off

To use LISP, actually YKTLISP.
Rudiments:

This memo describes one way that one user found satisfactory. It is thought
to be of some use for new users.

Step 0: Some familarity with the user of VM/CMS is assumed. No drivers test
is required.

Step 1: Having enough storage. YKTLISP wants a large virtual store, less than 2M requires actions beyond the scope of this rudimentary explanation. YKTLISP is most efficient in the 4 to 6 Megabyte range. The ususal procedure
at this installation is to impose a lower and upper limit of storage in the system directory entry for each user. The lower limit is automatically given
when you IPL, to approach the upper limit the user must explicitly invoke:

```
DEF STOR nnM          where nn is the number of Megs.
IPL CMS               You must re-IPL.
```

This may not work because nn is bigger than your assigned upper limit. In that case you must contact USER SERVICES (the userid adminstrator x1091) and request an increase.

Step 2. Using YKTLISP EXEC to get LISP loaded:

```
YKTLISP fn ft fm (NODROP GETMIN 400K
```

The intent of this is to load a specific LISP core image named "fn ft fm" in such a manner that when you leave lisp via the (RET) function, LISP remains loaded in core. The GETMIN 400k option provides a large space for file editing. This may not work in which case you may be talking to the YKTLISP EXEC.

The following documentation is available:

The PDOM -- LISP/370 Program Description/Operations Manual (SH20-2076-0).
This manual gives partial coverage of the available facilities.

LISP/370 Concepts and Facilities, Fred Blair RC 7771.
Supplies about 4 chapters of materials missing from the PDOM.

LISP/370: A Short Technical description of the Implementation, by Jon L.
White. Available as NOT IBM internal use only appendix 2 to
LISP/370 Marketing Guide ZZ 20-4287. This guide also contains
another such appendix which is a bibliography on LISP.

The current set of LISP370 source file is:

AMACS	Data access macros for the LAP assembler functions.
ASMUTL	Sub-functions of the LAP assembler.
CMACROS	Macros which must be available to the interpreter.

COLDSTRT	The machine to build the system.
COMPUTL	Sub-functions of the LISP compiler.
COMP1	Pass one of the LISP compiler.
COMP2	Pass two of the LISP compiler.
DISPATCH	Functions which handle asynchronous interrupts.
ERROR	Functions which handle synchronous interrupts (errors).
EWRITE	Functions for elided printing.
EXPAND	Sub-functions of pass one of the LAP assembler.
FIXUP	One of the files which initilizes various global values.
FTRACE	The stack back-trace, &.
GENERAT	Sub-functions of pass two of the LAP assembler.
HMACROS	Macros which correspond to functions.
IODEF	Stream definition and sub-functions for input/output.
IOREAD1	Part of the s-expression reader.
IOREAD2	Part of the s-expression reader.
IOWRITE	The s-expression writer.
LAPUTL	Sub-functions of both passes of the LAP assembler.
LAP1	Pass one of the LAP assembler.
LAP2	Pass two of the LAP assembler.
LISTVEC	Functions for manipulating lists and vectors.
MACUTL	Sub-functions of various macros.
MATH	LISP arithmetic functions.
NEWDEF	The function definition facility.
NEWMATH	LAP arithmetic functions.
OMACROS	Operator macros.
PRETTYNW	The prettyprinter.
PRINTER	Sub-functions of the LAP assembler, to print listings.
QDEFS	Functional definitions of the macros in QMACROS.
QMACROS	Macros which generate inline code which does not check types.
RECLAIM	The LISP callable entry to the garbage collector.
RIODEF	Common functions for I/O to LISPLIB format files.
RIOREAD	Input functions for LISPLIB format files.
RIOWRITE	Output functions for LISPLIB format files.
SETGLOBS	One of the files which initilizes various global values.
SLAPUTL	System utilitys written in LAP.
SLOWDEFS	Functional definitions for the macros in HMACROS.
SUPV	The read-eval-print loop, EXF, and helpers.
SUTIL	System utilitys written in LISP.
SYSMAINT	FILESEG, FILELISP and helpers.
UDEFS	Functional definitions for the macros in UMACROS.
UMACROS	Macros of operations which probably should be FRs.
UTIL	A mixed bag of utility routines.

The distribution of functions among these files is not perfect, thought it keeps improving (by my taste, at least). There exist files of file type DIRECT and FUNTYPES as well as a file (currently) called LISP0036 NOTES, which give a little more information.

The DIRECT files simply map function names to file names. They usually lag a little behind reality, so check the dates of the DIRECT file and the file it is pointing at. Even if the LISP370 file is newer, chances are the function hasn't moved.

The FUNTYPES files contain a one line, often cryptic, description of each function, macro or global variable in the "base" system, i.e., the one created by COLDSTRT. It also indicates how much you should trust the PDOM, and

lists
the function/macro pairs. There are, in addition, entries for all of the functions provided by the system dependent code, LISPCMS. These may be enough
to let you know that it will do what you want, but not enough to let you do it.
The source for LISPCMS is heavily annotated, however, and should allow you to
cypher out the needed parameters.

LISP0036 NOTES (the numeric portion of the name may change at any time) contains several tables of commentary on various numeric error and/or informative codes which may be given to you during your use of YKTLISP.

Currently these files are maintained on ALBERGA 197 (migrateable disks MAINTAINed in parallel on the YKTVMV YKTVMT and YKTVMX systems). The following LISTFILE illustrates these files.

* Name	Type	Mode	Format	Records	K-bytes	Date	Time	Label
LISP0043	DIRECT	V2	F	80	1154	92	82/04/07	14:25:00	ALB197
LISP0043	FUNTYPES	V2	V	80	1131	76	82/04/07	14:25:00	ALB197
LISP0043	NOTES	V2	V	80	157	6	82/04/07	14:25:00	ALB197
AMACS	LISP370	V2	V	61	274	6	81/10/29	08:25:00	ALB197
.									
.									
.									
UTIL	LISP370	V2	V	80	890	28	82/04/29	12:38:00	ALB197
YKTLISP	MEMO	V2	F	80	808	64	82/04/07	15:24:00	ALB197