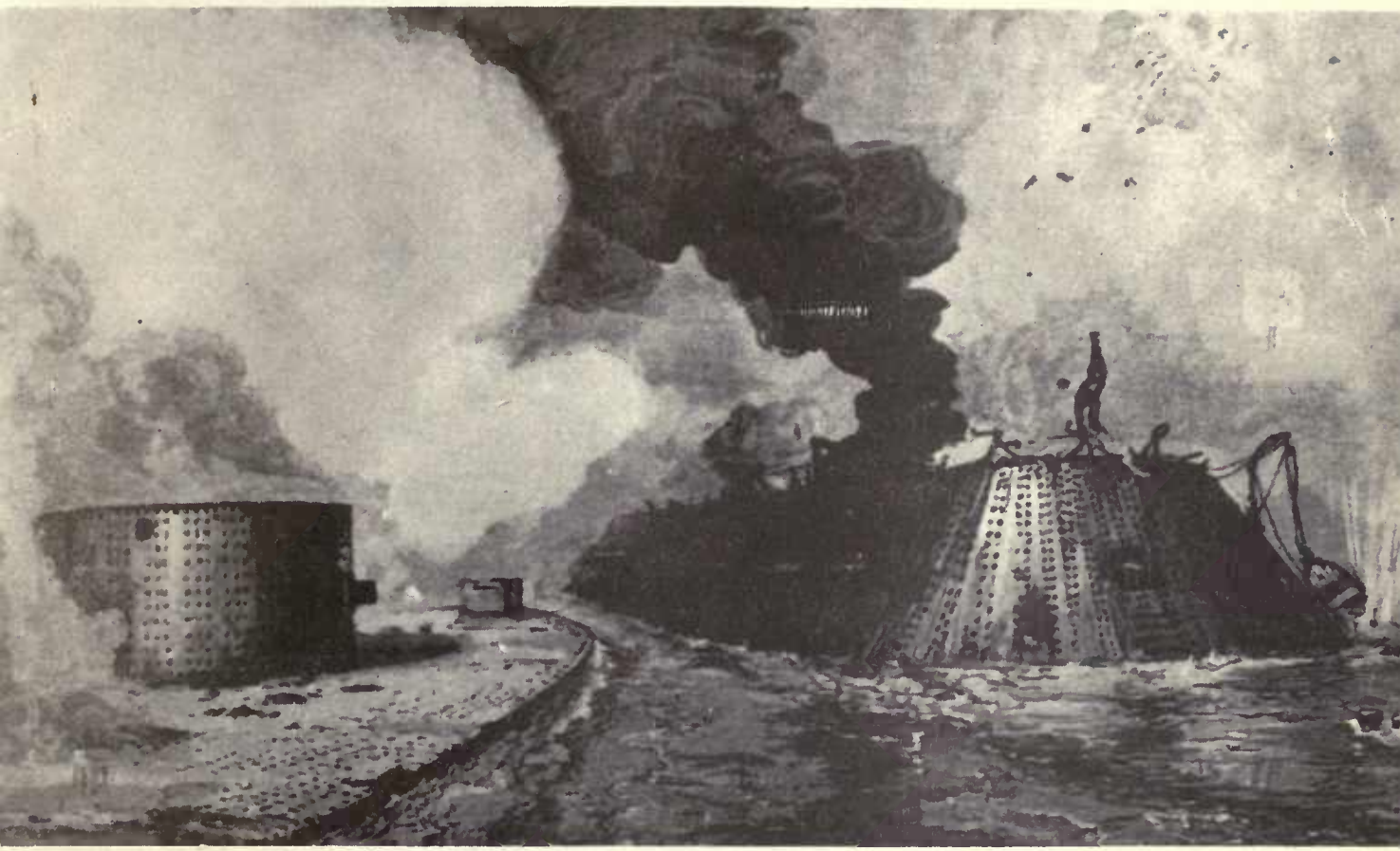
STANFORD ARTIFICIAL INTELLIGENCE LABORATORY December 1973
OPERATING NOTE 54.3



# MONITOR COMMAND MANUAL

STANFORD ARTIFICIAL INTELLIGENCE LABORATORY
OPERATING NOTE 54.3

December 1973

# MONITOR COMMAND MANUAL

by

Brian Harvey

ABSTRACT

This document describes the monitor commands available to users of the Stanford Artificial Intelligence Laboratory timesharing system, and the use of the terminals connected to this system. The first section is an introduction to the system for new users. Several appendices are included containing documentation of commonly used system programs. This manual supersedes SAILON 54.2 by Andy Moorer (*Monitor Manual, Chapter 1*).

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense under contract DAHC15-73-C-0435.

## ACKNOWLEDGMENTS

Several sections of this document were excerpted from program documentation written by others. Contributors include Ralph Gorin (RPG, SPOOL, DART), Dick Helliwell (COPY, DO, ZERO), Andy Moorer (TELNET), Dan Swinehart (FTP), Larry Tesler (FIND), Jim Stein (RSL), and Bo Eross (FIXIML). John McCarthy wrote the *Introduction to Timesharing* section. Martin Frost and Fred Wright provided invaluable technical assistance in publishing the manual, and Ralph Gorin, Brian McCune, and Fred Wright edited and corrected the draft versions. This document was produced on the Xerox Graphics Printer using the PUB program written by Larry Tesler and improved by Rich Johnsson and Tovar.

# TABLE OF CONTENTS

## APPENDICES

# SECTION 1

# INTRODUCTION FOR NEW USERS

This manual describes the monitor commands used in the PDP-10 computer system at the Stanford Artificial Intelligence Laboratory. The first section is an introduction for new users, including brief explanations of terminals, file storage, and some of the essential monitor commands. The computer is operated exclusively as a timesharing system, and the section begins with an introduction to timesharing.

## 1.1 Introduction to Timesharing

A timesharing computer system includes the computer itself (in our case a PDP-10 computer made by the Digital Equipment Corporation), user terminals for input to and output from the computer (in our case about 60 keyboard and display terminals), and a file system that keeps programs, data for the programs, reports, and other data for the users.

The basic idea of timesharing is to allow each user to behave as though he had a computer to himself controlled from his terminal. This is accomplished by having the computer cycle its attention among the users wanting service in rotation. It does this fast enough so that small requests are serviced in a time that ranges from a small fraction of a second to a few seconds, depending on the number of people demanding service at that instant. If the desired computation is long, the computer will do a bit of it, then service other people, then return to it, and so on until the computation is done. The time taken will depend on the size of the computation and how busy the machine is.

At any time the user's terminal is interacting with some program. These programs include the timesharing monitor; editors used to prepare programs, documents, reports and other data; various utility programs; and the user's own programs.

When you first sit down at a terminal, you will be interacting with the monitor, and you can always get back to the monitor by pressing the CALL key on the terminal. The first thing you have to do is to *log in*, which tells the monitor who you are so it can connect you to the file directory that contains your files and can do its accounting. The last thing you do is *log out* so that the terminal and other resources you have been occupying can be used by someone else.

You will have a certain quota of disk space and will be able to keep files such as programs between uses of the computer. Therefore, you will not ordinarily need any external form of storage such as punched cards or magnetic tape. (Our computer doesn't even have punched card equipment.)

A user gets service from programs by typing commands to the monitor. Some monitor commands perform the service and leave you again talking to the monitor. Others cause a program to be started; thereafter, you have to interact with that program in the way prescribed by it. When you are done with it, you go back to the monitor.

Your program also gets service from the monitor by means of special instructions, called UUOs, used in the program. The monitor provides many different services as a sort of subroutine of your program; most importantly, it is responsible for allocating resources, like core storage and input/output devices, among different jobs. Therefore, input and output must be done through the monitor to avoid conflicts. These UUOs are documented in their own manual; this one is about the commands you type at your terminal.

A typical task is to write a program in some computer language such as SAIL (our dialect of ALGOL) or LISP and then debug the program by running it, checking the results, and changing it until it gives the right answers. This is accomplished in the following steps:

1. First you run an editor. (We have several, but the current favorite is called E.) The editor reads whatever you type and displays it; what you type becomes the text of the program you are writing. However, the editor gives you convenient facilities for going back in your text and making changes that you think of as you go along. Most people don't write their programs on paper, but make them up as they use the editor. When the program is ready to try, you exit from the editor back to the monitor, and your program is a file on the disk with a name you have given it. At any time, you can re-edit this file to make changes in it. You can also print it on the line printer to take a copy home, and when you don't need it any more you can delete it.

