRINT, PRIN1 and PRIN2 are also defined in LISP using the one and only printing function PRIN0 which is defined as:

```
        (PRIN0 x a b)

        x         value to be printed (No TERPRI before or after!)
        a         =NIL Do not print % or "
                  =T   Print % or " when so necessary to read atoms
                       back.
        b         =NIL ordinary print
                  =T   pretty print with flag nr 7 = NIL
                  =n   (a number)     -"-             T
```

During pretty-print, lists headed by an atom carrying the function definition (FSUBR . QUOTE) will be treated as comments: They will be printed starting from 20 pos. to the left of the right margin.
Ex.: By doing (PUTD '* (GETD 'QUOTE)) * behaves as QUOTE and expressions like (* comment comment ...) will be printed as comments.
In addition two functions are defined:

```
        (PRINTL s1 s2 ..)
```

performs PRIN1 on s1 s2 etc.

```
        (PRINTL-SP s1 s2 ...)
```

works as PRINTL but separates s1 s2 etc. by spaces.

Finally a new function REWIND is defined:

```
        (REWIND n)       Rewinds the logical unit n.
```

f) <u>Save/Restore of the core image.</u>

The functions

```
        (ROLLOUT lu)
        (ROLLIN lu)
```

saves/restores a compact core image of the status of LISP F3. This offers you a way to dump all your functions, property-lists etc. and read them back at a later stage. (An other way of saving is to use MAKEFILE)

It is possible to perform ROLLIN also if the size of LISP F3 has been changed since the last ROLLOUT. If though the new version is too small to hold the saved core image ROLLIN returns NIL (= failure to rollin)

g) <u>The makefile package.</u>

This package is coded in LISP and follows the conventions for INTERLISP makefile (See He 76 page 98 for details). The only commands in FILEVARS which are implemented are:

```
        * (P ...) (PROP ...) (E ...) (IFPROP ...)
```

Before doing MAKEFILE (or LOAD) you must however open the file by:

```
        (OPEN file io nr)
        file    your symbolic name
        io      I or INPUT for input files
                O or OUTPUT for output files
                other       for input/output files
        nr      FORTRAN logical unit
```

and if you have no further use of the file you may close it by

```
        (CLOSE file)
```

The function

```
        (CURFILE file)
```

defines the "current file" and all new functions defined afterwards belong to this file and will be added to the list fileFNS. If (CURFILE file) is not evaluated, the name of the current file is CUR, and the function names are saved on CURFNS.

Ex.: Define some functions and save them as your file MYFILE on the logical unit 25.

```
        (OPEN 'MYFILE 'O 25)
        (CURFILE MYFILE)
        (DE ..... >
        (DE ..... > etc
        (MAKEFILE 'MYFILE T)
```

A pretty printed version of all functions is now written on 25. (argument nr 2 is used as PRINO's argument nr 3 when it performs the printout).

CHAPTER 5

ERROR HANDLING AND BREAK.

Mostly all errors detected by LISP F3 call the function SYSERROR which is a SUBR and which calls RESET after printing a message. SYSERROR is then redefined in one of the standard LISP packages as a LAMBDA function which calls BREAK1 after the message. BREAK1 is the ordinary "break-function" and may therefore also have been called by a user setup break, and inside BREAK1 the following commands exist:

```
        !       return to previous break if any. Otherwise reset.
        GO      print "broken form" and continue.
        OK      continue.
        RETURN x   return the value of x.
        EVAL    eval broken form and break afterwards.
                The value of the form is stored in the atom !VALUE
        !EVAL   as EVAL etc, but the function
        !GO        is first unbroken
        !OK        then rebroken.
        UB      unbreaks the function.
        BR      breaks the function.
        BT      backtrace of function calls (only LAMBDA and
                NLAMBDA's).
                This is only possible if you have performed
                (SETQ *BACKTRACEFLG T) before evaluation.
        ALIST   prints the current value-binding stack
                (except for variables bound in BREAK1 and SYSERROR).

        any other input is evaluated and value is printed.
```

In addition to BREAK1, the functions BREAK0 BREAK UNBREAK REBREAK TRACE are defined and work as in INTERLISP.

There also exists a function BREAK11, which is a LAMBDA version of BREAK1 (which in turn is a NLAMBDA) and a function UNTRACE.

