L110 Programmer’s Manual

Forrest William Howard, Jr.
HRSTS
Science Center
Harvard University

September 28,1975

Using L110

After you have logged into the HRSTS system , respond to the prompting "& " by:

```
& l110
```

L110 will respond with a message, and prompt you for input with a "->", indicating that you are in eval mode. Commands may now be typed to L110.

If one wishes to use the EVALQUOTE feature, then one should respond to the prompt with:

```
->($MUMBLE T)
```

L110 will then prompt you with "<-" indicating that you are in EVALQUOTE mode. The rules for EVALQUOTE are :

1.) A command has the form:
```
        F1 (F2 F3 F4 ... Fn)
```
where F1, ..., Fn are arbitrary forms.

2.) The form:
```
        (F1 (QUOTE2)(QUOTE F3)...(QUOTE Fn))
```
is then evaled and the result is printed on the terminal.

Example:

```
<-CAR((A B))
A
<-CDR((A B C))
(B C)
```

There are some special characters in L110, namely [,], and "'". Single quote serves the function of QUOTEing the s-expression, atom, or number following, for example:

```
'A=(QUOTE A)
'(A B C)=(QUOTE (A B C))
```

Right bracket serves the function of providing as many right parentheses as necessary to close back to either the last left bracket or to the beginning of the current s-expression. For example:

```
(A B C]=(A B C)
(A (B (C ]=(A (B (C)))
(A [B (C])=(A (B (C)))
()=[]=[)=(]=NIL
```

The standard editing commands on the HRSTS system work with L110, i.e. CTRL-U erases the current line, RUBOUT deletes the last character up to the begining of the line, etc.

Data Types

There are 5 data types in L110:

Type	*Description*
Dotted Pair (DTPR)	List element with CAR and CDR.
Integer (INT)	Number with range + or -2^30 -1
Atom (ATOM)	Unique internal representation of strings. Atoms have a Property List (PLIST) a Top Level Binding (TLB) and a Function Binding (FNB).
Binary#Code (BCD)	Machine instructions.
Ports (PORT)	Channels for input/output.

The CAR and CDR of a dotted pair may point to any other system objects, including the DTPR to which they belong. This is true for the PLIST, TLB, and FNB of atoms, although these cells are usually reserved for special things (see Weissman,C., LISP 1.5 PRIMER).

The name for an atom is a string of up to 64 characters of the set

a-z,A-Z,0-9,!,$,",',/,-,+,%,&,<,>,?,*,;,:,","

i.e. all characters except (,), ".", [,]. The atom must start with a printing character other than (,), [,], ".", or 0-9, and ends with a character from the set (,), [,], ., <space>, <tab>, <cr>, <lf>, <bs>, <ht>, <vt>, <ff>, <altmode>. When a token begins with a "-", a check is made on the next character to see if it is numeric (0-9); if the second character is numeric, the token is considered an integer. Otherwise, the token is made an atom. Non-printing or control characters may be imbedded within an atom (but may not start an atom). It should be noted that certain character have special meaning in general (^m is a carrige return) or may have special meaning to the Unix monitor (^c, ^d, etc.). Only the low order 7 bits of any character are saved; the "parity" bit is disposed of.

A character other than these may be included by starting an atom with a double quote; any character other than '"' may be included in such an atom. The atom ends with the next occurrence of '"'.

If two strings are identical, they represent the same atom. Note that upper case atoms (in full or part) are different then lower case atoms. All the atoms that the system knows about at the start are lower case.

The atom "ABC" is equivilent to ABC.

The cell of the atoms may be referenced by CAR, CDR, RPLACA, RPLACD, DEF, PUTD, and GETD, where the "CAR" of an atom is its PLIST, and "CDR" of an atom is its TLB.

In L110, numbers may or may not have a unique representation, depending on the value and the version of L110 in use. Please note that EQ always returns T when comparing numbers if the value of the numbers are the same, whether or not the numbers have the same cell.

The input format for numbers is a string of decimal digits, optionally starting with "-". The number is terminated when a non-numeric character is encountered.

FUNCTIONS

In L110 there are four function types:

1.) BINARY#CODE taking one argument unevaluated.

2.) BINARY#CODE taking n arguments, each one evaluated.

3.) Defined LAMBDA taking n arguments, each one evaled.

4.) Defined NLAMBDA taking one argument, unevaluated.

