A Summary of
MacLisp
Functions and Flags

David S. Touretzky

August, 1979

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-78-C-1551.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

The author was supported by a Fannie and John Hertz Foundation fellowship.

Table of Contents

1. Manipulating S-Expressions 2
    - 1.1. Basic List Structure 2
    - 1.2. Extracting Components of Lists 3
    - 1.3. Predicates on S-Expressions 3
    - 1.4. Searching and Substitution 4
    - 1.5. Hashing List Structure 5
    - 1.6. Sorting 5
    - 1.7. Hunks 6
2. Type Predicates 6
3. Atomic Symbols 7
    - 3.1. Symbols As Variables 7
    - 3.2. The Property List 8
    - 3.3. Characters and Print-Names 9
    - 3.4. The OBARRAY 10
4. Numbers 11
    - 4.1. Predicates on Numbers 11
    - 4.2. Conversion 12
    - 4.3. General Arithmetic 13
    - 4.4. Fixnum Arithmetic 14
    - 4.5. Flonum Arithmetic 15
    - 4.6. Logs and Powers 15
    - 4.7. Trigonometric Functions 16
    - 4.8. Logical Operations on Numbers 16
    - 4.9. Miscellaneous 17
5. Programs 17
    - 5.1. The Evaluator 17
    - 5.2. Evaluator Special Forms 18
    - 5.3. PROG Forms 19
    - 5.4. Conditionals 21
    - 5.5. LEXPRS and LSUBRS 21
    - 5.6. Non-Local Exits 22
    - 5.7. Error Signalling 23
6. Mapping Functions 23
7. Arrays 24
8. Input/Output 26
    - 8.1. Functions On Files 26
    - 8.2. Functions on Filespecs and File Objects 27
    - 8.3. Basic I/O 28
    - 8.4. Character I/O 29
    - 8.5. General I/O Control 30
    - 8.6. Terminal I/O 31
    - 8.7. Binary and Random Access I/O 32
    - 8.8. Miscellaneous Functions 33
    - 8.9. OLDIO Functions 33
9. Programming Tools 35
    - 9.1. Common Functions 35
    - 9.2. Packages 36
10. Storage Management 36
    - 10.1. Garbage Collection 36
    - 10.2. Storage Allocation Concepts 37
    - 10.3. Storage Spaces 38
    - 10.4. Allocation 39

11. Status Functions 39
    - 11.1. Environment Enqueries 39
    - 11.2. Garbage Collector Status 41
    - 11.3. I/O Status 42
    - 11.4. Time 43

Index 45

Preface

This document is a summary of CMU MacLisp's principle functions and flags. It is not a complete list, and certainly not a reference manual.

Many of the items listed here are taken from chapters 1 through 3 of the MacLisp Reference Manual. Since the manual is incomplete, the remainder are drawn from information in a cumulative file of MacLisp update notices, ARCHIV.DOC[C380ML5P]/A.

A special notation is used to indicate the calling syntax for functions:

- Evaluated arguments appear as bare atoms, such as X and Y in (EQ X Y).
- Unevaluated arguments, i.e. arguments taken by foxprs and left unevaluated, appear in quotation marks, e.g. (SETQ "X" Y).
- Arguments that are destructively modified by a function are preceded by an asterisk, as in (RPLACD *X Y).
- Optional arguments appear in brackets, e.g. (TERPRI [FILE])
- Numbers in brackets refer to pages of the Maclisp Reference Manual.

Building a summary from a mostly non-existent reference manual is a difficult task. To simplify things, system-level features (such as the interrupt system, pure pages, and the evalhook mechanism) have been omitted. Also omitted are those features that are not applicable to the version of MacLisp in use at CMU. Please mail all corrections to this summary to MacLisp@CMUA.

1. Manipulating S-Expressions

1.1. Basic List Structure

(CONS X Y) [2-16]

Eg: (CONS 'A 'B) = (A . B)

(NCONS X) [2-16]

Same as (CONS X NIL)

(XCONS X Y) [2-17]

Same as (CONS Y X)

(LIST X_1 X_2 ... X_N) [2-19]

Returns a list containing the X_i. Eg: (LIST 'A 'B 'C) = (A B C)

(LIST X_1 X_2 ... X_N)* [?]

An lsubr version of CONS. Eg: (LIST* 'A 'B 'C) = (A B . C)

(APPEND X_1 X_2 ... X_N) [2-19]

Returns a list of all the X_i appended together. This is a non-destructive append: all but the last argument are copied at the top level. Eg: (APPEND '(A B) '(C D) '(E)) = (A B C D E)

*(NCONC *X_1 *X_2 ... X_N)* [2-20]

Similar to APPEND, but all X_i except the last are modified rather than copied. Returns the modified X_1.

(REVERSE L) [2-20]

Returns the reverse of the top-level list L.

*(NREVERSE *L)* [2-21]

Like REVERSE, but destructive.

*(NRECONC *X Y)* [2-21]

Same as (NCONC (NREVERSE X) Y)

*(RPLACA *X Y)* [2-22]

Physically replaces the CAR portion of X with Y, returning the modified X.

*(RPLACD *X Y)* [2-22]

Physically replaces the CDR portion of X with Y, returning the modified X.

(LENGTH L) [2-18]

Returns the number of top-level elements of the list L.

1.2. Extracting Components of Lists

(CAR L) [2-15]

Eg: (CAR '(A B C)) = A

(CDR L) [2-15]

Eg: (CDR '(A B C)) = (B C)

(C....R L) [2-16]

Composite CAR's and CDR's, up to four deep. Eg: (CADDR L) = (CAR (CDR (CDR L)))

(NTH N L) [?]

Returns the Nth element of list L, with 0 being the first element. Eg: (NTH 1 '(A B C)) = B

(NTHCDR N L) [?]

Returns the result of taking the CDR of list L, repeated N times. Eg: (NTHCDR 1 '(A B C)) = (B C)

(LAST L) [2-18]

The last cons cell of the list L. Eg: (LAST '(A B C)) = (C), (LAST '(A B . C)) = (B . C)

1.3. Predicates on S-Expressions

(EQ X Y) [2-3]

Returns T if objects X and Y are the same pointer. EQ will correctly compare symbols and lists, but not numbers. Eg: (EQ 'A 'A) = T, but (EQ '(A) '(A)) = NIL

(EQUAL X Y) [2-3]

Returns T if objects X and Y are identical s-expressions. Eg: (EQUAL '(FOO BAR) '(FOO BAR)) = T

(NULL X) [2-4]

Returns T if X is NIL, otherwise returns NIL.

(NOT X) [2-4]

Same as NULL. Returns T if X is NIL, otherwise returns NIL.

(MEMBER X L) [2-24]

If X is EQUAL to any top-level element of Y, then the tail of Y starting with the point where X is found is returned. Otherwise NIL is returned. Eg: (MEMBER 'C '(A B C D E)) = (C D E)

(MEMQ X L) [2-25]

Like MEMBER, but uses EQ instead of EQUAL.

1.4. Searching and Substitution

(SUBST X Y L) [2-22]

Substitutes X for all elements EQ to Y in L. Returns L.

(SUBLIS A L) [2-23]

Uses the list of dotted pairs A to make substitutions in L. Eg: (SUBLIS '((A . FOO) (B . BAR)) '(SETQ A B)) = (SETQ FOO BAR)

*(DELETE X *L [N])* [2-25]

(DELETE X L) returns list L after all elements EQUAL to X have been destructively removed. DELETE should be used with a SETQ, not by itself, as old pointers to the list L may be left pointing to a deleted element. (DELETE X L N) will delete only the first N occurrences of X from L.

*(DELQ X *L [N])* [2-26]

Like DELETE, but uses EQ instead of EQUAL.

(ASSOC X L) [2-27]

Search the list of dotted pairs L for a pair whose CAR is EQ to X. Returns the first such pair found, else NIL. Eg: (ASSOC 'TWO '((ONE . 1) (TWO . 2) (THREE . 3))) = (TWO . 2)

(ASSQ X L) [2-28]

Like ASSOC, but uses EQ instead of EQUAL.

(SASSOC X L FN) [2-28]

Like ASSOC, but if X can't be found in the association list L, returns the value of a call to function FN, a function of zero arguments.

(SASSQ X L FN) [2-29]

Like ASSQ, but if X can't be found in the association list L, returns the value of a call to function FN, a function of zero arguments.

1.5. Hashing List Structure

(SXHASH X) [2-26]

Hashes an s-expression into a fixnum. EQUAL s-expressions hash to the same number.

(MAKNUM X) [2-29]

Translates an object into a fixnum, by returning the memory address of object X.

(MUNKAM N) [2-29]

Opposite of MAKNUM. Returns the object which was given to MAKNUM to get the number (memory address) N.

1.6. Sorting

*(SORT *X FN)* [2-30]

Destructively sorts the list or array X, using FN as a predicate to compare pairs of elements. FN should return T if the first arguement should appear before the second in the sorted list. For alphabetical sorting, use ALPHALESSP as the predicate.

