6. INPUT/OUTPUT

Input/output on the Xerox 1108 is supported in many ways: by keyboard, display, mouse, rigid disk, Ethernet, floppy disk, and RS232 communications. This chapter only covers aspects of the last two: the floppy disk and RS232 communications.

Floppy Support

This section describes Interlisp-D's file device {Floppy}, which is used to interact with floppy drives on Xerox 1108 machines.

Use an 8" double-sided, double-density, soft-sectored floppy disk (two kinds that will work are Datalife #18071 and Dysan #800803-02). Insert the floppy into the floppy drive face up, the edge of the floppy with two 2"-long cover holes (one hole per side) going in first. The write-protect notch will be on the same edge as and to the left of a cover hole as you look down on the floppy from above.

In general, Interlisp-D can be used to read, write, and otherwise interact with files on Pilot-formatted floppy disks through standard Interlisp input/output functions. All familiar operations such as LOAD, OPENFILE, READ, PRINT, BIN, BOUT, GETFILEINFO, SETFILEINFO, GETFILEPTR, SETFILEPTR, etc., work with floppies. COPYFILE is commonly used to archive and unarchive files between floppies and other file devices. Files on floppies can be compiled, edited, hard-copied, etc., just as files may be on all other ordinary file devices.

Naming, Erasing, and Formatting Floppies

To erase and establish track information on a floppy, especially a floppy that is brand-new, call

(FLOPPY.FORMAT *Name*
AutoConfirmFlg SlowFlg) [Function]

The *Name* argument becomes the name of your floppy. It can be any string or atom of 106 or fewer characters. It's a good idea to label the outside of your floppy with the same name using a sticky label and a soft marker.

AutoConfirmFlg controls questioning you about destroying the contents of a floppy that appears to contain valid information. If AUTOCONFIRMFLG is T, you will not be asked to confirm.

SlowFlg determines how thorough a formatting job is to be done on the floppy you format. If *SlowFlg* is T, FLOPPY.FORMAT completely erases your floppy, setting down track information and critical Pilot records on it. When *SlowFlg* is left NIL, only the Pilot records needed to give your floppy an empty directory are written. Use *SlowFlg* = T with a brand-new floppy.

Typically you will leave *AutoConfirmFlg* and *SlowFlg* as NIL. For example, doing (FLOPPY.FORMAT "My Floppy") will create a blank floppy named "My Floppy." Because formatting a floppy is a very intensive operation for the input/output processor (IOP), the IOP has less time to look at the mouse and keyboard. While you are formatting a floppy, the mouse cursor does not track the mouse as well and keystrokes may be lost. This is normal behavior; it simply means you cannot do much else while formatting a floppy.

To reset the name put onto a floppy by FLOPPY.FORMAT, use

(FLOPPY.NAME *Name*) [Function]

If *Name* is NIL, then FLOPPY.NAME reads the name put onto a floppy by FLOPPY.FORMAT or FLOPPY.NAME.

Copying Floppies

The ability to copy Pilot floppies is supported through two functions:

(FLOPPY.TO.FILE *ToFile*) [Function]

(FLOPPY.FROM.FILE *FromFile*) [Function]

FLOPPY.TO.FILE copies the contents of the current floppy to a file, and FLOPPY.FROM.FILE is FLOPPY.TO.FILE's inverse. For instance, to copy the contents of one floppy onto another, insert the floppy to be copied into the floppy drive. Then type (FLOPPY.TO.FILE 'ToFile). Remove the first floppy and insert a blank floppy. Type (FLOPPY.FROM.FILE 'FromFile). The first floppy has now been copied.

The *ToFile* outputted by FLOPPY.TO.FILE is approximately 2,500 pages long and can be placed on a file server or a logical volume of your machine. FLOPPY.FROM.FILE can be used more than once if you would like to make more than one copy. As an alternative to using FLOPPY.SCAVENGE (described below), the *ToFile* produced by FLOPPY.TO.FILE can be usefully edited to salvage the contents of a floppy that has been damaged.

Archiving to Floppies

Two special functions are available to make archiving to floppies exceptionally easy. These are:

(FLOPPY.ARCHIVE *Files Name*) [Function]

and

(FLOPPY.UNARCHIVE
{Host}<Directory>) [Function]

FLOPPY.ARCHIVE formats a floppy inserted into the floppy drive, giving the floppy the name *"Name* #1." FLOPPY.ARCHIVE then copies each file in Files to the freshly formatted floppy. If necessary, FLOPPY.ARCHIVE overflows onto successive floppies, which it names *Name*#2, *Name*#3, etc., each time prompting you to insert a new floppy. It is very convenient to archive an entire directory or subdirectory using FLOPPY.ARCHIVE in conjunction with Interlisp's DIRECTORY function. For example, (FLOPPY.ARCHIVE (DIRECTORY '{Server}<User>Project>*) 'Project) will copy all files on {Server}<User>Project> to floppies.

FLOPPY.UNARCHIVE is the inverse of FLOPPY.ARCHIVE. FLOPPY.UNARCHIVE copies all files on the current floppy to the directory {Host}<Directory>. Thus, (FLOPPY.UNARCHIVE '{Server}<User>Project>) will copy each file on the current floppy to the directory {Server}<User>Project>. If there is more than one floppy to unarchive, you should separately FLOPPY.UNARCHIVE each floppy.