Expressions used to define functions must start with either LAMBDA or NLAMBDA. Following LAMBDA or NLAMBDA is a list of arguments (possibly null), where each atom of the list is the name for an argument for the function defined by the expression. For LAMBDA's, the arguments to the function will be evaluated and paired with these argument names. If not enough arguments are supplied, the argument names will be initilized to NIL. If too many arguments are supplied, the extra arguments will be evaluated and discarded. For NLAMBDA's, the argument list is bound to the first variable in the variable list, and the remaining variables are bound to NIL.

Built-In Functions

The functions listed below are built-in functions, i.e. they are defined by the basic system.

FUNCTION	*ARGS*	*DESCRIPTION (see Weissman)*
EXIT	--	Causes return to monitor
SYS	--	same as exit
DEF	--	(see Weissman)
NULL	1	if ARG=NIL returns T; otherwise, return t.
PUTD	2	ARG1 must be atomic. Defines the function cell of arg1 to be arg2.
GETD	1	ARG must be atomic. Returns function definitioni (possibly nil).
CONS	2	(pg. 30)
CAR	1	(pg. 31) CAR of an atom is PLIST
CDR	1	(pg. 32) CDR of an atom is TLB
ATOM	1	(pg. 74) (ATOM NIL)=T
DTPR	1	Returns T iff arg is DTPR
BCD	1	Returns T iff arg is BCD
PORT	1	Returns T iff arg is PORT
QUOTE	--	(pg. 59)
EVAL	1	(pg. 61)
PLUS	--	(pg. 83)
TIMES	--	(pg. 83)
ADD	2	Returns ARG1+ARG2
DIFFERENCE	2	(pg. 83)
DIFF	2	same as DIFFERENCE
QUOTIENT	2	(pg. 83)
QUO	2	same as QUOTIENT
NUMBERP	1	(pg. 76)
NUMBP	1	same as NUMBERP

LESSP	2	(pg. 77)
GREATERP	2	(pg. 77)
RPLACA	2	(pg. 146) value of RPLACA is ARG1
RPLACD	2	(pg. 146) value of RPLACD is ARG1
EQ	2	If ARG1 is the same object as ARG2 return T. If ARG1 and ARG2 are both INTs and they are the same value then return T. Otherwise, return NIL.
COND	--	(pg. 70) The conditional expression has been generalized so that instead of of doubles it accepts (M+1) tuples, where if the first element of the tuple evals to non-NIL then the 2nd through m'th elements are evaluated. The value of failng a condtional is NIL.
PROG	--	(see Weissman) The value of falling off a PROG is NIL.
RETURN	--	RETURN causes control to be transferred to the function calling the most recent PROG with the value of ARG1 as the value of the PROG.
GO	--	If the argument is non-atomic it is evaluated to (hopefully) an atom. Then all entered PROGs are searched for that label. GO has no value.
SETQ	2	(SETQ ARG1 ARG2) ARG2 is evaluated. If ARG1 is atomic, its current binding is set to the result of evaluating ARG2; otherwise an error is invoked.
SET	2	(set arg1 arg2) Like SETQ, but EVAL'S its arg1.

I-O Functions

L110 allows 7 ports to be open at once.

In the functions below, P stands for PORT, which must evaluate to either NIL or a PORT returned by INFILE or OUTFILE. Doing I-O to port NIL uses the teletype. It should be noted that the teletype buffer is dumped only when PRINT or TERPR is used. Thus PATOM does not dump the buffer. The result of this is that occasionally characters output by PATOM will not show up immediatly.

(INFILE S X)

If X is omitted or is NIL then the file (EVAL S) is opened for it. If X is non-null, then (EVAL S) in the system library is opened for input. The value of INFILE is a PORT.

(OUTFILE S)

(EVAL S) is opened for output. The value of OUTFILE is a PORT.

(CLOSE P)

The file corresponding to port P is closed.

(DRAIN P)

P must be and output port. The buffer for P is output. Drain is called implicitly by CLOSE, and TERPR and PRINT with NIL as arg.

(RESETIO)

All ports are closed. Returns NIL.

(PRINT X P)

X is printed on the file corresponding to P.

(PATOM X P)

If x is an atom, X is printed without double-quotes to file corresponding to P. If x is numberic, its low order 8 bits are output to the port specified. For teletype raw mode output, attention is called to the function EXEC, applied to the system program stty (though it should be remembered that no input processing is done in raw mode, therefore no ^c's, etc.).

