CMS-LISP I/O Supervisory Functions

This note describes several LISP functions providing the CMS user with high-level I/O operations from the LISP environment. These functions are of two types. The first type reads LISP files and evaluates all, or part of the input. The second type reads input data and PRETTYPRINTs it.

Functions which evaluate doublets from input files:

EXF, EXFP, EXFN, RDF, RDFP, GETCOM, GETCOMP, GETDEF, GETDEFP

Functions which PRETTYPRINT files:

PRTF, PRTFP, PCHF, PCHFP

All of these functions will accept input data from a CMS disk file, or from the user's console, and will generate output in a disk file, or on the user's console. The suffix P in a function name indicates that a CMS PRINT command is to be automatically executed for an output file written to disk.

Disk input files should normally have a file type of either SYSIN or LISP. File type SYSIN designates a file containing fixed-length, 80-byte records, from which only the first 71 bytes of each record are processed. File type LISP designates a file containing variable-length records where every byte of the records is processed. Input files contain LISP doublets. The keyword FIN placed in the file where a doublet is expected will act as an end-of-file indicator, and stop the input operation. This keyword normally occurs as the last item in the input file.

Except for the PCHF functions, output files have a page width of 120 bytes -- that is, they contain fixed-length 120-byte records. PCHF functions use an output page width of 68 bytes -- they generate fixed-length, 68-byte records.

All of these functions return the value (filename filetype) which identifies the output file. A 'filename' of LISPIT or NIL for an input file indicates input from the user's console. A 'filename' of LISPOT or NIL for an output file indicates output to the user's console.

The EXF functions read LISP doublets and evaluate them immediately, as they are read. The input file should have a file type of LISP or SYSIN, and the name of the file is specified as the first argument of the function. Thus, to evaluate expressions from the file 'TEST SYSIN A1', an appropriate top-level call is:

```
EXF (TEST)
```

The record of doublet evaluations is written into a disk file identified by 'LISTING filename A1'. For the example given above, this file would be 'LISTING TEST A1'. It is possible to have output data written to the user's console, instead of a disk file, by specifying a second argument of NIL. For example, using the same input file as above:

```
EXF (TEST NIL)
```

The function EXFN is analogous to EXF, except that it produces no record of the evaluations it performs. It generates approximately three lines of output, the top line, a blank line, and the last line of what would be produced by EXF.

The PRTF and PCHF families of functions are similar to EXF as far as arguments are concerned, but they produce output files with the file name {PRINT / PUNCH} instead of LISTING.

The RDF functions use two arguments. The first argument specifies the file name of an input file, and the second specifies the file name of an output file. These two file names must be different. The file type of the output file generated by RDF is OUT. Thus, to continue the earlier example, RDF could be used thus:

```
RDF (TEST LISTING)
```

with the output file having the fileid 'LISTING OUT A1'.

The default characteristics of an input or output file may be changed by explicitly using DISKDEF before invoking one of the above functions. If the user does this, it is then his responsibility to use SHUT to close the files for which he has used DISKDEF. Files for which the user has not used DISKDEF will be automatically closed.

It is possible to make use of this last characteristic to obtain concatenation of data into one file. For example, the output of several invocations of EXF might be concatenated into one disk file by the sequence:

```
DISKDEF (LISTING ...)
EXF (...)
EXF (...)
EXF (...)
SHUT (LISTING)
```

It is conceivable that files used by more than one of these I/O supervisory functions could be concatenated, but this might become tricky and is not recommended for the casual user.

965 - IBM - 04

The GETCOM and GETDEF functions require that the input file be previously defined by one of several mechanisms. The general way of doing this is by:

```
     SET (INFILE (filename filetype))
```

Several other functions effectively do this for special cases. Thus,

```
     INNAME (filename)
     INTYPE (filetype)
```

will search the accessible file directories to find a fileid with the specified file name or file type. The mechanism used for performing this search depends upon the values of the SPECIAL variables INTYPELIST and INNAMELIST. For example, INTYPELIST has the default value (SYSIN LISP), so when a file name is given and a file type must be resolved, a SYSIN file is tried first. If no SYSIN file exists with the necessary file name, the file type LISP is tested. If the required file with file type LISP does not exist, then any file with the necessary file name will be used. If more than one file with the required file name exists, the file used will be the first one encountered by CMS while examining file directories. The default value of INNAMELIST is (PUNCH PRINT LISTING DLISTING).

```
     ED (filename)
     ED (filename filetype)
```

will also cause INFILE to be set. Note that the functions EXF, RDF, PRTF and PCHF use the INNAME function described above to determine which input file to use.

The output file generated by GETCOM and GETDEF has the filename DLISTING and a file type which is the file name of the input file. Specifying NIL as the second argument of either of these functions will cause output to the user's console. If output is to a disk file, then summary output will be written to the user's console.

965 - IBM - 04

GETDEF and GETCOM take as their first argument a list of the names of functions to be defined or compiled, respectively. GETCOM will scan its input file until it encounters FIN in doublet position. There may be several FIN's in one input file -- the first one read stops the input operation. GETDEF

will scan its input file until either it encounters FIN, or until it succeeds in reading all of the functions specified by the first argument.

GETCOM knows about COMP360, DEFINE, COMPILE, MACRO, LAP360, EMBED, SET, CSET, SETQ and CSETQ, and executes any of these functions for arguments contained in the list supplied to GETCOM. If several arguments for one of these functions appear in the input doublet, and only some of these arguments were specified in the first argument of GETCOM, the COMP360 (or DEFINE, MACRO, etc.) function will be performed only for those arguments specified for GETCOM.

GETCOM executes all SPECIAL, UNSPECIAL, COMMON and UNCOMMON doublets irrespective of their arguments. All other doublets are ignored. Because GETCOM reads until a FIN is encountered, the same function may be redefined many times by GETCOM.

GETDEF also executes all SPECIAL, COMMON, UNSPECIAL and UNCOMMON doublets, but the only other operations it recognizes are COMP360, DEFINE, SET, CSET, SETQ and CSETQ. This latter group of functions is treated in a manner analogous to GETCOM's -- that is, a function must appear as the first item of a doublet, and it will be executed only for arguments which were specified in the list given as a parameter to GETDEF. When a COMP360 doublet is found for one of the functions specified in the first argument for GETDEF, it is evaluated as if it said DEFINE instead of COMP360.

Example: to read the input file used in the earlier examples in order to define the functions RED, GREEN and BLUE, the following might be used:

```
     SET (INFILE (TEST SYSIN))
     GETDEF ((RED GREEN BLUE) NIL)
```

In this example, output is to the user's console.

These functions were written in their present form by Mark Pivovonsky. This note was prepared by Richard Ryniker and Mark Pivovonsky.

May 28, 1974.