*(SORTCAR *X FN)* [2-31]

Like SORT, but calls the predicate on the CAR's of the elements.

1.7. Hunks

(HUNK X_1 X_2 ... X_{N-1} X_0) [2-32]

Builds a hunk from the X_i. Note that the 0'th element appears last in the argument list. Hunk sizes are always a power of two, no matter how many arguments are actually given.

(CXR N H) [2-33]

Returns the Nth component of hunk H.

*(RPLACX N *H X)* [2-33]

Physically replaces the Nth component of hunk H with X, and returns H.

(MAKHUNK N) [2-33]

Creates and returns an N-element hunk, filled with NILs. (MAKHUNK L), where L is a list, creates a hunk of the appropriate size and initalizes it from L.

(HUNKSIZE H) [2-33]

Returns the number of components in hunk H.

HUNKP [2-33]

If the global variable HUNKP is NIL, the functions PRINT, EQUAL and PURCOPY treat hunks as conses. If non-NIL (the default), hunks are treated as hunks.

2. Type Predicates

(ATOM X) [2-1]

Returns T if argument is any kind of atomic object, such as a symbol or a number, otherwise NIL.

(SYMBOLP X) [2-1]

Returns T if X is an atomic symbol, otherwise NIL.

(FIXP X) [2-1]

Returns T if X is a fixnum or bignum, otherwise NIL.

(FLOATP X) [2-1]

Returns T if X is a flonum, otherwise NIL.

(BIGP X) [2-1]

Returns T if X is a bignum, otherwise NIL.

(NUMBERP X) [2-2]

Returns T if X is any kind of number, otherwise NIL.

(HUNKP X) [2-2]

Returns T if X is a hunk, otherwise NIL.

(TYPEP X) [2-2]

Returns an atomic symbol describing the type of object X. Possible values are FIXNUM, FLONUM, BIGNUM, LIST, SYMBOL, STRING, ARRAY, and RANDOM.

3. Atomic Symbols

3.1. Symbols As Variables

(SETQ "X" Y) [2-49]

The canonical assignment statement. Sets the value of variable X to Y. X is left unevaluated, Y is not. More than one variable may be set at once, eg (SETQ X 3 Z 4).

(SET X Y) [2-50]

Like SETQ, but X is evaluated and must yield an atomic symbol.

(PUSH X "L") [?]

Equivalent to (SETQ L (CONS X L)). Use where L is acting as a stack.

(POP "L" ["X"]) [?]

Returns CAR of L, setting L to CDR of L. (i.e. pops the top element off a stack and returns it.) Assigns the popped value to the (optional) variable X.

(SYMEVAL X) [2-50]

Returns the value of atomic symbol X. More efficient than doing an ordinary EVAL.

(BOUNDP X) [2-51]

Returns T if atom X has a value, otherwise NIL.

(MAKUNBOUND X) [2-51]

Removes any value associated with atomic symbol X.

3.2. The Property List

(GET X P) [2-53]

Returns the P property of atomic symbol X, or NIL if there is no such property.

(GETL X L) [2-53]

Returns a portion of the property list of symbol X beginning with the first property in the list L, or NIL if X has no properties in L.

(PUTPROP X V P) [2-54]

For atomic symbol X, make V be the P property.

(DEFPROP "X" "V" "P") [2-54]

Like PUTPROP, but arguments are left unevaluated. Eg: (DEFPROP JOHN MALE SEX) = (PUTPROP 'JOHN 'MALE 'SEX)

(REMPROP X P) [2-55]

Remove X's P property. Returns a portion of the property list beginning with property X, or NIL. X may be an atomic symbol or any list that looks like a property list.

(PLIST X) [2-55]

Returns the property list of atomic symbol X. Note that in MacLisp the value cell and print-name are not kept on the property list.

(SETPLIST X L) [2-55]

Sets the property list of atomic symbol X to L.

3.3. Characters and Print-Names

(ASCII N) [2-83]

Returns the character object for ASCII code N.

(GETCHAR X N) [2-83]

Returns the Nth character of X's print-name, starting from 1. The character is returned as a character object.

(GETCHARN X N) [2-83]

Same as GETCHAR, except the character is returned as a fixnum instead of a character object.

(PNGET X N) [2-57]

Returns the print-name of atom X as a list of fixnums containing packed N-bit bytes. N may be 6 or 7.

(PNPUT L FLAG) [2-57]

Creates a new symbol whose print-name is defined by the list of fixnums L, and interns it if FLAG is non-NIL. L is assumed to contain packed 7-bit bytes.

(EXPLODE X) [2-85]

Returns a list of characters, which are the characters that would be typed out if (PRIN1 X) were done, including slashes for special characters but not including extra newlines that PRIN1 would insert to prevent exceeding the page width. Each character is represented by a character object.

(EXPLODEC X) [2-85]

Like EXPLODE, but in the form of PRINC rather than PRIN1, i.e. special characters aren't slashified.

(EXPLODEN X) [2-85]

Like EXPLODEC, but returns a list of fixnums rather than character objects.

(FLATSIZE X) [2-85]

Returns the number of characters PRIN1 would use to print X.

(FLATC X) [2-85]

Returns the number of characters PRINC would use to print X, i.e. without slashifying special characters.

(MAKNAM L) [2-84]

Creates an uninterned atomic symbol whose print-name is created from the characters in the list L.

(IMPLODE L) [2-84]

Same as MAKNAM, except the atom is interned.

(READLIST L) [2-84]

Creates a new atom or list by parsing the character sequence in the list L. All atoms are interned. Inverse of EXPLODE.

(SAMEPNAMEP X Y) [2-56]

Returns T if atoms X and Y have the same print-name.

(ALPHALESSP X Y) [2-56]

Returns T if the print-name of atom X is lower in the ASCII collating sequence than the print-name of atom Y.

3.4. The OBARRAY

(INTERN X) [2-58]

Returns from the obarray the unique atomic symbol whose print-name is identical to that of X. If there is no such symbol, X itself is added to the obarray and returned as value.

(REMOB X) [2-59]

Removes atomic symbol X from the obarray.

(COPYSYMBOL X FLAG) [2-59]

Creates and returns a new, uninterned symbol whose print-name is the same as that of X. If FLAG is non-NIL, X's value and properties are also copied into the new atom.

(GENSYM X) [2-59]

Generates and returns a new, uninterned atomic symbol, whose name is derived from a counter and a one-letter prefix. (GENSYM) returns the next such symbol. (GENSYM N) sets the counter to N and returns a new symbol. (GENSYM X) sets the prefix to the first character of X's print-name and returns a new symbol.

4. Numbers

4.1. Predicates on Numbers

(ZEROP X) [2-63]

Returns T if X is zero.

(PLUSP X) [2-63]

Returns T if X is greater than zero.

(MINUSP X) [2-63]

Returns T is X is less than zero.

(ODDP X) [2-63]

Returns T if X is odd. X must be a fixnum or bignum.

(SIGNP "C" X) [2-63]

General predicate for testing the sign of a number. C is not evaluated; it must be one of L, LE, E, N, GE, or G. Returns T if the specified relation between X and zero is true.

(= X Y) [2-65]

Returns T if X and Y are numerically equal. X and Y may be fixnums or flonums, but must be of the same type.

(> X Y) [2-65]

Returns T if X is numerically greater than Y. X and Y may be fixnums or flonums, but must be of the same type.

(< X Y) [2-66]

Returns T if X is numerically less than Y. X and Y may be fixnums or flonums, but must be of the same type.

(GREATERP X_1 X_2 ... X_N) [2-65]

Compares the X_i from left to right, and returns T if each is greater than the next.

(LESSP X_1 X_2 ... X_N) [2-65]

Compares the X_i from left to right, and returns T if each is less than the next.

(MAX X_1 X_2 ... X_N) [2-66]

Returns the largest of the X_i. If any argument is a flonum, the result will be a flonum; otherwise the result is either a fixnum or a bignum.

(MIN X_1 X_2 ... X_N) [2-66]

Returns the smallest of the X_i. If any argument is a flonum, the result will be a flonum; otherwise the result is either a fixnum or a bignum.

(HAULONG X) [2-64]

Returns the number of signifigant bits in X, which must be a fixnum of bignum. The result is the least integer not less than the base-2 log of abs(X)-1.

4.2. Conversion

(FIX X) [2-67]

Converts X to a fixnum or bignum, depending on its magnitude.

(IFIX X) [2-67]

Converts X from a flonum to a fixnum. IFIX never returns a bignum; this allows it to compile more efficiently. Rounding is always down, as in the Algol ENTIER function.

(FLOAT X) [2-67]

Converts X to a flonum.

(ABS X) [2-67]

Returns the absolute value of X.

(HAIPART X N) [2-68]

Returns the N leading bits of the internal representation of abs(X). X must be a fixnum or bignum. If N is negative, the N trailing bits of abs(X) are returned.