Each error is associated with a number. The function

```
        (ERRORN)
```

returns the number for the last error occurred, and

```
        (ERRORMESS n)
```

prints out a corresponding message.

CHAPTER 6

EDIT.

Two edit functions are implemented:

(EDITF fn . edcom)	edit a function. Value = NIL.
(EDITS s edcom)	edit any s-expr. Value = s
	edcom = list of edit commands (or NIL).

The following commands are implemented.
Commands explained in HA 76 are accomplished by a page reference.

P	Print to level 2
PP	PrettyPrint to level 2
?	Print to level 1000
??	Prettyprint to level 1000

Note: In INTERLISP the print commands are not exactly as ours.

OK	p 52
UP	p 50
F	p 50
F s	p 117
NX	p 114
!	p 49
S x	Set x to the current expression. Useful in combination with US. (New command)
n	p 49
(n)	p 49
(n e1..)	p 49
(-n e1..)	p 49
(N e1..)	p 49
(R x y)	p 50
(BI n m)	p 51
(BO n)	p 51
(LI n)	p 51
(LO n)	p 51
(RI n m)	p 51
(RO n)	p 51
(: e1..)	p 114
(MBD e1..)	p117
(XTR e1..)	p117
(US x commands)	Use a copy of the saved value of x in commands
(MARK x)	Save the current chain in x.
(\ x)	Reset the edit chain to x.

S and US can be used in different edit sessions.

Ex.: Move the PROG expression of FOO to be the PROG expression of another function FII.

```
(EDITF FOO)
F PROG S DEF OK
(EDITF FII)
(US DEF (3 DEF)) OK
```

The 3:rd element (the prog expression of FII) is replaced by the one stored in DEF.

CHAPTER 7

MISCELLANEOUS.

A new function GO* is defined as a FSUBR.

```
        (GO* LAB)
```

searches through all current PROG's for a label LAB. If it is found, a jump is performed. If it is not, NIL is returned and no other action takes place.

GO* is a way of implementing ERRORSET, ERRORBANG, TRYTOEVALUATE, FAIL, etc.

Ex.:
ERRORSET is defined as:

```
(DE ERRORSET (ERRORFORM ERRFLG)
             (PROG NIL
                   (RETURN (LIST (EVAL ERRORFORM)))
              ERRORSET>
```

and SYSERROR is defined as:

```
(DE SYSERROR (ERRORTYPE FN ARG FORM)
             - print message if ERRORFLG = T -
             (GO* ERRORSET)
             (BREAK11 FORM T NIL>
```

When SYSERROR is called it tries to jump to the label ERRORSET. If it succeeds (error occurred under errorset) a "big jump" to ERRORSET is performed and the function ERRORSET returns NIL. Otherwise BREAK11 is called.

String functions:

In addition to those explained in Ha 75 (page 108) three new string functions are defined:

```
       (STRALLOC n c)
```

The first character of the literal atom (or string) c is fetched, and a new string of length n is allocated and filled with the character from c.

(PUTINT s x form)

Writes x in the (sub)string s using the format.
Value is s.

form	means:
NIL	write x's printname left justified
T	Write x binary in s.
others	Write x's printname right justified

(GETINT s form)

form	means:
T	Return the binary value of s.
others	as ([illegible] s).

GETINT and PUTINT are useful for example if you have a record (as a string) with different fields (as substrings) and want to read/write values in different fields.
Warning! The binary mode of GETINT/PUTINT may not work correctly if a word in your computer is not completely filled by bytes.

Other functions not reported in Ha 75 but [illegible] are:

(ABS n)	
(ADDLIST a l)	If memb(a,l) then l else cons(a,l)
(DSORT l)	Destructive sorting [illegible]
(EVLIS l)	mapcar(l,'EVAL)
(GCGAG flg)	Print message when [illegible] flg = T)
(NTH l n)	Performs CDR n-1 times
(RPT n s)	Evaluate s n times
(RPTQ n s)	As RPT but s is not evaluated at calling time.
(SIGN n)	[illegible] 1 or -1 depending on the sign of n.
(CLOCK)	time in milliseconds.
(RECLAIM n)	n=0 Normal GBC
	1 Compacting GBC
	2 Big number GBC
	3 Big number [illegible]
(XCALL fn l)	A way of calling FORTRAN routines.
	Returns NIL in the [illegible] system.
	See your system implementor if
	the definition has [illegible]

Note: CLOCK and RECLAIM return the value as dotted pair

(m . n)

This pair should be understood as 1000*m+n.
(The reason for this is that they should return a number too large to be stored as a small integer).