Loading Sysouts and Other Large Files Onto Floppies

Sysouts may be created on floppies through Interlisp's SYSOUT or MAKESYS functions, then later installed on the same or another Xerox 1108 through the Installation Utility. Sysouts may also be taken from other file devices and put onto floppies through the use of the function FLOPPY.MODE (see below). To copy a sysout to floppies, simply do

(SYSOUT '{FLOPPY}) [Function]

or

(MAKESYS '{FLOPPY}) [Function]

You are prompted to insert new floppies as they are needed. It generally takes three to five floppies to store a sysout. To load in a sysout from floppies, you may do a 2-boot with the *Installation Utility* floppy in the floppy drive. Once in the *Installation Utility*, you can load a sysout onto logical volumes with the names Lisp, Lisp2, and Lisp3.

The normal mode of operation for {FLOPPY} is

(FLOPPY.MODE 'PILOT) [Function]

Three special modes of operation for floppies, SYSOUT, HUGEPILOT, and CPM, are also available. You can put floppies into either of the first two modes to copy sysouts or huge files from file servers to floppies or from floppies to file servers.

In SYSOUT mode, you may use COPYFILE to move a sysout off another file device onto floppies. To do this, you must first set the floppies into SYSOUT mode, then do the COPYFILE. For example, (FLOPPY.MODE 'SYSOUT) (COPYFILE '{FileServer}<Directory>LISP.SYSOUT '{FLOPPY}) will put the current Interlisp-D LISP.SYSOUT onto floppies. While in SYSOUT mode, you can copy as many sysouts as you like onto floppies. The *Installation Utility* is then used to load these sysouts onto a Xerox 1108. Similarly, sysouts can be copied from floppies onto another file device using COPYFILE when in SYSOUT mode. For example, (FLOPPY.MODE 'SYSOUT (COPYFILE '{FLOPPY} '{YourFileServer}<YourDirectory>Your.Sysout) will copy the sysout onto your directory on a file server. To get back to ordinary floppy operation, type (FLOPPY.MODE 'PILOT).

You can write and read huge Pilot files onto multiple floppies in HUGEPILOT mode, which can be set by typing (FLOPPY.MODE 'HUGEPILOT). This mode is

practically identical to SYSOUT mode, with the exception that you have control over the names of files and floppies.

Using CPM-Formatted Floppies

CPM is a particular kind of operating system that runs on many smaller computers. Such computers that also have 8" floppy drives, including the Xerox 820-II, can trade information with Xerox 1108 machines via CPM-formatted floppies. A number of CPM formats exist, but single-density single-sided (SDSS) CPM is standard. You can create, read, and write SDSS CPM-formatted floppies on 1108 machines.

CPM-formatted floppies are formatted differently than Pilot floppies, so you should use FLOPPY.MODE to switch to CPM mode when planning to work with CPM floppies. To create, read, or write CPM-formatted floppies, switch to CPM by typing (FLOPPY.MODE 'CPM). After switching to CPM mode, you may use FLOPPY.FORMAT to create CPM-formatted floppies.

When the IOP is in CPM mode, the usual input/output operations work with CPM-formatted floppies just as they do with Pilot-formatted floppies when you are in PILOT mode. There are a few limitations, however. CPM file names are limited to eight or fewer characters, with extensions of three or fewer characters. They do not have directories or version numbers. And CPM files are padded out with blanks to make file lengths multiples of 128.

To switch back to ordinary floppy operations, type (FLOPPY.MODE 'PILOT). This takes you out of CPM mode and puts you into PILOT mode, allowing you to resume formatting, reading, and writing ordinary Xerox Pilot floppies.

Manipulating Floppy Space

Two functions are available to tell you how much space is left on a floppy and to compact a floppy.

(FLOPPY.FREE.PAGES) [Function]

returns the number of unallocated free pages on the current floppy. Pilot floppy files are contiguously represented on a floppy disk. If you are using your floppy interactively (not just doing a simple series of COPYFILEs after a FLOPPY.FORMAT), don't cram your floppy to capacity. Try to keep such a floppy less than 75 percent full.

(FLOPPY.COMPACT) [Function]

can be used to compact your floppy, moving files toward the front of your floppy so that all free blocks on your floppy are combined into one free block. However, this may only be done if your floppy is stable, i.e., if there are no extant {FLOPPY} streams.

Testing Whether a Floppy Is in the Drive

Three functions are available for testing whether a floppy is in the floppy drive.

(FLOPPY.CAN.READP) [Function]

predicate tests if there is a floppy in the floppy drive. FLOPPY.CAN.READP does not provide any debouncing (protection against not fully closing the floppy drive door). You may wish to use FLOPPY.WAIT.FOR.FLOPPY (see below).

(FLOPPY.CAN.WRITEP) [Function]

predicate tests if there is a floppy in the floppy drive and the floppy drive can write on this floppy. (The floppy drive can only write on floppies whose write-protect notches are covered with tape.)

(FLOPPY.WAIT.FOR.FLOPPY *NewFlg*) [Function]

When *NewFlg* is NIL, this command waits until a floppy is in the floppy drive before returning. When *NewFlg* is T, the function first waits until the existing floppy in the floppy drive, if any, is removed, then waits for a floppy to be inserted into the drive before returning.

Scavenging Floppies

(FLOPPY.SCAVENGE) [Function]

attempts to repair a floppy whose critical records have become confused. Also, if you accidentally delete floppy files you shouldn't have deleted, FLOPPY.SCAVENGE retrieves them (provided you don't wait till after they have been overwritten by new files).