4.3. General Arithmetic

(PLUS X_1 X_2 ... X_N) [2-69]

Returns the sum of 0 or more arguments, which may be any type of numbers.

(DIFFERENCE X_1 X_2 ... X_N) [2-69]

Returns the first argument minus the rest of the arguments. Works for any type of number.

(MINUS X) [2-68]

Returns the negative of its argument.

(TIMES X_1 X_2 ... X_N) [2-69]

Returns the product of 0 or more arguments, which may be any type of numbers.

(QUOTIENT X_1 X_2 ... X_N) [2-69]

Returns the first argument divided by the rest of the arguments. Works for any kind of numbers.

(ADD1 X) [2-70]

Adds 1 to X.

(SUB1 X) [2-70]

Subtracts 1 from X.

(REMAINDER X Y) [2-70]

Returns the remainder after dividing X by Y. The sign of the remainder will be the same as that of X. Works for fixnums or bignums.

(GCD X Y) [2-70]

Returns the greatest common divisor of X and Y. Arguments must be fixnums or bignums.

(EXPT X Y) [2-70]

Raises X to the Y power. If Y is a bignum, X must be 0, 1 or -1. If Y is a flonum, X is converted to floating point and the exponentiation is done using logarithms.

*(*DIF X Y)* [2-70]

Obsolete, 2-argument version of DIFFERENCE.

*(*QUO X Y)* [2-71]

Obsolete, 2-argument version of QUOTIENT.

4.4. Fixnum Arithmetic

(+ X_1 X_2 ... X_N) [2-72]

Returns the sum of the integers X_i.

(- X_1 X_2 ... X_N) [2-72]

Returns the first argument minus the rest. All must be integers. If called with only one argument, returns its negation.

(X_1 X_2 ... X_N)* [2-72]

Returns the product of the integers X_i.

(// X_1 X_2 ... X_N) [2-73]

Integer division. Returns the first argument divided by the rest. If called with only one argument, returns its integer reciprocal, which is -1, 0, 1, or undefined.

(1+ X) [2-73]

Adds 1 to the integer X.

(1- X) [2-73]

Subtracts 1 from the integer X.

(\ X Y) [2-73]

Returns the remainder of the integer division of X by Y. The result will have the sign of X.

(\\ X Y) [2-73]

Fixnum version of the gcd function. Returns the greatest common divisor of X and Y.

(^ X Y) [2-74]

Fixnum exponentiation. Always uses fixnum arithmetic; will be incorrect if the result is too large.

4.5. Flonum Arithmetic

(+$ X_1 X_2 ... X_N) [2-75]

Returns the floating point sum of the X_i.

(-$ X_1 X_2 ... X_N) [2-75]

Floating point subtraction. Returns the first argument minus the rest. When called with only one argument, returns its negation.

($ X_1 X_2 ... X_N)* [2-75]

Returns the floating point product of the X_i.

(//$ X_1 X_2 ... X_N) [2-76]

Floating point division. Returns the first argument divided by the rest. When called with only one argument, returns its reciprocal.

(1+$ X) [2-76]

Adds 1.0 to X, which must be a flonum.

(1-$ X) [2-76]

Subtracts 1.0 from X, which must be a flonum.

(^$ X Y) [2-76]

Floating point exponentiation. The first argument must be a flonum, the second must be a fixnum. To raise a flonum to a floating power, use (EXPT X Y) or (EXP (*$ Y (LOG X))).

4.6. Logs and Powers

(LOG X) [2-77]

Returns the natural log of X.

(EXP X) [2-77]

Returns e^X.

(SQRT X) [2-77]

Returns the square root of X. More accurate than (EXPT X 0.5)

4.7. Trigonometric Functions

(SIN X) [2-78]

Returns the trigonometric sine of X, which may be a fixnum or flonum. X is in radians.

(COS X) [2-78]

Returns the cosine of X, which may be a fixnum or flonum. X is in radians.

(ATAN X Y) [2-78]

Returns the arctangent of x/y, in radians. X and Y may be fixnums or flonums. Y may be 0 as long as X is not also 0.

4.8. Logical Operations on Numbers

(BOOLE K X Y) [2-80]

Computes a bit-by-bit Boolean function on X and Y. The function is specified by K, which must be a fixnum between 0 and 15. The four bits of K, from left to right, specify the result of the Boolean function when (X,Y) is (0,0), (1,0), (0,1), and (1,1). If BOOLE is called with more than three arguments, the function is applied to the first two numbers, then to the result and the third number, etc. Some common values for K are: 1 for logical And, 7 for logical Or, and 6 for logical Xor.

(LSH X Y) [2-81]

Logically shifts the bits of X by Y places, to the left if Y is positive, else to the right. X and Y must be fixnums. The result is undefined if abs(Y) exceeds 36.

(ROT X Y) [2-81]

Rotates the bits of X by Y places, to the left if Y is positive, else to the right. X and Y must be fixnums. The result is undefined if abs(Y) exceeds 36.

(FSC X Y) [2-82]

Performs an FSC instruction on the floating point numbers X and Y. Consult the PDP-10 processor manual for details.

4.9. Miscellaneous

(RANDOM X) [2-79]

(RANDOM X) returns a random fixnum between 0 and X-1 inclusive. Also, (RANDOM) returns a random fixnum, (RANDOM X Y) uses X and Y to set the random number seed, and (RANDOM NIL) restarts the random sequence from the beginning.

ZUNDERFLOW [2-79]

If the global variable ZUNDERFLOW is non-NIL, floating point underflows will return 0.0 as a result. If NIL, floating point underflows will be treated as errors. The initial value of ZUNDERFLOW is NIL. This flag has no effect on compiled arithmetic operations that were open-coded. Also see (SSTATUS DIVOV), which controls division by zero.

5. Programs

5.1. The Evaluator

(EVAL X [P]) [2-7]

Evaluates x as a LISP form and returns the result. (EVAL X P) evaluates X using binding context pointer P. Eg: (EVAL '(CONS 'A 'B)) = (A . B)

(APPLY FN L [P]) [2-7]

Applies function FN to argument list L. The arguments in the list L are used without further evaluation. (APPLY FN L P) applies function FN to argument list L using binding context pointer P.

(FUNCALL FN X_1 X_2 ... X_N) [2-13]

Calls function FN with arguments X_i. Similar to APPLY, except the arguments are specified individually instead of as a list. Should not be used with fexprs or fsubrs.

(SUBRCALL "TYPE" P X_1 X_2 ... X_N) [2-13]

Used to invoke a subr pointer directly rather than through an atomic symbol with a subr property. All arguments except the first are evaluated. TYPE is the type of result expected, either FIXNUM, FLONUM, or NIL (any type). P is the subr pointer to be called; the X_i are its arguments.

(LSUBRCALL "TYPE" P X_1 X_2 ... X_N) [2-13]

Like SUBRCALL, except the pointer P must be to an lsubr instead of a subr.

(ARRAYCALL "TYPE" P X_1 X_2 ... X_N) [2-13]

Like SUBRCALL, except an array pointer is used instead of a subr pointer. TYPE must match the type of the array when it was created. An ARRAYCALL may be used as a first argument to STORE.

5.2. Evaluator Special Forms

(QUOTE "X") [2-7]

Returns X without evaluating it. This is the standard way to include s-expression constants in a LISP form. (QUOTE X) is entirely equivalent to 'X. Eg: (QUOTE (FOO BAR)) or '(FOO BAR) evals to (FOO BAR)

(FUNCTION "X") [2-8]

Like QUOTE, but indicates that the expression is a LISP form that may be compiled. Useful for passing functional arguments to map functions and the like. FUNCTION does not worry about the "funarg problem".

*(*FUNCTION "X")* [2-9]

Like FUNCTION, but handles the "funarg problem" by generating a binding context pointer that is passed along with the functional argument.

BACKQUOTE [?]

Like QUOTE, but a comma within the argument causes the following s-expression to be evaluated, and the sequence ,@ causes the next s-expression to be evaluated and spliced in. Implemented via a macro character (`) called the backquote. Eg: Let A = FOO and B = (BAR BAZ). Then `(ALL ,A ARE ,(CAR B)) = (ALL FOO ARE BAR), and `(,A ,@B) = (FOO BAR BAZ). The comma is a reserved character used by backquote.

(LAMBDA ARGS F_1 F_2 ... F_N) [1-15]

The mechanism for binding formal to actual parameters in a function call. ARGS is the argument list, the forms F_i are evaluated in sequence and the value of F_N returned. If ARGS is an atom instead of a list, the atom will be bound to the number of actual arguments passed, and the function is called a lexpr. LAMBDA isn't itself a function, it is a special form that is recognized by the evaluator as denoting a functional form. Thus a lambda expression may appear wherever an atomic function name could appear. Eg: ((LAMBDA (X) (TIMES X X)) 5) = 25

(LABEL NAME LAMBDA-EXPRESSION) [1-17]

A somewhat obscure method of writing recursive expressions, rather than the usual recursive functions. During the interpretation of the LABEL special form, NAME is a local variable bound to the given lambda expression. However, MacLisp does not allow variables in function position, so APPLY or FUNCALL must be used to call the expression.

