MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

System Program Memo.
Artificial Intelligence Memo. No. 127 March 1967.

# Incorporating MIDAS Routines into PDP6 LISP

Roland Silver

Some PDP6 LISP users have felt a need for a way to incorporate MIDAS subroutines into LISP. LISP has been changed to let you do this, using files found on the LISP SYSTEM microtape.

## 1. ASSEMBLING

You write a routine for LISP in much the same way that you write any other MIDAS relocatable subroutine. You must, however, observe the constraints imposed by LISP's allocation and use of accumulators, and its method of handling input, output, and interrupts. In addition, you require linkage to LISP before your routine can operate properly: The entry point(s) of the subroutine must be put on the property list(s) of the appropriate atom(s), and the address fields of instructions pointing to other routines, to list structure, or the other LISP data structures must be set properly. This is done when LISP begins operation -- after allocation, but before going into its listen loop.

We provide eight macros to ease the job of creating such linkages: SUBR, FSUBR, LSUBR, MACRO, QUOTE, E, SPECIAL, and SYM.

If you write "SUBR name" at a location a in your routine, LISP will subsequently ascribe the property SUBR to the atom name, with entry location a. Similar remarks apply to the use of FSUBR, LSUBR, and MACRO.

The significance and use of the other four macros is perhaps best communicated through examples:

1) An instruction like "MOVEI A,QUOTE (X Y Z)" will be assembled as "MOVEI A,0". At link time, however, LISP will insert the location of the list (X Y Z) into the address field of the instruction.

2) Suppose that the atom FOO has the properties shown in Figure 1. Then the instructions "MOVEI A,QUOTE FOO", "MOVEM B,SPECIAL FOO", "PUSHJ P,SYM FOO", and "CALL E FOO" will each be assembled with a zero address field, which will be modified at link time to be b, c, 106, and 101, respectively.

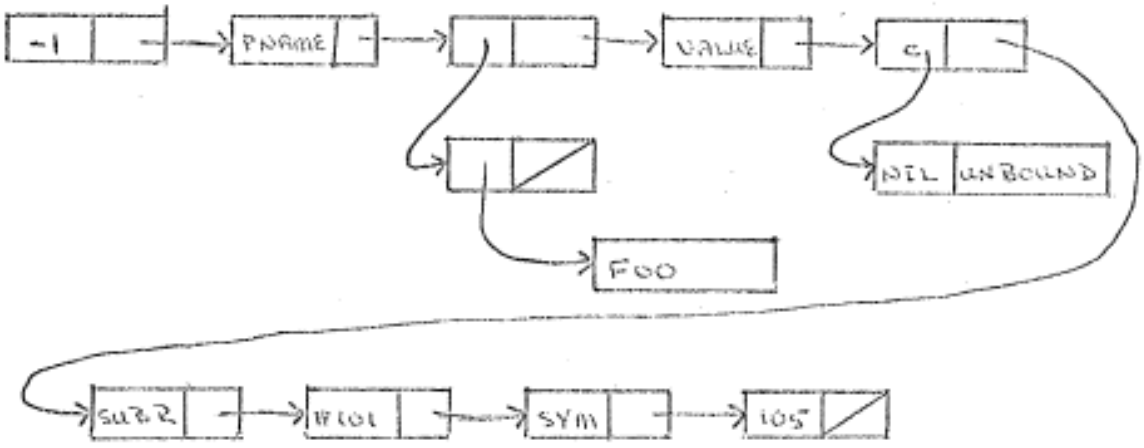


Figure 1

You should note that:

1) Of these macros, only QUOTE may be used with a nonatomic argument.

2) E can only be used with a routine which is known as a SUBR to LISP at link time.

3) SPECIAL will cause a value cell to be created for the atom named, if it does not already have one.

4) If your routine must do I/O in a more intimate manner than merely calling other subroutines such as print, read, or uwrite, then consult a system programmer before blindly using instructions such as CONO or DATAI.

Appendix I maps LISP's allocation and use of the accumulators. Appendix II lists code which properly defines the eight macros, as well as defining the UUOs CALL, JCALL, CALLF, and JCALLF, as well as difining the accumulators A,B,C,T, and P. This code appears as the file LISP MACROS on the LISP SYSTEM tape. We suggest that you merge it in at the beginning of your program.

If you are mystified, peruse AI Memo 116, PDP-6 LISP (LISP 1.6), January 1967.

## 2. LOADING

Your programs must be loaded together with LISP system programs by STINKD or STINK. Here is a recipe:

1) Have the 65K system tape on drive 1, LISP SYSTEM on drive 2 (say), and on drive 3 (say), a tape bearing the relocatable binary files (e.g. FOO BAR, and BAR FOO) that you wish to load.

2) Load STINKD (say), and then perform the following incantation:

```
2M1$L$$
3MFOO BAR$LMBAR FOO$L$$
2M2$@$$
```

The process will terminate with LISP and DDT loaded, with control in DDT. When you are ready, start LISP as usual with 100$G. LISP will begin by allocating storage, then linking your routines into the system. If an error occurs in processing a macro statement of the form mac expr, LISP will print out the vague comment "LINKAGE ERROR LINKING TO expr". After processing all linkages, LISP goes into its listen loop.

## 3. ACKNOWLEDGMENTS

Most of the ideas for this hack were suggested by Stuart Nelson. The other hackers contributed many useful suggestions and much help.

# APPENDIX I

## LISP ASSIGNMENT AND USE OF ACCUMULATORS

| | | |
|---|---|---|
| 0 | NIL | Atom head for NIL. |
| 1 | A" | Value of functions. Arg 1. Marked from. |
| 2 | B" | Arg 2. Marked from. |
| 3 | C" | Arg 3. Marked from. |
| 4 | AR1 | Arg 4. Marked from. |
| 5 | AR2A | Arg 5. Marked from. |
| 6 | T" | Used in calls to LSUBRs. Marked from. |
| 7 | TT | Marked from. |
| 10 | | Not marked. Clobbered by Garbage Collector. |
| 11 | S | Ditto. |
| 12 | D | Ditto. |
| 13 | R | Not marked. |
| 14 | P" | Pdl ac. |
| 15 | F | Free storage list ac. |
| 16 | FF | Full word space ac. |
| 17 | SP | Special pdl ac. |

## APPENDIX II

## CODE FOR DEFINING MACROS, UUOS, AND ACS

```
IRP MAC,,[SUBR,FSUBR,LSUBR,MACRO,QUOTE,SPECIAL,E,SYM]TYPE,,[0,1,2,3,4,5,6,7]
DEFINE MAC NAME/HERE,OLDEND

HERE=.-IFGE TYPE-4 1
EQUALS OLDEND END
DEFINE END
     HERE,,.LINKLIST"
.LINKLIST"=$.-1
     TYPE ←29.+ASCII /@NAME/
EQUALS END OLDEND
EXPUNGE HERE,OLDEND
END
TERMIN
TERMIN
TERMIN

CALL=74000,,
JCALL=75000,,
CALLF=76000,,
JCALLF=77000,,

.GLOBAL A B C T P
```