2. Normally, the next step is to call a compiler to translate the program from a source language into machine language, call the loader to load it into your *core image*, and finally ask the computer to run it. You can do these things as separate operations, but facilities are provided for initiating all of this with a single command.

3. When the computer runs your program, it may just produce answers and stop. If you like the answers, perhaps you are done. Otherwise, you go back to the editor to make changes in the *source file* and try again. On the other hand, you may have written an interactive program which expects you to type input on the terminal. It interprets your input and gives output, then asks for more input, etc. Your program may read files and write them. It may display pictures on your terminal. Also, it may interact with the physical world by reading pictures from one of the TV cameras attached to the computer and by waving the mechanical arm.

4. In order to debug a program, you may have to do more than just look at the output. Therefore, the timesharing system has facilities for examining and changing the contents of registers and for putting break-points in programs.

## 1.2 Timesharing at the AI Lab

Our timesharing system runs on a Digital Equipment Corporation (DEC) PDP-10 computer. This is a 36-bit machine designed for timesharing applications, with 262,000 words of core storage. Our timesharing system, derived from old DEC software, has been extensively modified for our special requirements.

Programming languages available here include SAIL, a version of ALGOL with added features for backtracking, list processing, associative processing, and coroutining; several versions of LISP; FAIL, our PDP-10 assembly language; and the MICRO-PLANNER problem solving language.

To use the computer, you must type a LOGIN command at your terminal. (See Section 1.6 for a description of this command.) The monitor assigns you a job number, which it uses internally to keep track of you. Associated with your job are various resources, of which the most important is the core image, i.e., the simulated core storage maintained for you by the monitor so that your program can use addresses starting from zero although it may really be located anywhere in core while running. Your job only has one core image; if you type a monitor command which causes a program to be run, that program replaces your old core image. There is no way for one job to run two programs at once.

As a user of the Stanford AI Lab system, you will be given a *programmer name* to identify you to the computer. The main reason for this is so that the monitor knows which files on the disk storage belong to you. This name will probably be your initials. When you log in, you must type in this name, along with a *project name* which may identify what project you are working on. For example, if you are doing homework for a Computer Science course, the course number might be your project name. The combination is called a *project-programmer name* or *PPN* and is typed PRJ,PRG or sometimes [PRJ,PRG] when used in a file specification. Project "1" is a popular name with no special meaning.

Ours is just about the most heavily used computer we know of. It's hard to get good service, especially weekday afternoons. Therefore, please be careful about leaving jobs logged in when you leave, and don't play Spacewar during the day! Even a job which is not running uses up scarce resources.

Sometimes the system stops working while you are using it. You will notice that characters you type no longer appear on your terminal. In many cases, the system fixes itself after a few seconds, so all that happens is that a few characters you typed are lost. Otherwise, someone has to go fix it manually. Sometimes they can get it working again so that you can just continue with what you were doing. Other times, they have to load a new copy of the system, and you have to start over again. This is called *reloading*. If this happens, you will see a message to that effect on your terminal. (It is a good idea, when editing a long file, to save it on the disk every so often in case of a system crash.) If the system crashes and nobody is around to fix it, you may have to reload it yourself. See Appendix 10 to find out how.

Sometimes the system is stopped intentionally for software or hardware maintenance. This downtime is scheduled in advance, and part of the system messages you see when you log in is a maintenance schedule. Some time before the system is taken down for scheduled maintenance, you will see a warning on your terminal which counts down to the end. This warns you to save any files you are writing and stop what you are doing. Such maintenance is often scheduled between 5 and 7 weekday evenings.

This document, the *Monitor Command Manual*, describes the use of monitor commands typed in at terminals. Another manual describes the UUOs, instructions used by programs to communicate with the monitor. Other system information, including documentation of specific system programs, can be found in some printed manuals and on the disk; look at the [S,DOC] and [UP,DOC] file directories for program documentation. Some system programs, which are invoked by special monitor commands, are written up in appendices to this manual. Most of these appendices are derived from [UP,DOC] files; the latter are updated more frequently than this manual and should be considered the primary reference for those programs.

## 1.3 Introduction to Terminals

Most terminals on this system are display terminals; the characters you type appear on a display screen. There are also a few Teletypes, which use paper, and do not have the full character set of our displays. Teletypes are rarely used here, although there are a few on campus connected to our system.

There are two kinds of display terminals in use at the lab, Data Discs and IIIs ("triple-I"). They use the same keyboards, but have different display screens. The Data Discs, which are more numerous (about 60 of them), use TV monitors. (The name Data Disc actually refers not to the terminals themselves, but to the machine in the computer room which controls them.) The six IIIs are vector plotting displays, used mainly for graphics display programs like Spacewar. Unless you are writing a display program, the difference is not too important except that the character set on the IIIs is less legible than that on the Data Discs.

A vacant terminal should say TAKE ME, I'M YOURS on an otherwise blank screen. If it's completely blank, make sure it's turned on, and that the brightness and contrast (intensity on IIIs) are turned up. If it says NO DATA DISK CHANNELS LEFT, it means that the Data Disc, which can run only 31 terminals at once, is full. A free terminal is called *available* because the message used to say THIS CONSOLE AVAILABLE until Dick got to it.

Each of the printing character keys on our terminals has two characters printed on it (except the 0 key). The bottom one is usually a letter or digit, and the top one is some special character. There are two SHIFT keys and a SHIFT LOCK key, which make letters upper case. The keys which produce the special characters are labelled TOP. For example, hitting the "<D" key alone produces lower case "d"; also holding down SHIFT produces capital "D"; and holding down the TOP key makes it produce "<" (SHIFT and TOP together is interpreted as TOP).

When you type something on an available terminal, the available message is replaced by whatever you typed. The characters you type are kept in a special buffer called your *line editor* until you *activate* them by typing RETURN or a few other special characters. The line editor allows you to change the contents of the line by using special editing command characters. The characters in the line editor are displayed on the screen with two *cursors:* underlines on Data Discs and triangles on IIIs. The left cursor is underneath the first character in the line editor; the right cursor is under the place where the next character you type will go (generally the end of the line). When your line editor is empty, the cursors coincide. The line editor commands are explained in Section 2.2. Note: when you type RETURN, the monitor generally supplies a LINE character automatically. RETURN signifies a return to the left margin, and LINE an advance to the next line.

To get you started using display terminals before you read about all the line editor commands, all you need to remember is that the BS key will delete the last character, and the CLEAR key will delete the entire line. The CONTROL and META keys are used for other line editor functions.

When you are done with a terminal, be sure to log it out. Also, don't leave anything in the line editor--type RETURN before you leave if you're not sure. Your Data Disc channel will be made available soon after you log out, but not if there is anything in the line editor.

A note on terminology: In the body of this manual, characters are referred to by the name printed

on the keyboard, e.g., RETURN and FORM. In other documentation, including some of the appendices to this manual, different notations are used. Some of these are shown here:

| function | key | other notation |
|---|---|---|
| break | BREAK | <BRK> [BRK] |
| escape | ESC | <ESC> [ESC] <ESCAPE> |
| call | CALL | <CALL> [CALL] ↑C |
| clear | CLEAR | [CLEAR] ↑U |
| tab | TAB | <TAB> [TAB] <HT> |
| form feed | FORM | <FORM> [FORM] <FF> ± |
| vertical tab | VT | <VT> ∫ |
| backspace | BS | <BS> [BS] |
| carriage return | RETURN | [RETURN] <CR> CR |
| line feed | LINE | <LF> LF <LINEFEED> |
| shift | SHIFT | <SHIFT> |
| top | TOP | <TOP> |
| alt mode | ALT | <ALT> <ALTMODE> $ |
| meta | META | <META> β ± |
| control | CONTROL | <CONTROL> <CTRL> α ∫ |

The combination of RETURN and LINE (remember that LINE is supplied by the monitor) is sometimes referred to as CRLF.

## 1.4 Typing Commands to the Monitor

What you type may be read by different programs at different times, so you should be aware of what program you're talking to. When you first type on a vacant terminal, you are talking to the monitor's command decoder. You continue to talk to the monitor until you start running a program; then you're generally talking to that program. To distinguish these, the state of the terminal is described as *monitor mode* or *user mode*. When the monitor is ready to read a command from your terminal, it types a period. System programs which you run, like compilers and editors, generally type an asterisk when they are ready for a command.

It is possible to *type ahead* when the system is not waiting for typein. Characters you type which are not immediately read by either the monitor or a program are stored in your terminal's input buffer until they are needed. Programs can cause this buffer to be cleared. For example, some programs ask for confirmation of certain commands, e.g., when you specify for output a filename which already exists. Some such programs clear your input buffer first and others don't, so you can get confused by typing ahead to a program whose behavior you don't expect. Your input buffer is also cleared when you type CALL.