(COMMENT ...) [?]

The comment function. Ignores its arguments, and returns COMMENT. This is not the same as semicolon-style comments.

(DECLARE $DECL_1$ $DECL_2$ ... $DECL_N$) [?]

In the interpreter, DECLARE is treated as a comment. In the compiler, each of the $DECL_i$ are interpreted as declarations or compiler directives, generally by evaluating them.

5.3. PROG Forms

(PROGN F_1 F_2 ... F_N) [2-11]

Evaluates the forms F_i in sequence and returns the value of the last one.

(PROG2 F_1 F_2 ... F_N) [2-10]

Like PROGN, but returns the value of F_2 no matter how many arguments it receives. Useful for hacking obscure side effects.

(PROGV VARS VALS F_1 F_2 ... F_N) [2-11]

Evaluates VARS to get a variable list and VALS to get a list of values. Binds the values to the variables, then evaluates the F_i and returns the last result. Useful for super-powerful binding control.

(PROG VARS F_1 F_2 ... F_N) [2-38]

The "program" special form. VARS is a list of local variables which are initialized to NIL when the PROG is entered. The F_i are evaluated sequentially unless a function such as GO is called to alter the flow of control. Atomic F_i are taken as program labels. PROG returns NIL unless an explicit RETURN function is executed.

(GO "TAG") [2-42]

Alters the flow of control of a DO or PROG to proceed from the point named by TAG. If TAG is not an atom it will be evaluated and should yield one. GO may not be used to branch outside the current PROG.

(RETURN X) [2-43]

Forces the current DO or PROG to return with value X.

(DO VARLIST EXITLIST F_1 F_2 ... F_N) [2-40]

All-powerful iteration facility. VARLIST is a list of entries (VAR INIT REPEAT), where VAR is a variable name, INIT an expression yielding an initial value, and REPEAT an expression for iterating that variable's value. EXITLIST is a list (E_1 E_2 ... E_n), where E_1 is a termination predicate. If E_1 returns a non-NIL value the rest of the E_i are evaluated and the value of the last is returned. Otherwise the forms F_i are evaluated in sequence up to F_N, then the DO variables are iterated, etc. The DO body is like that of a PROG, i.e. it may include labels and GO and RETURN statements. See the MacLisp Reference Manual for examples.

(DO VAR INIT REPEAT TEST F_1 F_2 ... F_N) [2-41]

The old DO, less general than the one described above. The VAR bound by the DO, a single variable, receives initial value INIT and is iterated until TEST returns non-NIL. The F_i are evaluated on each iteration. Eg: the following prints the numbers 1 through 10: (DO I 1 (1+ I) (> I 10) (PRINT I))

5.4. Conditionals

(AND F_1 F_2 ... F_N) [2-36]

Evaluates the F_i in sequence. If any one returns NIL, AND returns NIL without evaluating the rest. Otherwise the value of F_N is returned. Eg: (AND (NOT (ZEROP X)) (QUOTIENT 1 X))

(OR F_1 F_2 ... F_N) [2-26]

Evaluates the F_i in sequence. If any one returns a non-NIL value, that value is returned immediately. Otherwise NIL is returned. Eg: (OR (NULL X) (PRINT (CAR X)))

(COND (P_l $E_{l,1}$ $E_{l,2}$ ...) ...) [2-36]

Generalized conditional facility. The P_i are evaluated in sequence until one is found that returns a non-NIL value, then all $E_{i,j}$ of that P_i are evaluated and the value of the last is returned. If there are no $E_{i,j}$ for that P_i, the value of P_i itself is returned. If no P_i evaluates to non-NIL, the COND returns NIL.

(CASEQ SEL ("A_l" $E_{l,1}$ $E_{l,2}$ ...) ...) [?]

SEL is evaluated and yields an atom. If the atom is EQ to any unevaluated atom A_i, the $E_{i,j}$ of that A_i are evaluated and the value of the last one is returned. If an A_i is a list, the test is MEMQ instead of EQ. An "else" clause can be obtained by making A_N be the atom T. If no test is satisfied, CASEQ returns NIL.

5.5. LEXPRS and LSUBRS

(ARG N) [2-12]

(ARG N) where N is a number returns the value of the Nth argument to the lexpr. (ARG NIL) returns the number of arguments that were passed. This is also the value that the lexpr's single lambda variable is bound to.

(SETARG N X) [2-12]

Sets the lexpr's Nth argument to X. This is the equivalent of doing an assignment to a lambda variable of an expr or fexpr.

(LISTIFY N) [2-13]

Returns a list of the lexpr's first N arguments. If N is negative, returns a list of the lexpr's last N arguments.

5.6. Non-Local Exits

*(*CATCH TAG E_1 E_2 ... E_N)* [?]

Receiving half of the non-local exit mechanism. Evaluates the E_i in sequence and returns the value of the last one if no non-local exit is forced. If a *THROW (or THROW) whose tag matches the first argument to the *CATCH is executed by one of the E_i, the value returned is the the value of the *THROW. If the tag doesn't match the first argument, the non-local exit searches down the stack for the next *CATCH, CATCH, CATCHALL, or CATCH-BARRIER.

*(*THROW TAG VAL)* [?]

Forces a non-local exit to occur, passing along both the tag and the return value. At some higher level the exit will be caught. If it is by a *CATCH (or CATCH), the value is passed to the catcher. If by CATCHALL, both the tag and the value are passed.

(CATCHALL FN E_1 E_2 ... E_N) [?]

Has the same semantics as *CATCH, except that all *THROWs, independent of tag, will be caught. FN must be a function of two arguments. If a non-local exit occurs, FN will be called on the tag and value passed by the *THROW. FN may itself issue a *THROW, in which case the CATCHALL acts like a filter between the exiting function and higher levels.

(CATCH-BARRIER TAGLIST E_1 E_2 ... E_N) [?]

Similar to *CATCH, but if a *THROW is executed whose tag is not in the tag list, an Unseen Throw Tag error is signalled instead of searching further down the stack for another catcher.

(UNWIND-PROTECT E U_1 U_2 ... U_N) [?]

Evaluates the form E, then the forms U_1 through U_N. If, during the evaluation of E, an event occurs that causes the stack to be unwound (e.g. a non-local exit, an error, a QUIT, etc.), the unwinding will pause at the point of the UNWIND-PROTECT and the U_i will be evaluated. NOINTERRUPT is set to T before the U_i are evaluated, so asynchronous conditions can't interefere with the cleanup routines.

(CATCH X ["TAG"]) [2-44]

Older form of *CATCH, being phased out. Evaluates X, catching all THROWs with a matching tag. If the tag is omitted, all THROWs are caught.

(THROW X ["TAG"]) [2-45]

Older form of *THROW, being phased out. Forces a non-local exit, returning X as value, to a CATCH with matching tag or no tag at all. If the second argument is omitted, THROW returns to the nearest enclosing CATCH.

5.7. Error Signalling

(ERROR [MSG] [DATUM] [UINT-CHN]) [2-46]

(ERROR) is the same as (ERR). (ERROR MSG) signals a simple error and prints the error message. (ERROR MSG DATUM) signals an error with an object to be printed and an error message. (ERROR MSG DATUM UINT-CHN) signals an error but first signals a user interrupt on the specified channel. The value returned by the user interrupt handler determines how the error will be treated. All arguments to ERROR are evaluated.

(ERRSET FORM [FLAG]) [2-46]

Evaluates FORM and returns its value in a list. If FORM signals an error, the error is trapped and ERRSET returns NIL. If FLAG is specified and is NIL, the error message is suppressed as well.

(ERR [FORM] [FLAG]) [2-47]

(ERR) causes a regular LISP error with no message and no user interrupt. (ERR X) causes the surrounding ERRSET to return X, or signals an error if there is no ERRSET. (ERR X T) is like (ERR X), except that X is not evaluated until just before the enclosing ERRSET returns, i.e. after the pdl has been unwound.

6. Mapping Functions

(MAPC FN L) [2-99]

Applies function FN to successive elements of the list L. Returns L.

(MAPCAR FN L) [2-99]

Applies function FN to successive elements of the list L. Returns a list of the results.

(MAPCAN FN L) [2-99]

Applies function FN to successive elements of the list L. Returns NCONC of the results.

(MAP FN L) [2-99]

Applies function FN to successive subslists of the list L. Returns L.

(MAPLIST FN L) [2-99]

Applies function FN to successive sublists of the list L. Returns a list of the results.

(MAPCON FN L) [2-99]

Applies function FN to successive sublists of the list L. Returns NCONC of the results.

(MAPATOMS FN X) [2-99]

(MAPATOMS FN) applies function FN to all atoms in the current obarray. (MAPATOMS FN X) applies FN to all atoms in obarray X.