FUNCTION - FUNARG
FUNCTION FUNARG works as in LISP 1.5.

APPENDIX A

LIST OF FUNCTIONS.

NAME(ARGS)	TYPE	CHAPT.	HA 75	TE 74
ABS(N)	L	7		13.8
ADDLIST(A L)	S	7		
ADDPROP(A P V)	L		17	7.1
ADD1(N)	S		62	13.3
ADVISE(FN WHEN WHERE WHAT)	L		131	19.4-5
ALIST ()	S	2D		
ALPHORDER(A B)	S		150	6.11
AND L	FS		67	5.12
APPEND(L1 L2)	S		55	6.1
APPLY(FN L)	S		82	8.9
APPLYA(FN L AL)	S	2D		
APPLY*(FN . L)	L		82	8.10
ASSOC(A L)	L		58	5.15
ATOM(S)	S		12	5.11
BREAK 'L	NL	5	127	15.18
BREAK0(FN WHEN COMS)	L	5	127	15.16-19
BREAK1('BRKEXPR 'BRKWHEN 'BRKFN 'BRKCOMS)	NL	5	127	
BREAK11(BRKEXPR BRKWHEN BRKFN BRKCOMS)	L	5		
CAAAR(S)	S		10	5.1
CAADR(S)	S		10	5.1
CAAR(S)	S		10	5.1
CADAR(S)	S		10	5.1
CADDR(S)	S		10	5.1
CADR(S)	S		10	5.1
CAR(S)	S		10	5.1
CDAAR(S)	S		10	5.1
CDADR(S)	S		10	5.1
CDAR(S)	S		10	5.1
CDDAR(S)	S		10	5.1
CDDDR(S)	S		10	5.1
CDDR(S)	S		10	5.1
CDR(S)	S		10	5.1
CHTAB(A N)	S	4B		
CLOCK ()	S	7	149	21.3
CLOSE(FILE)	L	4G		
CONCAT L	S		108	10.7
COND(...)	FS		21	5.4
CONS(S1 S2)	S		10	5.1
COPY(S)	L		56	6.4
CURFILE('FILE)	NL	4G		

DE('FN 'ARGS . 'BODY)	NL		71	
DEFINEQ 'L	NL		73	8.7
DF('FN 'ARGS . 'BODY)	NL		71	
DIFFERENCE(N1 N2)	S		63	13.7
DSORT(L)	L	7		
EDITF('FN . 'EDCOM)	NL	6	113	9.84
EDITS(S EDCOM)	L	6	113	
EJECT ()	S	4E	92	
EQ(S1 S2)	S		11	5.11
EQUAL(S1 S2)	S		11	5.12
ERRORB ()	L		121	16.12-13
ERRORMESS(N)	S	5		16.13
ERRORN ()	L	5		16.13
ERRORSET(ERRFORM ERRFLG)	L	7	121	16.14
EVAL(S)	S		82	8.9
EVALA(S AL)	S	2D		8.10
EVLIS(L)	S	7		
EXIT ()	S		29	
FUNCTION('FN)	FS		86	11.1
GCGAG(FLAG)	L	7		10.15
GENSYM ()	S		108	10.4-5
GETD(FN)	S		73	8.3
GETINT(S FORM)	S	7		
GETP(A P)	S		17	7.3
GO('LAB)	FS		81	5.7
GO*('LAB)	FS	7		
GREATERP(N1 N2)	S		63	13.8
INUNIT(N)	L	4E		
IOTAB(N1 N2)	S	4C		
LAST(L)	L		57	6.7
LENGTH(L)	S		57	6.8
LESSP(N1 N2)	S		63	13.8
LISPX ()	S		95	22.47
LIST L	S		55	6.1
LISTP(S)	S		55	5.11
LITATOM(S)	S		54	5.11
LOAD(FILE)	L		98	14.27
MAKEFILE(FILE FLAG)	L		98	14.45-48
MAP(L FN1 FN2)	S		86	11.2
MAPC(L FN1 FN2)	S		87	11.3
MAPCAR(L FN1 FN2)	S		86	11.3
MAPLIST(L FN1 FN2)	S		87	11.3
MEMB(A L)	S		12	5.14
MEMBER(A L)	S		55	5.14
MINUS(N)	L		63	13.7
MINUSP(N)	L		62	13.6
MKATOM(S)	L		109	10.7
MKSTRING(S)	L		108	10.5
NCHARS(S)	S		107	10.3
NCONC(L1 L2)	S		104	6.2-3
NCONC1(L A)	S		104	6.2-3
NEQ(S1 S2)	S		55	5.12
NLISTP(S)	S		55	5.11
NTH(L N)	L	7		6.8
NULL(S)	S		12	5.12
NUMBERP(S)	S		61	5.11
OBLIST(A)	S	2C		
OPEN(FILE OPT N)	L	4G		
OR L	FS		67	5.13
OUTUNIT(N)	L	4E		
PACK(S FLAG)	S		107	10.2
PLUS L	S		63	13.7
PP 'L	NL		92	14.29