Monitor commands consist of the command name, possibly followed by arguments separated by spaces or punctuation, and then a RETURN. The command may be typed using upper and lower case letters interchangeably. Monitor command names may be typed in completely, or they may be abbreviated to only as many letters as are necessary to make the name unique; for example, the

LOAD command may be abbreviated LOA, but not LO because there is also a LOGIN command. A few common commands have single-letter abbreviations (L is accepted to mean LOGIN). Only the first six letters of a command name are relevant.

Some monitor commands are completely processed by the monitor, without running a program in your core image. For example, the DAYTIME command types the current date and time, without disturbing your core image. Most commands, however, require a program to be run. The monitor usually requires you to log in before you may run programs; there are a few exceptions, to permit system information programs like WHO, and of course the LOGIN program itself. If, when not logged in, you try to run a program which requires you to be logged in, the monitor automatically runs LOGIN, which will ask you for your project-programmer name. In this situation, when LOGIN finishes running, it will re-enter the command which you typed in the first place, so the command will be performed without making you retype it. Commands which require login but which refer to a pre-existing core image rather than starting a new program (e.g., START) do not cause this automatic login, but give an error message.

When you first enter a command which runs a program, the system assigns you a job number and types a line like

```
JOB 19     Stanford 6.13/F  8-15-73
```

which tells you what version of the monitor is running and when it was created, as well as your job number. (If it types

```
JOB CAPACITY EXCEEDED
```

too many jobs are already logged in.) When a program like WHO is run without logging in, the job is killed when the program exits. The LOGIN program, however, leaves you logged in when it exits.

Some of the monitor commands are briefly described starting in Section 1.6. The full descriptions of all the commands are in Sections 4 and 5, and in the appendices. Appendix 14 is a summary of the commands for quick reference.

## 1.5 File Storage

A file is a stored collection of information, perhaps a program or data. Most files are stored on a permanently available disk unit. Files on the disk must be identified by name and by owner so that you can get at the one you want. Files on other devices may also be used; we have magnetic tape, DECtape (a small mag tape unit unique to Digital Equipment Corp. machines), and paper tape, as well as a line printer and stuff like that. Whenever a program does input or output, it has to specify the particular file it wants. A file identifier is typed in the form

```
DEV:FILNAM.EXT[PRJ,PRG]
```

(some parts of that can be left out). DEV specifies the device you want. Generally, if you want the

## 1.7 Editing Commands

To enter your program, or to correct it, you will use one of our text editors. There are four editors in common use here:

1. E. This is the one you'll generally use. It is a display editor, much like TV (below) but much faster and with slightly different features.

2. TV. This is an earlier version of our display editor. It has been pretty much replaced by E, but there are still a few things TV can do which E can't, like switching between files in the middle of an edit.

3. SOS. This editor is designed for use at Teletypes or other non-display terminals. Files created with SOS include a line number at the beginning of every line. Editing commands use the line numbers to specify what lines to change, delete, or whatever. Hopefully you will have no need to use this editor, since you'll be using display terminals.

4. TECO. This editor is supplied by DEC. It is extremely powerful, in that it allows you to write editing programs which can process a file systematically, and it is very flexible about strange characters in the file. However, for routine editing it is somewhat inconvenient to use because it has an obscure command notation and does not use the display facility of the terminals.

Each of these editors is described in its own manual. A bibliography is included as an appendix to this manual. The point to be explained here is how to use monitor commands to invoke an editor; we'll assume here that you're using TV.

Suppose you want to write a program called PROG in the SAIL programming language. The monitor command to create a new file with TV is CTV. So you would type

```
CTV PROG.SAI
```

(SAI is the standard extension for SAIL programs.) This command tells the monitor to run the RPG (Rapid Program Generator) program, a command interpreter which abbreviates the sequence of commands needed to edit, compile, load, and debug programs. RPG reads the CTV command, which tells it to run TV, telling TV to create a new file named PROG.SAI. RPG remembers the last editing command you typed, so later you'll be able to edit the file simply by entering a TV command without having to type the file name again. (Note–LOGOUT makes RPG "forget" your commands, so you have to type the file name again when you next log in.)

The command for editing an already existing file with TV is TV. You can enter this command with no argument and it will remember the file from the last edit command, or you can specify a filename as with the CTV command above.

The E editor, which is the most commonly used here, includes provisions within it for carrying out the RPG functions itself. Therefore, the CETV and ETV commands analogous to CTV and TV do not run RPG; they start E directly. However, these commands are included with the RPG commands in this manual because they act the same to the user.

There is a file you can edit to learn how to use E. To do this, log in and enter the following command:

```
ET TEACH←TEACH[UP,DOC] (2P)
```

Then just read what appears on your screen and follow instructions.

## 1.8 Commands for Compiling and Running Programs

Once you have typed in your program, you have to compile it, load it, and run it. Compilers like SAIL put out relocatable binary programs. This means that the program is in a form which contains enough information so that it can be loaded into any address in core. Addresses within the program are all considered relative to the program's first word, so the LOADER has to add the actual address of that word to all the relative addresses in the program. Your program can be loaded along with others, for example a debugging program; the actual location of any program in your core image will depend on what other programs are loaded with it. (Note that this relocatable binary program format has nothing to do with the hardware relocation discussed on page 3.) The core image produced by the LOADER can be saved in a DMP (dump) file, which can then be run directly by a monitor command without going through the LOADER again.

Suppose you have typed in your program, PROG.SAI. You want to compile, load, and run it. You could say

```
EXECUTE PROG
```

and all those things would happen automatically. RPG would generate a command file for SAIL telling it to compile PROG.SAI and start up the loader, which also gets a command file. (RPG knows that it should use the SAIL compiler because of the SAI extension in your filename.) Again, RPG remembers the command, so next time you just have to say EXECUTE and it'll all happen. In fact, the editors all have commands which automatically re-do the last compile-type RPG command.

Suppose, to make this clearer, you find a bug in the program. To fix it, you can say

```
ETV
```

and RPG will start E editing PROG.SAI.

Then you use E commands to edit the file. When you're done, you use the exit-to-RPG command, which in E is CONTROL-X GO. This will automatically repeat the EXECUTE command you typed before.

There are many different versions of this compilation command: COMPILE will just compile the program, LOAD will compile and load, etc. The RPG commands are listed in Section 5.3 and fully described in Appendix 1.

## 1.9 Information Commands

You can sometimes find out how to use a system command by typing HELP followed by the command name. Typing HELP with no argument will list the kinds of help available this way. For instance, if you forget how to log in, you can type HELP LOGIN.

You can see the status of all the jobs on the system with the WHO command, abbreviated W. You can find out about jobs belonging to a particular user by typing WHERE prg (prg is the programmer name) or FINGER prg. These type out information about the given user's jobs; try them to find out more. All of these (HELP, WHO, WHERE, and FINGER) run programs in your core image, so you shouldn't run them if you need the program you have there. These information commands can be run without logging in first.

To get a list of the files in your directory, type the DIRECTORY command. You can get the list for another disk area by including a PPN in square brackets as an argument. You must be logged in to give this command.

## 1.10 Programs and Core Images

This section discusses the allocation and use of core storage for a job. The PDP-10 storage is divided into blocks of 1024 words, called 1K. Each job's core image is a multiple of 1K long. The core image may be located in any contiguous set of actual core blocks while the program is running.

Actually, a job may use two distinct contiguous sets of core blocks, called the *lower* and *upper segments*. The job must have a lower segment in order to run. This segment belongs to that job alone. If the job has an upper segment, it may be shared with other jobs. One use of this is to allow several users to run the same system program with only one copy of the program in core; the data for the individual users is stored in each job's lower segment, and the common code is in the upper segment.

An upper segment may be write protected, so the program in it cannot be accidentally changed. This is usually done with shared upper segments. Programs can control the write protection of their upper segment by UUO. Upper segments are more fully explained in the *UUO Manual*.

Each upper segment has an entry in the system job table, including a job number, a segment name, and much of the status information associated with jobs. Some of the monitor commands described in later sections of this manual refer to upper segments.

The lowest locations in each job's core image, called the *Job Data Area*, are used by the monitor to store information about the job. Some of this information is of interest only to the monitor, while other parts can be used and modified by the job itself. In particular, some monitor commands use information from Job Data Area words, e.g., starting addresses for the program. The description of such commands will refer to the relevant word by its symbolic name, of the form JOBxxx, e.g., JOBSA is the word containing the program's normal start address. The complete description of the Job Data Area is contained in the *UUO Manual*.