7. Arrays

ARRAY BOUNDS [2-90]

The bounds of an array, denoted in this section as B_i, give the number of distinct subscript values for each dimension Arrays in MacLisp are zero-based. Therefore, the maximum subscript along any dimension is one less than the bound.

(ARRAY "X" "Y" B_1 ... B_N) [2-92]

Creates an N-dimensional array named X of type Y with bounds B_1 through B_N. Only the B_i are evaluated. The type code Y may be T for an ordinary array, FIXNUM or FLONUM for numeric arrays, NIL for un-garbage-collected arrays, or OBARRAY or READTABLE. Returns X. X may be NIL, in which case an anonymous array is created and an array pointer is returned.

*(*ARRAY X Y B_1 ... B_N)* [2-92]

Like ARRAY, but all arguments are evaluated.

*(*REARRAY *X TYPE B_1 ... B_N)* [2-93]

Redefines array X, copying the contents of the old array into the new array in row-major order. (*REARRAY X) kills array pointer X.

(STORE ARRAY-REF VALUE) [2-93]

Stores VALUE in the specifed array element. ARRAY-REF must be a subscripted array reference, or an arraycall.

(ARRAYDIMS X) [2-93]

Returns a list of the type and bounds of array X. X must be an array pointer or an atomic symbol with an ARRAY property.

*(FILLARRAY *X Y)* [2-94]

Fills array X from object Y, which may be a list or another array. Extra elements are ignored. If there are too few elements to fill X, the remaining elements of X are unchanged.

(LISTARRAY A [N]) [2-94]

Creates a list of the first N elements of array A, starting with the zeroth element. (LISTARRAY A) creates a list of all elements of A. A may be either an array pointer or an atomic symbol with an array property.

(DUMPARRAYS ARRAYLIST FILESPEC) [2-95]

ARRAYLIST is a list of array names. Dumps specified fixnum or flonum arrays to file named by filespec.

(LOADARRAYS FILESPEC) [2-95

Reloads the arrays in the file named by filespec. Returns a list of triplets of form (NEWNAME OLDNAME SIZE), where NEWNAME is a new gensym'ed atom, OLDNAME is the name the array had when it was dumped, and SIZE is the number of elements in the array.

8. Input/Output

In this section, the term FILESPEC refers to a name for a file. A FILESPEC may be a list, a string, or the name part of a file object. If a list, it may be in NEWIO format: ((dev dir) name ext), or OLDIO format: (name ext dev dir).

A file object is a special kind of array that contains information about an open or closed file. When open, the file object is the channel through which i/o operations are directed to the file. In this section, when the symbol FILE appears in an argument specification it indicates a file object. Some functions (e.g. PRINT) can take a list of file objects instead. Also, most i/o functions will do i/o to the terminal if the FILE argument is omitted. Passing T instead of a file object tells MacLisp to use the terminal.

8.1. Functions On Files

(OPEN FILESPEC [MODELIST]) [?]

Opens the file in the specified mode and returns a file object. Available mode options are IN, OUT, APPEND, ASCII, FIXNUM, IMAGE, DSK, TTY, BLOCK, and SINGLE. The default mode is (IN ASCII DSK BLOCK).

(CLOSE FILE) [?]

Closes the specified file. FILE must be a file object, as returned by OPEN.

(PROBEF FILESPEC) [?]

Tests for the existence of the specified file. Returns a completed filespec if found, else NIL.

(DELETEF FILESPEC) [?]

Deletes the specified file. Returns the completed filespec if successful.

(RENAMEF FROMFILESPEC TOFILESPEC) [?]

Renames a file. Returns the completed filespec if successful.

(LENGTHF FILE) [?]

Given a file object, returns the length of the file in words or bytes, depending on how the file was opened.

(FASLP FILESPEC) [?]

Returns T if the specified file is a FASL (compiled LISP) file, else NIL.

8.2. Functions on Filespecs and File Objects

(NAMELIST FILESPEC) [?]

Converts the given filespec to list form.

(NAMESTRING FILESPEC) [?]

Converts the given filespec to string form.

(SHORTNAMESTRING FILESPEC) [?]

Returns the file name portion of the given filespec, in string form. (Omits the device and directory.)

(TRUENAME FILE) [?]

Returns a filespec for the actual name of the file associated with the given file object.

(DEFAULTF FILESPEC) [?]

Sets the DEFAULTF variable from the given filespec.

DEFAULTF [?]

Global variable containing defaults for each component of a filespec. Used by various i/o functions to complete partially specified filespec arguments.

(MERGEF FILESPEC$_1$ FILESPEC$_2$) [?]

Merges two filespecs and returns the result. An asterisk is used as the wildcard character.

*(CNAMEF *FILE FILESPEC)* [?]

Changes the name of the closed file object FILE to that given in FILESPEC. Used to avoid creating extra file arrays. Obscure.

(FILEP FILE) [?]

Returns T if its argument is a file object, otherwise NIL.

8.3. Basic I/O

(READ [FILE] [EOFVAL]) [?]

Reads one s-expression from the specified file. Returns EOFVAL if end of file is encountered.

*(*READ)* [?]

Like READ with no arguments. Compiles faster.

(PRIN1 X [FILE]) [?]

Prints s-expression X on the specified file. Special characters are slashified. Eg: (PRINC '|FOO BAR|) prints |FOO BAR|.

(PRINT X [FILE]) [?]

Like PRIN1, but does a TERPRI first and prints a space afterwards.

(PRINC X [FILE]) [?]

Like PRIN1, but does not slashify special characters. Eg: (PRINC '|FOO BAR|) prints FOO BAR.

(TERPRI [FILE]) [?]

Writes a carriage return to the specified file.

BASE [?]

The global variable BASE controls the output radix for displaying numbers. In a bare MacLisp, BASE defaults to 8. With a CMU LISP.INI file, it is set to 10. Setting BASE to ROMAN causes numbers to be output as roman numerals

IBASE [?]

The global variable IBASE controls the input radix for reading numbers. In a bare MacLisp, IBASE defaults to 8. With a CMU LISP.INI file, it is set to 10. Setting IBASE to ROMAN causes numbers to be input in roman numeral form.

**NOPOINT* [?]

If the global variable *NOPOINT is NIL, numbers will be written with decimal points when BASE is set to 10. If non-NIL, decimal points will be omitted.

*(*NOPOINT FILE)* [?]

Inhibits printing of decimal points when outputting to the specified file.

8.4. Character I/O

(READCH [FILE] [EOFVAL]) [?]

Reads one character from the specified file, and returns a character object. EOFVAL is returned if end of file is encountered.

*(*READCH)* [?]

Like READCH with no arguments. Compiles faster.

(TYI [FILE] [EOFVAL]) [?]

Like READCH, but returns a fixnum instead of a character object.

TYI [?]

Global variable containing the tty input file object.

*(*TYI)* [?]

Like TYI with no arguments. Compiles faster.

(TYIPEEK [PEEKMODE] [FILE] [EOFVAL]) [?]

Returns the fixnum representation of the next character in the input buffer of FILE, without removing the character. PEEKMODE defaults to NIL.

(READLINE [FILE]) [?]

Reads a line of text, delimited by a carriage return, and returns it as a symbol.

(TYO N [FILE]) [?].

Writes the ASCII character denoted by fixnum N to the specified file.

TYO [?]

Global variable containing the tty output file object.

(+TYO N [FILE]) [?]

Super-fast TYO. Does not check line length. FILE must be a single file object, not T or a list.

8.5. General I/O Control

(LINEL FILE [N]) [?]

With one argument, returns the line length associated with the file object. With two arguments, sets the line length.

(PAGEL FILE [N]) [?]

With one argument, returns the page length associated with the file object. With two arguments, sets the page length.

(LINENUM FILE [N]) [?]

With one argument, returns the current line number as stored in the file object. With two arguments, sets the line number.

(PAGENUM FILE [N]) [?]

With one argument, returns the current page number as stored in the file object. With two arguments, sets the page number.

(CHARPOS FILE [N]) [?]

With one argument, returns the current character position as stored in the file object. With two arguments, sets the character position.

(EOFFN FILE [FN]) [?]

With one argument, returns the end-of-file function associated with the specified file object. With two arguments, sets FN to be the function called when end-of file is encountered on the file object. If FILE is NIL, sets the default end-of-file function. If FN is NIL, clears the eof function.

(ENDPAGEFN FILE [FN]) [?]

Like EOFFN, but the function is called on every end-of-page interrupt, i.e. whenever the line count exceeds the page length. Useful for doing **MORE** mode processing.

(CLEAR-INPUT FILE) [?]

Clears the input buffer associated with FILE.

(CLEAR-OUTPUT FILE) [?]

Clears the output buffer associated with FILE.

(FORCE-OUTPUT FILE) [?]

Forces the output buffer of the specified file object to be written.

8.6. Terminal I/O

INFILE [?]

Global variable containing the current console input file object. Usually T. Console input will be done through INFILE only when the global variable ^Q is non-NIL.

^Q [?]