(READ P)

One s-expression is read from the file corresponding to port P.

(RATOM P)

One token is input from the file corresponding to port P. On end of file, the special atom "eof" is returned.

(READC X)

Reads one character of port X and turns that character into an atom. The atom eof is returned on end of file. If input is being taken from the teletype (port nil) then a ^d with no other characters before it will cause "eof" to be returned. Otherwise, ^d will cause immediate transmission of the preceeding characters. ^c works as for ususal teletype input

(LOAD S X)

The file referred to by the algorithm in INFILE is opened, and (EVAL(READ)) is performed upon it until end-of-file is encountered.

Debugging Functions

(PROTOCOL X)

If X is NIL, the file L110.PROTOCOL is opened. Otherwise open the file X. Then copy out the input and output done on the teletype to that file.

(UNPROTOCOL B)

Close the PROTOCOL file if it is open. If B is non-NIL, then also Print and delete the protocol file.

(BT)

Prints history of EVAL activations on the teletype.

(RESET)

Causes immediate return to top level interpreter; current state is lost.

(RETBRK INT)

if arg is not an int, return nil. otherwise, if arg is >=0, then return to that particular break level. if arg is negative, then return to the <current level + n>'th level.

(BREAK X)

X is printed on the teletype, and control is transferred to the break package. Break can be used to suspend the execution of a program.

(CONT X)

Control is transferred to the most recent break. If the break was due to a non-continuable error,

CAN'T CONTINUE

is printed and control is transferred to the break level interpreter. Otherwise, the value of X is used to try to continue, or, if the break was due to a call from (BREAK), the value of X becomes the value of (BREAK).

Special Functions

(CONCATP 'atom)

CONCATP concats the string corresponding to the Lisp system's process ID to the atom.

Example:

foo becomes foo1234.

foo1234 is an atom.

(CONCAT 'atom 'arg)

If ARG is an integer, CONCAT results in the concatenation of the atom and the string representation of the integer. If ARF is an atom, results in the concatenation of the two atoms. The result is an atom.

(NTHCHAR 'atom int) or
   (NTHCHAR 'atom)

If only the atom is supplied, the result is an int which corresponds to the length of the atom's printname. If the int is supplied, an atom with only one character is returned. that character is the n'th character in the original atom, or null ("") if the int is out of range.

(GENSYM arg)

If ARG is non-INT, returns an atom such as GENSYM9999. If arg is INT, resets gensym number to low 16 bits of arg, and returns the INT. (GENSYM) decrements its number after each call.

(LINELENGTH X)

If X is an INT, it is taken to be the new width of the output line in characters. If X is not an INT, no action is taken. The width of the TTY line at point of return is always returned.

(CHARCNT P)

The number of characters remaining on the current line in port P is returned.

(TERPR P)

Prints a linefeed on port P. Resets the character count used by CHARCNT.

(PNTLEN ARG)

Returns length of PRINTNAME of ARG1 if it is INT or ATOM.

(RECLAIM ARG1 ARG2)

ARG1 and ARG2 must be NIL or numeric. Sets garbage collection parameters to return at least ARG1 DTPRS and ARG2 INTS. Invokes garbage collection; returns

(FDTPR.FINT).

($MUMBLE A B C)

If the type of C is an integer, the namestack length is adjusted to provide that many bound variable pairs before overflow. If B is NIL, then the garbage collector is enabled to collect just DTPRS and INTS. Otherwise, everything is collected. If A is NIL, the EVAL READER is set to be used. If A is non-NIL, then EVALQUOTE is set to be used. The value of $MUMBLE is A if C is non-numeric. If C is numeric, a reset is executed at the conclusion of the function call.

The prompts are as follows:

	EVAL	EVALQUOTE
Normal	->	<-
Break	nn:>	<nn:

where nn is the present break level.

Non-Primitive Functions

A number of non-primitive functions, i.e., functions defined by LAMBDA and NLAMBDA expressions are available in L110. A complete listing can be obtained by typing the file /LIB/LISP/AUXFNS. In most cases, the semantics for the functions can be found in Weissman.