# SECTION 2

# DISPLAY TERMINALS

## 2.1 Keyboards

Our display terminals use a character set called *Stanford ASCII*. ASCII is a standard computer character set; our version is similar to the standard but somewhat extended. In particular, the "control" codes in ASCII, which mean things like "end of text," are used here for printing characters like α and ∃ and ∩. There are seven bits in a character code; since the PDP-10 has 36-bit words, five characters fit in a word with one bit left over. The complete Stanford ASCII character set is listed in Appendix 13.

Each of the printing character keys on our terminals has two characters printed on it (except the 0 key). The bottom one is usually a letter or digit, and the top one is some special character. There are two SHIFT keys and a SHIFT LOCK key, which make letters upper case. The keys which produce the special characters are labelled TOP. For example, hitting the "<D" key alone produces lower case "d"; also holding down SHIFT produces capital "D"; and holding down the TOP key with it produces "<" (SHIFT and TOP together is interpreted as TOP).

There are several special character codes, generally not printing characters, which are not affected by SHIFT and TOP. One is the space bar, which spaces as on any typewriter. The TAB key sends a tab code, which prints by spacing out to the next tab stop; these are set eight spaces apart and cannot be changed. The RETURN key sends a carriage return code, and the monitor generally provides a line feed code after it automatically. The LINE key sends a line feed alone; this is not too often useful. The FORM key sends a form feed code, and the VT key a vertical tab. These two are used by some programs as control codes, but not often for their printing effect. The BS key is backspace (in real ASCII this code is called delete). This code is usually not sent to your program, but is taken by the line editor as a command to back up the cursor. The ALT key sends an *alt mode*; this code has no printing significance, but is used as a control code by programs.

Although ASCII is a seven-bit code, display terminals actually send nine-bit codes to programs. The two extra bits represent the CONTROL and META keys. These keys are like SHIFT and TOP in that they are held down along with some other key. They have two different functions; when the character they are used with is sent to a program, it can be used as a control command. Editing and debugging programs use such single-character commands heavily. For example, in the E editor, CONTROL-META-D means to delete a line. However, the CONTROL and META keys are also used as commands to the system line editor for inserting characters inside a line, etc. The use of the line editor is explained in Section 2.2.

There are four more keys to be described. They are unusual in that none of them send characters to your program or to the monitor's command decoder; they are processed by the monitor's keyboard scanner and the line editor. CLEAR is simply a line editor command to delete the entire

line in the line editor. CALL is used to interrupt a program running in your core image and return your terminal to monitor mode. ESC (escape) and BREAK are used to signal that the characters which follow are a special command to the display service routines in the monitor; the many different functions of these commands will be explained in Section 2.4 and following sections. There should be a list of the ESC and BREAK commands attached to your keyboard; if not, see page 16 to find out how to print one.

## 2.2 The Line Editor

At a display, the characters you type are accumulated by a part of the monitor called the line editor. When you end a line by typing an *activation character*, the entire line is sent to your program (or to the monitor command decoder). Generally you will activate a line with the RETURN key, although others also work. The exact workings of activation are discussed in Section 2.3, but first, there are several editing command characters which can be used to change the text in the line editor.

There are two line editor *cursors* displayed on your screen. The left cursor is underneath the first character in the line editor; the right cursor is under the character position you are about to write into. The cursors are underlines on Data Discs, and triangles on IIIs. Also, on IIIs, the characters in the line editor are displayed somewhat below any other characters on the same line. If there is no text in the line editor, the two cursors coincide.

When you type a printing character into the line editor, it is displayed at the right cursor position, replacing any other character which was there before. The CONTROL and META keys are used to change the function of characters typed in. Also, several characters have different functions depending on whether or not they are typed with the right cursor at the end of the line. For example, BS at the end of the line deletes the last character; inside the line, it moves the cursor left one position without deleting anything.

META along with a printing character (including space and tab) inserts that character at the cursor position, in front of the character already there, rather than replacing the old character. At the end of the line, a META character will activate.

BS at the end of the line deletes the last character. If it's not at the end, BS just backs up the cursor without deleting. People talk about *deleting backspace* and *non-deleting backspace* to distinguish these. CONTROL-BS is always non-deleting, and META-BS is always deleting. (CONTROL-META-BS, like CONTROL-META-anything, is not a line editor command but an activation character.)

CONTROL-SPACE is the opposite of CONTROL-BS: it moves the cursor forward without changing the text. SPACE alone and META-SPACE act like any printing character. At the end of the line, CONTROL-SPACE does nothing (it neither moves the cursor nor activates).

The remaining editing commands consist of CONTROL-something. Some of them are only considered as line editor commands inside a line; at the end of the line, they activate. Whether or not this is true for a particular command will be mentioned in each description.

CONTROL-TAB moves the cursor to the end of the line, as if you typed CONTROL-SPACE many times. (Inside the line only.)

CONTROL-FORM is the opposite: it moves the cursor to the beginning of the line, so the two cursors coincide. (Inside or end of the line is ok for this one.)

CONTROL-D deletes the character above the right cursor. (Inside only.) Note that this is different from META-BS, which deletes the character to the left of the cursor.

CONTROL-S skips ahead from the current position of the right cursor to the next occurrence of the character you type next. If the next character you type does not occur to the right of the cursor, nothing happens. If the search character is a letter, the case must match. (Inside only.)

CONTROL-K is like CONTROL-S but instead of moving the cursor it kills characters up to (not including) the one you select. Again, nothing happens if the character does not occur. (Inside only.)

CONTROL-I enters *insert mode*. In this mode, all characters are treated as if you were holding down the META key, until you leave insert mode by typing another CONTROL command or activation character. That is, characters which you type are inserted instead of overwriting old ones, and BS is a deleting backspace. (Inside only.)

CONTROL-number (that is, any decimal number typed while holding down the CONTROL key) will cause a CONTROL command immediately following to be repeated that many times. For example, CONTROL-5 CONTROL-BS will move the cursor left five positions. This can be used to repeat the BS command as well as CONTROL-BS, META-BS, CONTROL-SPACE, CONTROL-D, CONTROL-S (finds the nth occurrence), and CONTROL-K.

CLEAR will delete everything in the line editor.

CONTROL-RETURN will retrieve the last line which was edited, as long as you haven't typed anything since then. That is, after you activate a line, if the very next character you type is CONTROL-RETURN, the line is returned to the line editor. You can send the same line again by just typing RETURN, or you can edit it. This is very handy when a monitor command, for example, fails because of a spelling error; you can simply say CONTROL-RETURN, correct the error, and say RETURN to activate the corrected command.

The TAB character is treated in a special way by the line editor. Commands like CONTROL-SPACE treat a TAB as a single character, so a CONTROL-SPACE when the cursor points to a TAB will move several spaces to the right. Typing a character without control bits, however, only replaces the first space of the TAB. There will still be a TAB in the buffer, unless the TAB was positioned so that it was only equivalent to one space, in which case it goes away completely when another character is typed over it.

## 2.3 Line Editor Activation

This section is about the workings of line editor activation: how the line editor decides to release characters to your terminal input buffer. There is quite a bit of arcane detail here, covering special cases, which is not generally important to the user. This section can be skipped by most readers.

The activation mechanism can be controlled to some extent by UUOs given by your program. Specifically, your terminal can be in *normal activation mode* or *special activation mode*. In the latter case, your program determines exactly which characters activate. The program can also disable the CONTROL-RETURN function, as well as the automatic insertion of line feed after return. However, the rest of this discussion assumes you are in normal activation mode.

Any character with CONTROL or META on will activate unless it is taken as a meaningful line editor command. In particular, META will activate at the end of the line. CONTROL always activates on characters which are not meaningful line editor commands, e.g., CONTROL-A always activates, and CONTROL-D activates at the end of the line only. CONTROL-META together always activates. RETURN, LINE, and ALT always activate, regardless of the CONTROL and META bits used. (Actually, in the case of RETURN, it is the line feed, supplied by the monitor, which is really the activation character.)