Console input switch. If the global variable ^Q (two characters) is non-NIL, input is from the source selected by the global variable INFILE, otherwise input is from the TYI file object. In the reader control-Q is a macro character which sets the variable ^Q to T.

(INPUSH FILE) [?]

Pushes the current value of INFILE onto the input stack, and makes FILE be the new value of INFILE. (INPUSH -1) pops the input stack.

INSTACK [?]

A global variable containing the current input stack, as maintained by INPUSH.

OUTFILES [?]

A list of console output file objects. Usually NIL. Console output will go to the specified files, in addition to the TYO file object, only when the global variable ^R is non-NIL.

^R [?]

Console output switch. If the global variable ^R (two characters) is non-NIL, console output is directed to the files specified in the global variable OUTFILES, as well as to the terminal. In the reader control-R is a macro character which sets the variable ^R to T.

^W [?]

Terminal output switch. If the global variable ^W (two characters) is non-NIL, terminal output is suppressed. May be used in conjunction with OUTFILES and ^R to redirect output to a file instead of the terminal. In the reader, control-W is a macro character which sets the variable ^W to T.

MSGFILES [?]

A global variable similar to OUTFILES, but used fo system-type messages, i.e. those generated by ERRORs, BREAKs, and system packages, as well as user-generated console output. Defaults to (T). Not controlled by the ^R switch.

ECHOFILES [?]

Global variable containing a list of file objects for echoing terminal input to. Usually NIL. Useful in dribble packages that record a LISP session.

(LISTEN [FILE]) [?]

Returns 1 if there are characters in the tty input buffer of FILE, else 0.

8.7. Binary and Random Access I/O

(IN FILE) [?]

Reads one word from FILE and returns it as a fixnum. The file must have been opened in FIXNUM mode.

(OUT FILE X) [?]

Writes one word to a file. The file must have been opened in FIXNUM mode.

(FILEPOS FILE [N]) [?]

With one argument, returns the current position in the file (characters or words.) With two arguments, sets the current position to N. The file may be opened in ASCII, FIXNUM, or IMAGE mode. An error will be signalled if N is greater than the length of the file. A position of NIL means "beginning of file", and T means "end of file".

8.8. Miscellaneous Functions

(RUBOUT CHAR [FILE]) [?]

Rubs one character out of FILE's input buffer. Returns T if the rubout was successful, else NIL. Useful for writing your own tty scanner.

(ERRPRINT P [FILE]) [?]

Reprints the nearest error down the stack from P, which must be a pdl pointer. If P is NIL, the latest error is printed.

(FASLOAD "(DEV DIR)" "FILE" "EXT") [?]

Loads a compiled LISP file, called a fasl file. The extension defaults to FAS. FASLOAD also accepts file names in OLDIO format. All the arguments are optional; MacLisp tries to figure out the filespec and uses DEFAULTF to complete unspecified fields.

8.9. OLDIO Functions

These are functions left over from the old MacLisp i/o system. They are retained for compatibility with existing code. All the arguments are optional; MacLisp tries to figure out the filespec and uses DEFAULTF to complete unspecified fields. OLDIO functions also accept filespecs in the NEWIO format, e.g. (dev dir) name ext.

(UREAD "NAME" "EXT" "DEV" "DIR") [?]

Opens the specified file and pushes it onto the input stack. The ^Q switch must be turned on before the file will actually be read.

UREAD [?]

Global variable containing the file object for the file currently opened by UREAD.

(UCLOSE) [?]

Closes the current input file opened by UREAD.

(UWRITE "DEV" "DIR") [?]

Opens a file for output on the specified device and directory, pushing the file object onto OUTFILES. The ^R switch must be turned on before output will actually be directed to the file.

UWRITE [?]

Global variable containing the file object for the file currently opened by UWRITE.

(UFILE "NAME" "EXT") [?]

Closes the current output file opened by UWRITE and renames it to the specified file name.

(UAPPEND "NAME" "EXT" "DEV" "DIR") [?]

Opens the specified file for output in APPEND mode, pushing the file object onto OUTFILES. The ^R switch must be turned on before output will actually be directed to the file.

(UKILL "NAME" "EXT" "DEV" "DIR") [?]

OLDIO equivalent of DELETEF. Deletes the specified file.

(UPROBE "NAME" "EXT" "DEV" "DIR") [?]

OLDIO equivalent of PROBEF. Returns T if the specified file exists, else NIL.

(CRUNIT "DEV" "DIR") [?]

With no arguments, returns the current device and directory. OLDIO functions update this by setting DEFAULTF. With arguments, sets the current device and directory.

9. Programming Tools

9.1. Common Functions

(DEFUN NAME TYPE ARGS BODY ...) [?]

Special form for defining a function. TYPE should be one of EXPR, FEXPR, or MACRO; it defaults to EXPR if omitted. ARGS is the argument list. It is followed by one or more s-expressions that make up the function body. Eg: (DEFUN KWOTE FEXPR (X) (CAR X))

(GRINDEF "FN_1" "FN_2" ... "FN_N") [?].

Pretty-prints the definitions of the specified functions.

(EDITF "FN") [?]

Invokes the editor on the named function. See the editor section of the CMU TOPS Lisp manual for details.

(TRACE FN_1 FN_2 ... FN_N) [3-35]

Special form; traces the named functions. See the MacLisp Reference Manual for information about fancy trace options.

(UNTRACE FN_1 FN_2 ... FN_N) [3-38]

Untraces the named functions. If called with no arguments, untraces all traced functions. See the MacLisp Reference Manual for more details.

(STEP) [3-40]

The MacLisp single-stepper. See the MacLisp Reference Manual for instructions.

(DEBUG) [?]

The CMU MacLisp debugger. See the file FIXIT.DOC[C380ML5P]/A for details.

9.2. Packages

XPRINT [?]

The Waters printer. Contains the prettyprinter and many other print functions.

LET [?]

The LET package contains two useful prog forms, LET and LET*. It also contains a destructuring assignment function called DESETQ. See ARCHIV.DOC[C380ML5P]/A for details.

DEFVST [?]

The MacLisp structure package. Used to define and access hairy record structures.

DEFMAC [?]

An extension to DEFUN's syntax that provides more flexible argument definitions. Also, some functions for defining macros conveniently. See ARCHIV.DOC[C380ML5P]/A for details.

FORMAT [?]

The FORMAT package provides functions for formatting numbers and atoms into more complex messages. .

10. Storage Management

10.1. Garbage Collection

(GC) [3-59]

Causes a garbage collection to take place. Returns NIL.

(GCTWA ["FLAG"]) [3-59]

Controls the Garbage Collection of Truly Worthless Atoms. (GCTWA) causes truly worthless atoms to be removed on the next garbage collection. (GCTWA T) causes truly worthless atoms to be removed on all subsequent garbage collections. (GCTWA NIL) turns off removal of truly worthless atoms for all garbage collections after the next one. The value returned is a fixnum indicating the current GCTWA status.

^D [3-60]

The global variable ^D (two characters) controls the printing of messages after garbage collections. If non-NIL, messages will be printed whenever a space is expanded or garbage collected. In the reader control-D is a macro character which sets the variable ^D to T.

10.2. Storage Allocation Concepts

GCMAX [3-62]

The maximum size to which a space be should be allowed to grow. If the space exceeds this size, an error is signalled.

GCSIZE [3-62]

The expected size of the space. Garbage collections will be performed to keep the space within this size. If garbage collection fails to free enough storage, the space will be expanded as long as it does not exceed GCMAX.

GCMIN [3-62]

The minimum amount of free space that should be left after a garbage collection. It may be either a fixnum, indicating the size in words, or a flonum, indicating a percentage.

PDLSIZE [3-62]

The number of words of valid data in a pdl at the moment.

PDLMAX [3-62]

The maximum size to which a pdl may grow before intervention is required. Used to detect infinite recursion.

PDLROOM [3-62]

The size beyond which a pdl may not grow no matter what. This is slightly larger than the pdlmax, so that there will be some room left in which an error handling routine can run.

10.3. Storage Spaces

LIST [3-60

Cons cells.

FIXNUM [3-60

36-bit integers.

FLONUM [3-60

36-bit floating point numbers.

BIGNUM [3-60

Bignum headers. Bignums also occupy fixnum and list space.

SYMBOL [3-61

Atomic symbols.

HUNKn [3-61

Hunk space of size n, which must be a power of 2. Thus there exists HUNK2 space, HUNK4 space, HUNK8 space, etc.

ARRAY [3-61

Special array cells.

REGPDL [3-61

The regular pushdown list, used for passing arguments and doing recursion.

SPECPDL [3-61

The special pushdown list, used for binding.

FXPDL [3-61

The fixnum pushdown list, used for temporary numeric values.

FLPDL [3-61

The flonum pushdown list, used for temporary numeric values.

BPS [3-61]

Binary program space. Used for compiled LISP code, and also arrays. Must be allocated at initialization time.