PRINT(S)	L	4E	91	14.19
PRINTDEF(S)	L		92	14.38-39
PRINTL L	L	4E		
PRINTLENGTH(N)	L	4E		
PRINTLEVEL(N)	L	4E		14.19
PRINTL-SP L	L	4E		
PRINTPOS(N)	L	4E		
PRIN0(S ESCFLG PPFLG)	S	4E		
PRIN1(S)	L	4E	91	14.18-19
PRIN2(S)	L	4E	91	14.18-19
PROG(...)	FS		80	5.6
PROGN L	FS		54	5.6
PROG1 L	S		54	5.6
PUT(A P V)	S		17	7.1-2
PUTD(FN S)	L		73	8.4
PUTINT(S X FORM)	S	7		
QUOTE('S)	FS		22	5.3
QUOTIENT(N1 N2)	S		63	13.7
RATOM ()	S	4E	90	14.11-13
READ ()	S	4E	90	14.10-11
READC ()	S	4E	90	14.14
READPOS ()	L	4E		
READVISE 'L	NL		131	19.8
REBREAK 'L	NL	5	127	15.22-23
RECLAIM(N)	S	7	149	10.14
REMOVE(A L)	L		56	6.4
REMPROP(A P)	L		17	7.2
RESET ()	S	5	121	16.13
RETURN(S)	S		81	5.7
REVERSE(L)	S		56	6.4
REWIND(N)	S	4E		
ROLLIN(N)	S	4F		
ROLLOUT(N)	S	4F		
RPLACA(S1 S2)	S		101	5.3
RPLACD(S1 S2)	S		102	5.2
RPLSTRING(S1 N S2)	S		109	10.7
RPT(N S)	S	7		8.10-11
RPTQ(N 'S)	NL	7		8.10-11
SASSOC(A L)	L		58	5.15
SAVEDEF(FN)	L		73	8.7-8
SELECTQ(...)	FS		53	5.4-5
SET(A S)	S		25	5.8
SETQ('A S)	FS		26	5.8
SETQQ(A S)	NL		54	5.8
SIGN(N)	L	7		
SPACES(N)	L	4E	91	14.19
STRALLOC(N C)	S	7		
STREQUAL(S1 S2)	L		108	10.5
STRINGP(S)	L		108	5.11
SUBST(A B S)	S		56	6.5
SUBSTRING(S N1 N2)	S		109	10.6
SUB1(N)	S		62	13.3
SYSERROR(ERRTYPE FN ARG FORM)	L	5,7		
SYSFLAG(N)	S	4D		
TERPRI ()	S	4E	91	14.19
TIMES L	S		63	13.7
TRACE 'L	NL	5	127	15.18-19
UNADVISE 'L	NL		131	19.8
UNBREAK 'L	NL	5	127	15.21
UNPACK(S FLAG)	S		107	10.2-3
UNSAVEDEF(FN)	L		73	8.8
UNTRACE 'L	NL	5		
VIRGINFN(FN)	L		129	15.23
XCALL(FN L)	S	7		
ZEROP(N)	S		62	13.4

APPENDIX B

REFERENCES.

Ha 75	A. Haraldsson:	"LISP-DETAILS. INTERLISP 360/370" DLU 75/9
Mn 71	M. Nordstrom:	"LISP F1 - A FORTRAN Implementation of Lisp 1.5". DLU 71.
Mn 78	M. Nordstrom:	"LISP F3 - Implementation Guide". DLU 78/3.
Te 74	W. Teitelman:	"INTERLISP REFERENCE MANUAL" XEROX corp.