When you type an activation character, all the characters in the line editor are transferred to your terminal's input buffer and are no longer editable. If your program is doing line-at-a-time input, the character which activated the line editor may or may not be an activator for your program. (Normally, line editor activation characters also activate the program, but this may be changed with special activation. For example, the line editor cannot store characters with control bits, so such characters always activate the line editor if they are not line editor commands. Your program may, however, choose not to be activated by such characters, in which case they remain in your input buffer until a character which does activate the program is read.) When you type a program activation character inside a line, all characters in the line editor are moved to your input buffer, but your program can only read (in line mode) characters up to the activation character. The program may ask to get a nine-bit representation for the characters, so the CONTROL and META bits of the activation character are sent. There is a special kludge in the line editor so that whenever you type RETURN (without CONTROL or META), the RETURN is put at the end of the line, as if you had typed CONTROL-TAB first. This is not true for other activation characters, so characters after the activation character won't be editable, but won't be available to the program either until you activate again. Most programs are written so that something fairly intuitive happens when you activate in the middle of the line, but it is possible to confuse yourself. In particular, when you re-edit (by typing CONTROL-RETURN) a line which was activated with something other than RETURN, the activation character appears in the line (without control bits); if you type RETURN, you get the same old activation character, not a RETURN at all! There are some peculiar things which happen if you delete the old activation character first, though--it is possible to have the RETURN activate the line editor, but not put in any activation character and not actually activate your program until you type a second RETURN.

## 2.4 ESCAPE and BREAK Commands

Certain special characters are used to control the status of your display terminal. These are processed directly by the monitor's keyboard scanner, whether or not you are in monitor mode. One example, mentioned above, is CALL. This key stops your program, if one is running, and returns you to monitor mode. You can also type CONTROL-CALL or META-CALL for a *deferred call*—the program is stopped when it tries to read the CALL character from your input buffer. Thus you can type some commands into a program, followed by a deferred call; when the program is finished with your commands and asks for more, it will be stopped.

There are several commands using the ESC and BREAK keys. Generally, the format is ESC or BREAK, then possibly a numeric argument, then a letter which specifies the command. There should be a concise summary of these commands attached to your keyboard; if not, log in and type this:

```
XGPLIST DDKEY.BH[UP,DOC]/FONT=FIX20
```

(or IIIKEY instead of DDKEY if you are using a III terminal) to print the list on the XGP.

ESC O will stop typeout from your program. Any characters already in your output buffer are deleted, and any attempted typeout from your program is ignored. The program continues running, however. Typeout is resumed if you type BREAK O, or if your program reads any characters from your terminal.

Display terminals are normally in *full character set mode*, which means you can type in upper and lower case letters. You can leave this mode by typing BREAK F, which has essentially the same effect as pushing the SHIFT LOCK key. You type ESC F to return to full character set mode. The only use for this that I know of is that you can convert a line of text in a file to upper case in E by typing BREAK F, then something like CONTROL-SPACE to load the line into the line editor, then RETURN. Try it.

Some programs which read data from your terminal expect an "end of file" indication after the data. The thing to type to supply this signal is CONTROL-META-LINE.

It is possible to suspend typeout from your program temporarily without losing output, as with ESC O. Typing CONTROL-BREAK starts this automatic holding. The word HOLDING appears at the top of your screen, and the program is stopped if it tries to type more characters than fit in your output buffer. Automatic holding is released by typing any line editor activation character, or CONTROL-CLEAR.

Sometimes you may want to write a program which you can interrupt with a signal from your keyboard, without having to keep checking for terminal input in the program. For example, if you run a program which searches a large data base by command, you may want to be able to abort a search without stopping the program. Typing ESC I will generate an interrupt condition which your program may enable; the use of program interrupts is described in the *UUO Manual*.

## 2.5 The WHO Line

The next set of commands have to do with the *WHO line*, a display of system status and the status of a particular job on the top two lines of your screen. Once you start a WHO line display, it is automatically updated until you turn it off explicitly. To start a WHO line for the job at your terminal, type ESC W. You can see the WHO line for another job, job number n, by typing ESC n W. To turn off your WHO line display, type BREAK W. (Note: this does not actually make the WHO line disappear on Data Discs, where anything you display stays on the screen until it is explicitly erased.)

One use of the WHO line is to keep track of the progress of a job of yours other than the one at the terminal you're using. To facilitate this, typing ESC Q will display the WHO line of the next job in the monitor's job table with your programmer name (or with the same name as the job whose WHO line you are already displaying, if any). Typing ESC Q repeatedly will cycle through all such jobs. BREAK Q does the same thing, except that it searches the job table backwards.

This is the format of the WHO line:

```
          DD JBS,TCOR  R,RCOR UCOR  NL%  DSKQ  DATE  DAY  TIME
JOB PPN TTY  QUEUE   JOBNAM SIZE RUNTIME-RP XTIME-XP DSKOPS SEGNAM (ALIAS)
```

The first line is the *system WHO line*:

| | |
|---|---|
| DD | number of Data Disc channels in use (maximum=31) |
| JBS | number of jobs logged in |
| TCOR | total core of all jobs (including swapped out) in blocks of 1024 words |
| R | number of jobs actually run in the last few seconds |
| RCOR | total core of the jobs counted in R |
| UCOR | amount of core available for running user jobs |
| NL | system null-time percentage over the last few seconds |
| DSKQ | number of jobs waiting to use the disk |
| DATE | uh, let's see... |
| DAY | day of the week |
| TIME | time of day (24-hr clock) |

The second line refers to a particular job:

| | |
|---|---|
| JOB | the job number of the job being displayed |
| PPN | its logged-in project-programmer name |
| TTY | its terminal number (in octal) or DET for detached jobs |
| QUEUE | the system queue that the job is in (see below) |
| JOBNAM | the program being run |
| SIZE | size of the job's core image |
| RUNTIME | total run time since login, in minutes and seconds |
| RP | RUNTIME expressed as a percentage of the job's total wait time (time the job was runnable but not running) |
| XTIME | incremental runtime, normally reset on terminal input but see below |
| XP | XTIME as a percentage of incremental wait time |
| DSKOPS | number of incremental disk operations (reset along with XTIME) |
| SEGNAM | name of upper segment if any |
| ALIAS | disk PPN (see below) if any |

Some of these need more explanation. The QUEUE is an indication of the immediate status of the job--is it runnable, waiting for input, stopped, etc. These are the possible queue names:

| | |
|---|---|
| RUNQ | The job is runnable and not waiting for I/O, etc. |
| TQ | This is a high-priority run queue which jobs are in for a short time after leaving certain kinds of I/O wait. |
| STOP | The job is stopped because it exited, the user typed CALL, or there was an error in running the job. |
| NULQ | The job has no core image. |
| IOWQ | The job is waiting for some input/output operation, or for the completion of a SLEEP or JOBRD UUO. This most often means that the job is waiting for the user to type something. |
| DIOW | The job is waiting for some I/O operation and may not be swapped out until it is completed. |
| INTW | The job has suspended itself waiting for an interrupt. |
| DWQ | The job is waiting for a device (e.g., line printer) to be free. |
| MTQ | The job is waiting for the magnetic tape controller. |
| DTQ | The job is waiting for the DECtape controller. |
| DCQ | The job is waiting for the data controller (used for mag and DEC tapes). |
| CMQ | The job is not in core and needs to be brought in quickly; if a job stays in this queue for long, something is wrong. |

Some special characters may be displayed after the queue name:

| | |
|---|---|
| - | The job is currently running. |
| * | The job is swapped out. A second * means the upper segment is swapped out. |
| @ | The job has been locked in core by the LOCK UUO. |
| \ | The job is running a spacewar module on the PDP-10. |
| / | The job is running a spacewar module on the PDP-6. |
| X | The job is running spacewar modules on both processors. |

Some of the above may not be clear until you read the *UUO Manual*. The *alias* is a PPN used as the default directory for disk files instead of your logged-in PPN. There is an ALIAS monitor command to set the alias for your job.

XTIME is an incremental run time, measured in minutes, seconds, and 60ths of a second (a 60th of a second is called a *tick*). It is normally reset whenever the program is started by a monitor command other than CONTINUE, and also whenever the job is awakened by input from the terminal after having been in IOWQ waiting for such input. Note that if you type ahead, so that the job never enters IOWQ, terminal input will not reset XTIME. It is possible to inhibit resetting of XTIME with the BREAK X command (see below). If resetting is inhibited, the character separating spaces from ticks in the display will be ' instead of ' as it usually is.

BREAK X inhibits resetting your job's XTIME. ESC X resumes the automatic resetting as described above. Neither of these changes the current value of your XTIME. Typing ESC 1 X will set your XTIME to zero; BREAK 1 X will set it equal to your RTIME (total run time). Both ESC 1 X and BREAK 1 X inhibit automatic resetting, so you have to type ESC X if you want to restore this.

## 2.6 Page Printer Control

There are two ways of printing on display terminals. To exploit the full capabilities of the display for graphics, display programs can explicitly control the positioning of points, vectors, or characters.