10.4. Allocation

(ALLOC SPACELIST) [3-63]

Sets storage management parameters for various spaces. The argument should be a list of form (S_1 L_1 S_2 L_2 ...), where the S_i are space names and the L_i are fixnums or 3-lists. A fixnum specifies the pdlmax for a pdl, or gcsize and gcmax for other spaces. A 3-list is interpreted as (gcsize gcmax gcmin). NIL in any position means "don't change that paramter". (ALLOC T) returns a list of space names and their current parameters.

ALLOCATION PSEUDOCOMMENT [?]

Binary program space can't be expanded once MacLisp starts up. Thus it must be allocated in the LISP.INI file. This is done with a COMMENT that must appear as the first expression in the file. The COMMENT should contain a series of space names followed by initial allocations, e.g. (COMMENT BPS 10000 SYMBOL 5000).

11. Status Functions

(STATUS FUNCTION ARG_1 ARG_2 ... ARG_N) [3-77]

Special form for interrogating various system parameters. The arguments depend on the particular status function being executed.

(SSTATUS FUNCTION ARG_1 ARG_2 ... ARG_N) [3-77]

Special form for setting various system parameters. The arguments depend on the particular sstatus function being executed.

11.1. Environment Enqueries

DATE [3-89]

(STATUS DATE) returns the date as a 3-list of fixnums, representing the date as (yy mm dd).

DOW [3-89]

(STATUS DOW) returns the day of the week as an atomic symbol.

DAYTIME [3-89]

(STATUS DAYTIME) returns the time of day as a 3-list of fixnums, representing the time as (hh mm ss).

LISPVERSION [3-89]

(STATUS LISPVERSION) returns the version number of this MacLisp as an atomic symbol.

UDIR [3-90]

(STATUS UDIR) returns the name of the file directory the job is connected to, usually the user's own.

UNAME [3-90]

(STATUS UNAME) returns the user's ppn, e.g. C410HB00.

USERID [3-90]

(STATUS USERID) returns the user's name, e.g. BOVIK.

JNAME [3-90]

(STATUS JNAME) returns a job identifier of form nnnLSP, where nnn is a TOPS-10 job number.

SEGLOG [3-90]

(STATUS SEGLOG) returns the log base 2 of a segment, i.e. one unit of space allocation. On TOPS-10 systems this is one page (512 words), so the status call returns 9.

FEATURES [3-96]

(STATUS FEATURES) returns a list of symbols indicating features of the current LISP system.

FEATURE [3-98]

(STATUS FEATURE X) returns T if the atom X is in the features list, else nil. (SSTATUS FEATURE X) adds X to the feature list. X is not evaluated.

NOFEATURE [3-98]

(SSTATUS NOFEATURE X) removes X from the feature list. (STATUS NOFEATURE X) is equivalent to (NOT (STATUS FEATURE X)). X is not evaluated.

STATUS [3-98]

(STATUS STATUS) returns a list of valid status functions. (STATUS STATUS X) returns T if X is a valid status function, else NIL. X is not evaluated.

SSTATUS [3-98]

(STATUS SSTATUS) returns a list of valid sstatus functions. (STATUS SSTATUS X) returns T if X is a valid sstatus function, else NIL. X is not evaluated.

11.2. Garbage Collector Status

GCTIME [3-87]

(STATUS GCTIME) returns the number of microseconds spent garbage collecting. (SSTATUS GCTIME N) resets the time counter to N, and returns the previous value of the counter.

SPCNAMES [3-87]

(STATUS SPCNAMES) returns a list of space names, which may be used with ALLOC or with STATUS calls described below.

SPCSIZE [3-88]

(STATUS SPCSIZE SPACE) returns the actual size of SPACE in words. SPACE is evaluated.

GCMAX [3-88]

(STATUS GCMAX SPACE) returns the gcmax parameter for SPACE. (SSTATUS GCMAX SPACE N) sets the gcmax parameter to N. SPACE and N are evaluated.

GCMIN [3-88]

(STATUS GCMIN SPACE) returns the gcmin parameter for SPACE. (SSTATUS GCMIN SPACE N) sets the gcmin parameter to N. SPACE and N are evaluated.

GCSIZE [3-88]

(STATUS GCSIZE SPACE) returns the gcsize parameter for SPACE. (SSTATUS GCSIZE SPACE N) sets the gcsize parameter to N. SPACE and N are evaluated.

PURSPCNAMES [3-88]

(STATUS PURSPCNAMES) returns a list of spaces that have pure versions.

PURSIZE [3-88]

(STATUS PURSIZE SPACE) returns the actual size of the pure version of SPACE. SPACE is evaluated.

PDLNAMES [3-88]

(STATUS PDLNAMES) returns a list of all the pdls used by this LISP. Those names may be used in the STATUS calls described below.

PDLSIZE [3-88]

(STATUS PDLSIZE PDL) returns the current number of words on the pdl. PDL is evaluated.

PDLMAX [3-88]

(STATUS PDLRMAX PDL) returns the pdlmax parameter of the pdl. PDL is evaluated.

PDLROOM [3-88]

(STATUS PDLROOM PDL) returns the maximum size of the pdl. PDL is evaluated.

MEMFREE [3-89]

(STATUS MEMFREE) returns the number of words of address space not yet allocated for any purpose.

11.3. I/O Status

FILEMODE [3-80]

(STATUS FILEMODE FILE) returns a list of form (MODELIST . FEATURELIST), where MODELIST is a description of the mode in which the file is opened and FEATURELIST is a (possibly null) list of features from the set CURSORPOS, FILEPOS, RUBOUT, and SAIL.

TABSIZE [3-77]

(STATUS TABSIZE) returns the number of character positions assumed between tab stops. For TOPS-10 systems, the number is 8.

NEWLINE [3-77]

(STATUS NEWLINE) returns a fixnum which is the ASCII code for the system's end-of-line character. For TOPS-10 systems, this number is 15 octal, i.e. carriage return.

LINMODE [3-78]

(STATUS LINMODE) returns T if the terminal is in line-at-a-time input mode, or NIL if it is in character-at-a-time input mode. (SSTATUS LINMODE X) sets the linmode to X. This status/sstatus call may take a file object as an additional argument.

TTYINT [3-78]

(SSTATUS TTYINT CHAR FUNC FILE) turns on a tty interrupt character. See the MacLisp Reference manual for details.

TTYSCAN [3-81]

(SSTATUS TTYSCAN FUNC FILE) sets up a function to perform initial processing of terminal input. See ARCHIV.DOC[C380ML5P]/A and the MacLisp Reference Manual for details.

TTYCONS [3-79]

(SSTATUS TTYCONS TTY_1 TTY_2) binds two tty files into a console. See the MacLisp reference manual for details.

11.4. Time

(RUNTIME) [3-99]

Returns the amount of cpu time used by the job, in microseconds, since the last call to RUNTIME.

(TIME) [3-99]

Returns the time (in seconds) the system has been up, as a flonum.

Index

)		
*	[2-72]	14
*$	[2-75]	15
*ARRAY	[2-92]	25
*CATCH	[?]	22
*DIF	[2-70]	14
*FUNCTION	[2-9]	18
*NOPOINT	[?]	29
*QUO	[2-71]	14
*READ	[?]	28
*READCH	[?]	29
*REARRAY	[2-93]	25
*THROW	[?]	22
*TYI	[?]	29
+	[2-72]	14
+$	[2-75]	15
+TYO	[?]	30
-	[2-72]	14
-$	[2-75]	15
//	[2-73]	14
//$	[2-76]	15
1+	[2-73]	14
1+$	[2-76]	15
1-	[2-73]	14
1-$	[2-76]	15
<	[2-66]	12
=	[2-65]	11
>	[2-65]	11
ABS	[2-67]	13
ADD1	[2-70]	13
ALLOC	[3-63]	39
ALLOCATION PSEUDOCOMMENT	[?]	39
ALPHALESSP	[2-56]	10
AND	[2-36]	21
APPEND	[2-19]	2
APPLY	[2-7]	17
ARG	[2-12]	21
ARRAY	[2-92]	24
ARRAY	[3-61]	38
ARRAY BOUNDS	[2-90]	24
ARRAYCALL	[2-13]	18
ARRAYDIMS	[2-93]	25
ASCII	[2-83]	9
ASSOC	[2-27]	5
ASSQ	[2-28]	5
ATAN	[2-78]	16
ATOM	[2-1]	6
BACKQUOTE	[?]	18