Property List Functions

(PUTPROP ATOM PROPERTY IND)	A LAMBDA. This function puts PROPERTY on the PLIST of ATOM under IND. Returns PROPERTY.
(GET ATOM IND)	A LAMBDA. Searches PLIST of ATOM for IND, and returns associated property. If IND is not found, returns NIL.

Extended CAR and CDR

CAAR, CDAR, CADR, CDDR

Logical Functions

AND	pg. 78
OR	pg. 78

Arithmetic

(ADD1 X)	(PLUS X 1)
(SUB1 X)	(DIFF X 1)

MAP Functions

(MAPCAR FUNC LIST)	Applies FUNC to each element of LIST. Returns LIST of results.

(MAPC FUNC LIST)

Same as MAPCAR except does not copy LIST.

(FUNCTION F)

Used to QUOTE a function argument.

Function Definition

DEFEVQ(ATOM FUNCTION)

Same as DEF but used with EVALQUOTE.

Pretty Printing

(PP (FUNC1 FUNCN) or
PPEVQ((FUNC1 FUNCN))

Pretty prints the functions specified to the port currently bound to POPORT. Thus to PRETTY PRINT to a file,

```
(SETQ POPORT (OUTFILE 'FOO))
(pp (update teco))
(CLOSE POPORT)
(SETQ POPORT NIL)
```

($PRPR form)

Call to the inner pretty print routines. usefull to print out an arbitrary form.

User Functions

(EXEC arg1 arg2 ... argn)

Exec is a NLAMBDA that takes each arguement and passes them to the UNIX monitor as strings as described in the EXEC(II) system call. Exec returns the value of the process' %0 when it died. L110 waits until the process dies.

(SHELL)

Invokes a sub-shell; return to LISP with BY<e>.

(TECO 'file)

Teco is called on the file. If teco is exited with an EX or EQ the file is not loaded; if exited with EG, the file is loaded by LISP.

(TECF (func1 func2 ... funcn)) OR
TECFEVQ((func1.....funcn))

TECF creates a temporary file, PRETTY PRINTs the named functions to the file, then calls TECO on it. TECO may load the file on return, and then in any case the file and its backup are deleted.

(UPDATE 'file)

UPDATE replaces the definitions of any function defined in the current enviroment that occurs in the file. Useful in conjunction with TECF.

Loading Functions from Files

It is recommended that the user prepare on a separate file (using TECO) any function definition longer than one or two lines. The reason for this suggestion is that there is no way of editing the previous line of a function being entered on the teletype.

The function LOAD (see Primitives) reads and evaluates s-exps stored on a file. In particular, if the s-exps are of the form

```
        (DEF ATOM FUNCTION)
```

then the effect of the LOAD will be to define all of the functions in the file.

To edit a function definition, the user can either (EXIT) from L110 or invoke TECO. To do this:

```
        ->(TECO 'filename)
        UNIX TECO ...
        edit....
        EX$$
 OR
        EG$$
 OR
        EQ$$
        NIL
        ->
```

If EG is used the file will be loaded; if EX or EQ, the file will not be loaded.

Execution Errors

When an error occurs during the evaluation of an L110 function, the L110 break package is invoked. The break package will print out 1) a descriptive message identifying what the error is, a function name indicating where the error occurred, and either a :> or <: prompt. These prompts indicate that the stacks are suspended at the point of the error, and that the situation may be more thoroughly investigated.

Example:

```
        ->(DEF FOO (LAMBDA(X Y)(PROG() A (CAR X]
        FOO
        ->(FOO 1)
        CAN'T FOLLOW CAR OR CDR
        BREAKING        CAR
        1:>
```

The error occurred because we can't take the CAR of an INT. The user can now type commands just as he would at top level; the only difference is that the current bindings are available for examination. To continue with the example:

```
        1:>X
        1
        1:>Y
        NIL
```

To leave the break, the user may 1) return to top level with (RESET) or RESET(), 2) attempt to continue by (CONT above, the user may correct the bindings and continue with GO. For example:

```
        1:>(SETQ X (B C]
        (B C)
        1:>(GO A)
        B
        ->
```

Setting Breakpoints

The L110 break package can be invoked under circumstances other than error conditions. In particular, the function BREAK may be called from a user function to set a breakpoint at that spot, which is often useful during debugging. Break takes one argument which it evaluates and prints on the teletype.

```
        ->(DEF FUM (LAMBDA(X)
                (COND ((LESSP X 0)
                          (BREAK ' "VALUE NEGATIVE")))
                        (T (TIMES 2 X]
        FUM
        ->(FUM 3)
        6
        ->(FUM -3)
        "VALUE NEGATIVE"
        BREAKING BREAK
        1:>X
        -3
        1:>(CONT (PLUS X 3]
        0
        ->
```

"Emergency" Measures

CTRL-O and CTRL-C respectively suppress output and request L110 to stop prematurely. Typing CTRL-C causes control to pass to the break package; this error may be continued, although the value supplied by continue is ignored. Typing CNTRL-C five or six time will cause immediate execution of a reset.