On III terminals, 16 such display programs can be written independently, and any combination of them *activated* or displayed at any time under program control. These display programs are called *pieces of glass*. On Data Disc terminals, only one display program can be run on each Data Disc channel, although the effect of multiple pieces of glass can be gotten in a different way by displaying more than one channel on the screen (see Section 2.7, which describes the video switch).

The other way of printing on a display terminal is simply to output text to be printed in order as if on a piece of paper. The monitor automatically "scrolls" such output; when the area of the screen being used fills up, the text is moved up a few lines, with the top part disappearing. Sixteen of these *pieces of paper* can be displayed at once, on different parts of the screen. At any time, one of these pieces of paper must be selected by the program as the one to receive *page printer* output. The dimensions of each piece of paper are controlled by the program, but the currently selected one can also be controlled from the terminal with ESC commands.

The dimensions of the page printer include several factors. First, the vertical position of the top line of the page. This can be set by the command ESC n Y for line n. (Lines 1 and 2 are where the WHO line goes; the normal position is line 4.) ESC Y resets the vertical position to normal.

The scrolling function is controlled by two parameters: the number of lines typed before a *glitch* (the upward repositioning of the text), and the number of glitches on the page. These are set by ESC n L for lines/glitch and ESC n G for glitches/screen. The defaults are 4 glitches of 9 lines for Data Disc terminals, and 12 glitches of 2 lines for III terminals. ESC L and ESC G restore the respective counts to normal.

It is possible to cause automatic holding, as with CONTROL-BREAK, every n lines or every n glitches. The commands for this are ESC n E for every n lines, and ESC n J for every n glitches. At each pause, the word HOLDING will appear on the third line of the display. CONTROL-CLEAR will resume output as usual. ESC E and ESC J disable the respective automatic pauses.

Typing ESC N will normalize your page printer. BREAK N will clear your screen, normalize the page printer, and display the text in the page at the normal position. To refresh the page printer display without changing the page printer geometry, type ESC P. BREAK P will clear the screen and then do ESC P. These are useful for getting rid of any noise which may appear on your screen. There is a rarely-used ESC R to refresh just the display of your line editor.

Note: Any RESET of your job, either by the RESET UUO or automatically by the system, eliminates all but piece of paper 0, selects that piece of paper for further output, and normalizes the page printer display. The *UUO Manual* explains the RESET function.

## 2.7 Data Disc Control

Certain ESC and BREAK commands have special functions for Data Disc terminals only, and some for III terminals only. This section is about the Data Disc ones.

The usual display on a Data Disc screen is green letters (hereafter called "white") on a black background. It is also possible to display black on white, by typing BREAK C. To restore the normal white on black, type ESC C. Both of these clear the screen, so you have to type ESC P to refresh your page printer display. Black on white display requires increasing the contrast and brightness controls to be visible. This is not the world's most useful feature; it is also possible, under program control, to display only part of the screen in black on white.

The remaining Data Disc commands are used to control the *video switch*. This device allows you to select from the 32 Data Disc channels, plus various TV cameras and video synthesizers, what you want to see on your screen. You can see more than one Data Disc channel at a time, but only one of the *analog* channels (the cameras). You can select an analog channel along with a Data Disc channel only if the two are synchronized; e.g., the lounge TV signal can't be combined with a Data Disc channel because the synchronization of the former is not under our control.

Data Disc channels in use by a job may be *public* or *private*. Only public channels may be selected by other users. The channel used by a terminal is normally private. The command ESC H *hides* (makes private) your terminal's channel; BREAK H makes it public.

Video switch commands are generally of the form ESC n x, where n is a channel number (Data Disc or analog channel) and x is the command letter. The form BREAK n x interprets n as a terminal line number. Thus, ESC 30 S selects Data Disc channel 30, but BREAK 30 S selects whatever channel is assigned to TTY30.

ESC or BREAK n S makes a *permanent* selection, which remains in effect until you give another video switch command to change it. ESC or BREAK n T makes a *temporary* selection, which reverts to the permanent selection whenever your job is RESET by UUO or automatically. To display more than one Data Disc channel at once, type ESC or BREAK n A to add a channel; ESC or BREAK n D deletes a channel.

Using ESC with no numeric argument on a video switch command means your terminal's channel. Thus, ESC S restores your own display. BREAK with no numeric argument is the same as ESC 45, which selects the lounge TV. Thus, BREAK S at 6pm on a weekday selects Star Trek. The ESC N and BREAK N commands, which normalize the page printer on all displays, also imply ESC S on Data Discs.

Note that your program can also modify the video switch settings by UUO.

## 2.8 III Control

At a III terminal, several different display programs, called *pieces of glass* ("pog"), may be running at once. You can choose which of these (zero or more at once) are actually displayed.

ESC n S selects piece of glass number n only. ESC n A adds piece of glass n, and ESC n D deletes pog n. If no numeric argument is used, these commands apply to all pogs, e.g., ESC D clears all pogs from the screen.

ESC C eliminates all pogs (your page printer is still displayed). It is different from ESC D in that the display programs are completely forgotten, so they cannot be reselected.

Note that the selection of display programs can also be controlled by a UUO in your program.

## 2.9 Audio Switch Control

Associated with each display terminal is a loudspeaker which can be connected to various sound sources under computer control, by an audio switch analogous to the video switch for Data Disc channels. There are 20 (octal) audio switch input channels, numbered 0 to 17. Only one channel can be connected to your speaker at a time.

One of the input channels is normally silent, but is connected to the telephone paging system when a page is in progress (i.e., someone has dialed 9 on the com line). When you are listening to another channel, you may choose to be switched to the paging channel automatically when a page is in progress and then returned to your original channel.

The command ESC n U will select audio channel n and allow paging interruption. BREAK n U selects channel n and does not allow paging. ESC U or BREAK U selects the silence/paging channel.

At a Data Disc, the BREAK S, BREAK T, and BREAK A commands, which select the lounge TV picture for your screen, also select the lounge TV audio for your speaker.

As we go to press, the following audio switch inputs are connected:

```
0       silence/paging
1       lounge TV
2       tuner in Helliwell's office
3       tuner in machine room
4       digital-to-analog converter channel 1
5       beep generator
```

# SECTION 3

# TELETYPES

Teletypes do not have display screens, nor do they have the full Stanford ASCII character set. This means that certain things are typed differently.

The CONTROL (CTRL) key on a Teletype does not produce the same codes as CONTROL on a display. Instead, it produces codes which, on display terminals, represent some of the TOP characters. Most programs which deal with Teletypes, however, either cleverly use only characters which are the same on all terminals, or have special notations for use at Teletypes. In particular, there is an *SOS representation* (named after a Teletype editor) for printing characters not found on Teletypes, namely a question mark followed by some other character. See Appendix 13.

To get the effect of CALL, type CONTROL-C twice. (Control characters on Teletypes generally echo as ↑<character>, so now you know why CALL on displays prints ↑C.) Deferred CALL is a single CONTROL-C.

Although there are several models of Teletypes and similar terminals, most of ours do not have lower case letters. The SHIFT key on these terminals, like the TOP key on our displays, produces non-alphanumeric printing characters. Most of these are labelled on the Teletype keys, but you may not find [ (SHIFT-K), \ (SHIFT-L), or ] (SHIFT-M) labelled on some models.

There is no line editor on Teletypes. You can delete the last character on a line by typing RUBOUT, which prints a backslash and then the deleted character. Successive RUBOUTs will print the characters deleted, until a non-RUBOUT, which prints another backslash and then whatever you typed. CONTROL-U will delete the entire line, like CLEAR on a display.

Most of the ESC and BREAK commands have no Teletype equivalent. CONTROL-O flushes typeout like ESC O. A second CONTROL-O resumes typeout like BREAK O. CONTROL-B starts and stops typeout holding, like CONTROL-BREAK and CONTROL-CLEAR on display terminals. There are monitor commands to provide some of the ESC facilities; e.g.,

```
TTY WHO
```

will type out your who line once.

The end-of-file character for Teletypes is CONTROL-Z. If your program does input by TTYUUO, CONTROL-Z is converted to 612 (the code for CONTROL-META-LINE). TAB is typed as CONTROL-I. FORM is CONTROL-L, and VT is CONTROL-K. RETURN, LINE, and ALT have corresponding Teletype keys. (The key for ALT may be labelled ALT MODE, ESCAPE, or PREFIX depending on when the Teletype was built.)

It is possible to use our computer from another computer, over the ARPA network, or by telephone connection. The Teletype conventions also apply to these connections.