BASE	[?]	28
BIGNUM	[3-60]	38
BIGP	[2-1]	7
BOOLE	[2-80]	16
BOUNDP	[2-51]	8
BPS	[3-61]	39
C....R	[2-16]	3
CAR	[2-15]	3
CASEQ	[?]	21
CATCH	[2-44]	23
CATCH-BARRIER	[?]	22
CATCHALL	[?]	22
CDR	[2-15]	3
CHARPOS	[?]	30
CLEAR-INPUT	[?]	31
CLEAR-OUTPUT	[?]	31
CLOSE	[?]	26
CNAMEF	[?]	27
COMMENT	[?]	19
COND	[2-36]	21
CONS	[2-16]	2
COPYSMBOL	[2-59]	10
COS	[2-78]	16
CRUNIT	[?]	34
CXR	[2-33]	6
DATE	[3-89]	39
DAYTIME	[3-89]	40
DEBUG	[?]	35
DECLARE	[?]	19
DEFAULTF	[?]	27
DEFMAC	[?]	36
DEFPROP	[2-54]	8
DEFUN	[?]	35
DEFVST	[?]	36
DELETE	[2-25]	4
DELETEF	[?]	26
DELQ	[2-26]	4
DIFFERENCE	[2-69]	13
DO	[2-40]	20
DO	[2-41]	20
DOW	[3-89]	40
DUMPARRAYS	[2-95]	25
ECHOFILES	[?]	32
EDITF	[?]	35
ENDPAGEFN	[?]	31
EOFFN	[?]	30
EQ	[2-3]	3
EQUAL	[2-3]	4
ERR	[2-47]	23
ERROR	[2-46]	23
ERRPRINT	[?]	33
ERRSET	[2-46]	23
EVAL	[2-7]	17
EXP	[2-77]	15
EXPLODE	[2-85]	9
EXPLODEC	[2-85]	9
EXPLODEN	[2-85]	9
EXPT	[2-70]	14

FASLOAD	[?]	33
FASLP	[?]	27
FEATURE	[3-98]	40
FEATURES	[3-96]	40
FILEMODE	[3-80]	42
FILEP	[?]	28
FILEPOS	[?]	33
FILLARRAY	[2-94]	25
FIX	[2-67]	12
FIXNUM	[3-60]	38
FIXP	[2-1]	7
FLATC	[2-85]	10
FLATSIZE	[2-85]	10
FLOAT	[2-67]	12
FLOATP	[2-1]	7
FLONUM	[3-60]	38
FLPDL	[3-61]	38
FORCE-OUTPUT	[?]	31
FORMAT	[?]	36
FSC	[2-82]	17
FUNCALL	[2-13]	17
FUNCTION	[2-8]	18
FXPDL	[3-61]	38
GC	[3-59]	36
GCD	[2-70]	13
GCMAX	[3-62]	37
GCMAX	[3-88]	41
GCMIN	[3-62]	37
GCMIN	[3-88]	41
GCSIZE	[3-62]	37
GCSIZE	[3-88]	42
GCTIME	[3-87]	41
GCTWA	[3-59]	36
GENSYM	[2-59]	11
GET	[2-53]	8
GETCHAR	[2-83]	9
GETCHARN	[2-83]	9
GETL	[2-53]	8
GO	[2-42]	20
GREATERP	[2-65]	12
GRINDEF	[?]	35
HAIPART	[2-68]	13
HAULONG	[2-64]	12
HUNK	[2-32]	6
HUNKn	[3-61]	38
HUNKP	[2-33]	6
HUNKP	[2-2]	7
HUNKSIZE	[2-33]	6
IBASE	[?]	28
IFIX	[2-67]	12
IMPLODE	[2-84]	10
IN	[?]	32
INFILE	[?]	31
INPUSH	[?]	31
INSTACK	[?]	31
INTERN	[2-58]	10

JNAME	[3-90]	40
LABEL	[1-17]	19
LAMBDA	[1-15]	19
LAST	[2-18]	3
LENGTH	[2-18]	3
LENGTHF	[?]	27
LESSP	[2-65]	12
LET	[?]	36
LINEL	[?]	30
LINENUM	[?]	30
LINMODE	[3-78]	43
LISPVERSION	[3-89]	40
LIST	[2-19]	2
LIST	[3-60]	38
LIST*	[?]	2
LISTARRAY	[2-94]	25
LISTEN	[?]	32
LISTIFY	[2-13]	22
LOADARRAYS	[2-95]	25
LOG	[2-77]	15
LSH	[2-81]	16
LSUBRCALL	[2-13]	18
MAKHUNK	[2-33]	6
MAKNAM	[2-84]	10
MAKNUM	[2-29]	5
MAKUNBOUND	[2-51]	8
MAP	[2-99]	24
MAPATOMS	[2-99]	24
MAPC	[2-99]	23
MAPCAN	[2-99]	24
MAPCAR	[2-99]	24
MAPCON	[2-99]	24
MAPLIST	[2-99]	24
MAX	[2-66]	12
MEMBER	[2-24]	4
MEMFREE	[3-89]	42
MEMQ	[2-25]	4
MERGEF	[?]	27
MIN	[2-66]	12
MINUS	[2-68]	13
MINUSP	[2-63]	11
MSGFILES	[?]	32
MUNKAM	[2-29]	5
NAMELIST	[?]	27
NAMESTRING	[?]	27
NCONC	[2-20]	2
NCONS	[2-16]	2
NEWLINE	[3-77]	43
NOFEATURE	[3-98]	41
NOT	[2-4]	4
NRECONC	[2-21]	2
NREVERSE	[2-21]	2
NTH	[?]	3
NTHCDR	[?]	3
NULL	[2-4]	4
NUMBERP	[2-2]	7
ODDP	[2-63]	11

OPEN	[?]	26
OR	[2-26]	21
OUT	[?]	32
OUTFILES	[?]	31
PAGEL	[?]	30
PAGENUM	[?]	30
PDLMAX	[3-62]	37
PDLMAX	[3-88]	42
PDLNAMES	[3-88]	42
PDLROOM	[3-62]	37
PDLROOM	[3-88]	42
PDLSIZE	[3-62]	37
PDLSIZE	[3-88]	42
PLIST	[2-55]	8
PLUS	[2-69]	13
PLUSP	[2-63]	11
PNGET	[2-57]	9
PNPUT	[2-57]	9
POP	[?]	8
PRIN1	[?]	28
PRINC	[?]	28
PRINT	[?]	28
PROBEF	[?]	26
PROG	[2-38]	20
PROG2	[2-10]	19
PROGN	[2-11]	19
PROGV	[2-11]	20
PURSIZE	[3-88]	42
PURSPCNAMES	[3-88]	42
PUSH	[?]	7
PUTPROP	[2-54]	8
QUOTE	[2-7]	18
QUOTIENT	[2-69]	13
RANDOM	[2-79]	17
READ	[?]	28
READCH	[?]	29
READLINE	[?]	29
READLIST	[2-84]	10
REGPDL	[3-61]	38
REMAINDER	[2-70]	13
REMOB	[2-59]	10
REMPROP	[2-55]	8
RENAMEF	[?]	26
RETURN	[2-43]	20
REVERSE	[2-20]	2
ROT	[2-81]	16
RPLACA	[2-22]	3
RPLACD	[2-22]	3
RPLACX	[2-33]	6
RUBOUT	[?]	33
RUNTIME	[3-99]	43
SAMEPNAMEP	[2-56]	10
SASSOC	[2-28]	5
SASSQ	[2-29]	5
SEGLOG	[3-90]	40
SET	[2-50]	7
SETARG	[2-12]	21

SETPLIST	[2-55]	9
SETQ	[2-49]	7
SHORTNAMESTRING	[?]	27
SIGNP	[2-63]	11
SIN	[2-78]	16
SORT	[2-30]	5
SORTCAR	[2-31]	6
SPCNAMES	[3-87]	41
SPCSIZE	[3-88]	41
SPECPDL	[3-61]	38
SQRT	[2-77]	16
SSTATUS	[3-77]	39
SSTATUS	[3-98]	41
STATUS	[3-77]	39
STATUS	[3-98]	41
STEP	[3-40]	35
STORE	[2-93]	25
SUB1	[2-70]	13
SUBLIS	[2-23]	4
SUBRCALL	[2-13]	18
SUBST	[2-22]	4
SXHASH	[2-26]	5
SYMBOL	[3-61]	38
SYMBOLP	[2-1]	6
SYMEVAL	[2-50]	8
TABSIZE	[3-77]	43
TERPRI	[?]	28
THROW	[2-45]	23
TIME	[3-99]	43
TIMES	[2-69]	13
TRACE	[3-35]	35
TRUENAME	[?]	27
TTYCONS	[3-79]	43
TTYINT	[3-78]	43
TTYSCAN	[3-81]	43
TYI	[?]	29
TYIPEEK	[?]	29
TYO	[?]	29, 30
TYPEP	[2-2]	7
UAPPEND	[?]	34
UCLOSE	[?]	34
UDIR	[3-90]	40
UFILE	[?]	34
UKILL	[?]	34
UNAME	[3-90]	40
UNTRACE	[3-38]	35
UNWIND-PROTECT	[?]	22
UPROBE	[?]	34
UREAD	[?]	33, 34
USERID	[3-90]	40
UWRITE	[?]	34
XCONS	[2-17]	2
XPRINT	[?]	36
ZEROP	[2-63]	11
ZUNDERFLOW	[2-79]	17
\	[2-73]	14