CTRL-B causes a core dump to be taken of the system. It is usually not too useful.

If a ctrl-c is typed while reading a form, it will erase the entire form being input (even if it is several lines long) and simulate a retbrk to the most recent break level or else top level

Leaving L110

Typing CTRL-D to the prompt, or evaluating (EXIT) or (SYS) will cause return to the parent process.

Addition 1

In the LISP that is on the system, AND, ADD1, SUB1, OR, CAAR, CADR, CDAR, CDDR, LIST, APPEND, MAPCAR, MAPC, LENGTH, APPLY*,MEMBER, NCONC, and CONC are now hardcode.

Error Messages

Below are the error messages of the current implementation.

READ LIST ERROR	Reader didn’t like something you typed in.
ARITHMETIC OVERFLOW	You tried to make a number that was too large.
I-O ERROR	Probably trying to read a port in the wrong direction, or else trying to write on a port that you closed.
CAN’T READ PAST END OF PORT	You tried to read past the end of a file.
FILE NOT AVAILABLE	The file doesn’t exist or you have no permission.
ATTEMPt TO OPEN TOO MANY FILES	You tried to open more than seven files.
CANNOT ALLOCATE BUFFER FOR FILE	you are out of space. Try to close some unnecessary files and continue.
5 ˆC’S PANIC--RETURN TO LAST TOP LEVEL	You typed 5 ˆc’s.
ˆC DURING TYPE IN	You typed ˆc during type in. Return to last top level.
SEG VIOLATION	Lisp internal error--mainly intended for use with compiler.
CONTROL STACK OVERFLOW; RESET GENERATED	You had excessive recursion.

BUSS ERROR DURING GCOL-- LISP EXIT
A lisp internal error. Please send reports and the core dump to Forrest.

BUS ERROR
Internal bus error. Similar to Seg fault.

CAN'T CONTINUE
You attempted to continue from a non-continuable error.

CANNOT ALLOCATE ANOTHER ATOM PAGE
There is no more space-- try closing extra ports to free buffers up, if necessary. Non-continuable.

NAME STACK OVERFLOW
HARD NAME STACK OVERFLOW; RESET EXECUTED
You exceeded the name stack size. In the former case, ther was room to pass control to the break package--in the latter, a reset had to be executed.

NOT ENOUGH STACK SPACE TO ATTEMPT GCOL
This message may occur after a mumble that leaves very little control stack space. Try another $mumble.

CANNOT RECLAIM REQUIRED AMOUNT OF INTS OR DTPRS
The minimun amount of dtprs and ints (100 and 100 initially) are not available. free up space, and continue.

NO MORE DTPRS--HIT BREAK TO RETURN TO TOP LEVEL
This message is given when there are no more dtprs, hence no way to free up dtprs. As advertised, break will return to top level.

ATOM TOO LONG
you tried to make an atom with more than 100 characters.

ILLEGAL CHARACTER IN ATOM
An atom started with a garbage character.

UNDEFINED PROCEDURE
Eval could make no sense of your procedure.

NO PROG TO GO TO OR RETURN FROM
Return or goto could not find their destinations.

CAN'T FOLLOW CAR OR CDR
Attempt to take car or cdr of non-atom or non-dtpr.

IMPROPER USE OF SETQ
Set or Setq's first arg is not an atom.

ONLY ATOMS HAVE FUNCTION DEFINITIONS
You can only def or putd an atom.

NON-NUMERIC ARG TO ARITHMETIC SUBR
You tried to operate on a non-integer.

CANNOT MEET STACK REQUEST
Message from $mumble, telling you that you are greedy in your stack request.

BAD ARG TO SPECIAL SUBR
All purpose message used by concat, linelength, chrct, etc.