# SECTION 4

# BASIC MONITOR COMMANDS

Commands like LOGIN tell the monitor to run particular system programs in your core image. In this section, monitor commands which do not refer to a specific program are described. Some of them do not refer to programs in your core image at all, but are handled entirely within the monitor itself; others run programs, but allow you to specify the program as a command argument rather than implying a particular one like LOGIN.

## 4.1 Running Programs

The RUN command takes as arguments an optional device name, a file name, and an optional core size argument. If the first argument is not a device name, DSK is used. The only allowable devices are DSK, SYS, MTAn, and DTAn. Running programs from magnetic tape or DECtape is not recommended. If the file name does not include an extension, DMP is implied.

The core size argument, if any, should be a decimal number indicating the number of 1K (1024 word) blocks desired for your core image. This must be at least as much as the saved core image, but you can ask for more. Certain programs with variable buffer space will use as much as you initially allocate. (Programs can increase their core size dynamically by UUO.)

If you are running from magnetic tape, the core size argument must be used; otherwise, the size of your core image before giving the command will be used. This is because the monitor cannot determine the size of a file on magnetic tape before reading it.

The program is loaded into your core image, your job name is set to the name of the dump file, and the program is started at the starting address specified in the dump file.

The R command is used for running system programs. The command

```
R FOO 10
```

is identical to

```
RUN SYS:FOO 10
```

(The SYS device is actually the disk, but with the special system directory [1,3] implied. This directory is used to store system dump files.)

By using the GET command (abbreviated G), it is possible to load a dump file into your core image without starting it. This command takes the same arguments as RUN, but instead of starting the program, it types out a message indicating the size of your core image and leaves your terminal in monitor mode.

All of the commands in this section cause a RESET of your job.

## 4.2 Manipulating Core Images

The following commands refer to words in the Job Data Area (see page 11) of your core image. These words are generally set up by the monitor or the LOADER. The *UUO Manual* has a complete explanation of the Job Data Area.

The CORE command is used to find out or change the size of your core image. CORE may be abbreviated C. If you include an argument, the size of your core image will be set to that many blocks. If you do not give an argument, your core size is typed out, along with the size of your upper segment, if any; a final line gives the total amount of core available for user jobs. If you give the CORE command when you are not logged in, you get only the last line.

The command CORE Ø destroys your core image. It also causes a system RESET of your job, releasing any I/O devices you were using, etc.

The way to create core image dump files which can be loaded later is with the SAVE command. This takes device, file, and size arguments just like GET. This command also does a RESET, and it does not save your accumulators. Therefore, the program cannot be continued after a SAVE command, although it can be restarted.

Unless your core image includes DDT or RAID (JOBDDT nonzero), only the locations up to the address in JOBFF are saved. (This does not apply if you use a core size argument.) JOBFF is set by the LOADER to the highest address it loads into, so any core which was allocated dynamically by your program will not be saved unless the program updates JOBFF. I/O buffers allocated for you by the system are allocated above JOBFF, which the monitor then sets to the new highest address.

If your program includes an upper segment which you wish to save, you should use the command SSAVE. This is just like SAVE in other respects. SAVE never saves upper segments.

It is possible to examine and alter words in your core image individually by monitor commands. The command E (examine) with an octal argument will type the contents of the specified address in octal. The DE (deposit) command takes three arguments in octal: the left and right halves of the word you want to deposit, and the address. DE without the address argument uses the address of the last E or DE command. E with no argument examines the location following the last one used by E or DE. (Note that the DDT and RAID debugging programs provide a much more versatile way to do this.)

## 4.3 Starting Programs

The following commands refer to words in the Job Data Area (see page 11) of your core image.

These words are generally set up by the monitor or the LOADER (except for JOBREN). The *UUO Manual* has a complete explanation of the Job Data Area.

The START command (abbreviated S) is generally used to start a program at its normal starting address in JOBSA. This command can also be used with an octal argument specifying some other address.

The REENTER command starts the program at the reenter address specified in JOBREN, if there is one. JOBREN must be set by the program if it wants to use this capability. Program documentation will explain the use of the REENTER command for a particular program.

The DDT command starts the program at the address in JOBDDT, if any. This address is set by the LOADER if either DDT or RAID (the debugging programs for Teletypes or displays) is loaded with your program.

When a program is stopped by typing CALL, by a UUO in the program, or by various other conditions, the address of the next instruction to be executed is stored in JOBPC. If the job is continuable (not stopped because of an irrecoverable error), the CONTINUE command will start it at that address. (The START, REENTER, and DDT commands will work even if the job is not continuable, except for certain errors which destroy the core image irretrievably.)

All of the above commands place your terminal in user mode, that is, anything you type will be processed under control of your program. It is also possible to start a program, but leave the terminal in monitor mode. In this case you can still enter monitor commands which do not affect your core image. Monitor commands which are illegal while your program is running will give the message PLEASE TYPE ↑C FIRST.

The CSTART command is like START but leaves your terminal in monitor mode. Like START, it will take the desired start address as an argument or use the contents of JOBSA. The CCONTINUE command is the monitor mode equivalent of CONTINUE. There is no such equivalent for REENTER or DDT. With the terminal in monitor mode, the program can still type output on it, but an attempt at reading from the terminal will make the job wait (in IOWQ) until the terminal is placed in user mode.

The HALT command stops your program. The monitor converts the CALL key into this command, so you should never need to type it explicitly. (The only time you can type it explicitly, of course, is if you started the program with CSTART or CCONTINUE.)

## 4.4 Detached Jobs

Normally, any user job is associated with a particular terminal. However, it is possible to *detach* the job so it can continue running while you do something else (another job) at your terminal. Also, the monitor sometimes starts up detached jobs itself to perform various system functions.

Detached jobs can run, and do all the things attached jobs can do, except that if they try to output to device TTY, the output is lost; attempted input from TTY makes the job wait indefinitely (until it is attached). If a detached program executes an EXIT UUO, the job is killed.

The DETACH command detaches your job, leaving your terminal in monitor mode and not logged in. You can then log in again without affecting the old job. DETACH takes no arguments. Several commands described below combine the DETACH function with other useful things. Please remember not to leave jobs detached forever, but to log them out eventually!

The ATTACH command is used to attach your terminal to a detached job. If you are already logged in, your old job is detached. The ATTACH command takes two arguments, a job number and the PPN under which that job is logged in. (If you are already logged in with the same PPN, you need not give the second argument.) The job must be detached before you give the command. Note that if you were displaying a WHO line for your old job, it will still be displayed, indicating the job as detached. This is handy for re-attaching it when you want to, in that it shows the job number. (Typing ESC W again will, of course, display the WHO line for your new job.)

The FORK command will detach your old job and log in a new one. It is faster than DETACH followed by LOGIN, because it does not run the LOGIN program. Instead, it merely copies the system information from your old job into the new job. In particular, if you have an alias in your old job, it is copied to the new job as well.

Often you want your program to continue running in the old job while it is detached. You could say CCONTINUE and then DETACH, but the command CDETACH combines these functions. There is also a CFORK command which combines CCONTINUE and FORK.

## 4.5 Device Control

Certain I/O devices can only be used by one user at a time, such as a magnetic tape unit. In some cases, like the line printer, the system provides a facility for queueing print requests so that individual users need not actually control the device themselves. However, when a user does need to use such a device, the ASSIGN command is used to ensure that only one user at a time tries to use the device. The ASSIGN command (abbreviated A) can also be used to cause a *logical device name* to be associated with a particular device. Thereafter, any reference by your program to the logical name will be translated into the corresponding physical device.

The command takes two arguments, a physical device name, and an optional logical name. The physical name can be *generic*: the command

```
ASSIGN MTA
```

will select an available mag tape unit (MTA0 or MTA1) if there is one available. You can also select a particular unit explicitly.

The logical name feature is sometimes used by programs to allow you to control their I/O without having to type instructions to the program itself. For example, a program which produces a listing output might direct it to a particular filename on device name LST. Before running the program, you could type

```
ASSIGN DSK LST
```

to write the listing file on the disk, or

```
ASSIGN TTY LST
```

to type it out at your terminal. You can also change the meaning of what is normally a physical device name this way—the command

```
ASSIGN DSK LPT
```

will make a program which was written to write its output on the line printer use a disk file instead. You might do this if the line printer is not available when you want to run the program.

Once you have assigned a device with the ASSIGN command, no other job can use it until you release it with the DEASSIGN command. This command, abbreviated D, takes either a logical or a physical device name as argument. (Note—Assigning DSK or SYS does not prevent other users from using the disk.) It is possible for a program to use a device which is not assigned, but the device is released in that case as soon as the program stops using it. The DEASSIGN command with no argument will deassign all devices assigned by your job.

The job which has a device assigned may give it to another job with the REASSIGN command. This command takes two arguments, the device name and a job number. This can occasionally be useful if, for example, one job writes a tape which another should then read. The REASSIGN command prevents the possibility of another job grabbing the tape unit before the intended recipient does.

The FINISH command (abbreviated F) is like DEASSIGN except that it releases the device from your program as well as from the ASSIGN command assignment. If the device is a directory device which your program had open for output, the file is closed so that as much data as the program wrote will be saved. (This applies in the case in which the program was stopped before its normal exit, either by CALL or by an error.) F with no argument releases all devices.

The FLUSH command is designed to allow a terminal which is not in use but somehow has something in its input or output buffer to be emptied. It takes a device name as argument; the device must be a terminal which is not in use, or your own terminal. This command may be used without logging in.

## 4.6 System Information

Several commands exist to find out various things about your job or others. Some of these duplicate information available on the WHO line at display terminals. None of the commands in this section affect your core image.

The PJOB command types the job number of your job, if you give the command with no argument. If you use a device name as the argument, it types the number of the job using the device. (If the device is not in use, it says so.)

The PPPN command prints out the logged in PPN and the alias, if any, for your own job (no argument) or the job number you give as argument.

The PTTY command prints out the terminal line number of your own or another job. If the terminal is a DataDisc, it also prints the channel number. The TIME command, which also takes an optional job number argument, types out five quantities for the specified job; the first four are times, in hours:minutes:seconds'ticks (a tick is one sixtieth of a second):

| | |
|---|---|
| TOTAL | is the total run time for the job. |
| INCREMENT | is the run time since the last time a TIME command for this job was given by the job itself. That is, the INCREMENT time is reset when you give a TIME command for your own job, but not when you give one for another job. |
| XTIME | is the same as the XTIME displayed on the WHO line--it is reset whenever the job is started by a command other than CONTINUE, and whenever the program is returned to the RUN queue after waiting for terminal input. This meaning of XTIME can be changed; see page 18 for a complete discussion of XTIME. |
| WAIT | is the time the job has had to wait for the system, including time spent in RUNQ and disk I/O wait, but not tty I/O wait, STOP, etc. It is somewhat of a measure of the quality of service the job is getting. |
| KCS | (kilo-core-seconds) is a measure of the demands the job has made on the system. It is incremented by the job's core size (in K) every second of runtime. |

The command TIME 0 types the time since the last system reload, the percentage of that time spent in core shuffling, and the percentage spent running the null job (what the system does when there are no real jobs it can run).

The DAYTIME command with no argument types the current day of the week, date, and time. With a job number argument, it types the time that job logged in, the time it was last run, and the current time. DAYTIME 0 specifies your own job.

The RESOURCES command lists the available system resources. It types the number of free disk blocks, the number of Data Disc channels free, the number of job slots available, and the names of free devices (mag tape, etc.) other than the disk and terminals.

The FILES command lists disk files in use by a job, with some status information. It can take a job number argument (0 means your own job), in which case all files in use by that job are listed, or a filename argument, in which case the status of the specified file is displayed (if it is being used by any job). Each line typed by this command contains the job number of the job using the file; the filename, extension, and directory ppn of the file; the number of records in the file; the number of the record currently being read or written; and the way the file is being used: R for read, W for write, or RA for read-alter, possibly followed by E for end-of-file seen. The FILES command with no argument is normally equivalent to FILES 0; however, if a FILES command results in more information to type than fits in the output buffer, the command lists as much as will fit followed by a line with an ellipsis (...), and a FILES command with no argument will continue the list from that point.

The HELLO command types the name of the current version of the monitor.

All of the above commands are legal when you are not logged in. The following one is not.

The SLEVEL command types out your current service level reservation. The service level system attempts to guarantee the reserved percentage of the computer's time to programmers who make such reservations. See the RSL command, in Appendix 6.

## 4.7 Miscellaneous Commands

The TTY command can be used to change several characteristics of your terminal. It takes as argument a keyword specifying the thing to change. As in typing monitor commands, only enough letters of the keyword as necessary to make it unique need be typed. The keyword may be preceded by NO or - to reset the corresponding switch. The keywords are

| | |
|---|---|
| TABS | declares that the terminal has hardware tabs, so the monitor will not convert tabs to spaces on output. |
| ECHO | tells the monitor to send back to the terminal the characters you type in. |
| FILL | tells the monitor to insert extra carriage returns when a return is typed out at the terminal, to give the carriage time to return to the left margin. This is used principally for 30 character per second terminals. |
| FULL | declares that the terminal has lower case letters, and lower case should not be converted to upper case by the monitor. Equivalent to ESC F on a display; NO FULL is like BREAK F. |
| UPDATE | enables automatic resetting of XTIME, like ESC X from a display. NO UPDATE is like BREAK X. |
| TIME | resets XTIME and inhibits automatic resetting, like ESC 1 X. NO TIME, like BREAK 1 X, sets XTIME to the total run time. |
| WHO | types out your job's WHO line. "TTY WHO n" will type out the WHO line for job nj "TTY WHO 0" or "TTY -WHO" will type out the system WHO line. |

The TABS, ECHO, and FILL keywords have no effect at display terminals. You need not be logged in to give the TTY command, except for the UPDATE and TIME functions.

The ALIAS command is used to set your job's *disk PPN* or alias. If you have an alias, all disk references by your job which do not include an explicit PPN will use the alias directory rather than your logged-in PPN directory. The argument can be PRJ,PRG to set the alias to that PPN, or just PRJ to use your own programmer name in the alias. No argument resets your alias to zero, so your logged-in PPN will be used for disk files. If you have an alias, your logged-in PPN is still checked for file protection purposes.

The TALK command can be used to talk to a user at another terminal. It takes a device name as argument. The device must be a terminal, which must be in monitor mode and must not have any characters in its input buffer. (Device CTY, the PDP-10 console Teletype, is never busy; TALK CTY always succeeds. Note to remote users: There is no operator on this system; TALK CTY is not a good way to get assistance!) If the command succeeds, all characters you type are typed out on the other terminal, and vice versa. If the terminal you specify was already in a talk ring, you are added to the ring; all characters typed at any terminal in the ring appear on all the others. You leave the talk ring by typing CALL. You need not be logged in to enter a talk ring.

The KILL command can be used to kill another job with your programmer name. It takes the job number as its argument.

# SECTION 5

# SYSTEM PROGRAM COMMANDS

The remaining monitor commands all run particular system programs in your core image. There are a few main groups of these commands and a few miscellaneous ones. These commands are documented along with the programs they run. The next two sections are about the LOGIN and LOGOUT programs and the commands which run them; these were briefly introduced earlier. The major command groups are then presented, and finally the miscellaneous system programs. Many system programs are not run by special monitor commands, but by the R command. Those programs are not documented here, but in separate manuals.

## 5.1 LOGIN

The LOGIN command is used to begin a session of using the computer. It runs the LOGIN program, which provides several optional services as well as setting up the necessary system tables for your job.

The LOGIN command may be abbreviated L. It takes one argument, a project-programmer name. Different characters used between the two parts of the name are used for different options:

```
PRJ,PRG     types all system messages and processes OPTION.TXT (see below)
PRJ/PRG     types system messages new since last login, processes OPTION.TXT
PRJ.PRG     types no messages, ignores OPTION.TXT; for fast login
PRJ%PRG     sets new password, as explained below, then acts like PRJ,PRG
```

System messages are notices for all users, sent by the MAIL * command and stored in the file NOTICE.TXT[2,2]. There may also be messages addressed to a particular project, to a particular programmer, or to a PPN. Mail to your project is treated like system messages. The handling of programmer or PPN mail is explained below.

It is possible to set a password to restrict logins on a particular PPN. If there is a password, LOGIN types PASSWORD= and you must type the correct password to log in. Echoing of input characters is turned off so the password does not print. Passwords may be one to six letters or digits followed by RETURN. You can set a new password by using % in the LOGIN command. If there is already a password, you must type it in. LOGIN types out NEW PASSWORD= and you can enter a new one; if you type RETURN with no new password, you will no longer have a password. Passwords are checked by the COPY program for access to other users' files, but there are lots of programs that don't check. If you have any information that you consider secret, you are advised to take it elsewhere. The A.I. Laboratory is a research establishment. There are facilities in the monitor which provide valuable research tools; these facilities may be misapplied to circumvent passwords and file protection. The protection that does exist is for accident prevention, not for secrecy. Users are assumed to be responsible persons.

If you log in from a remote terminal (by a dial-up telephone line), you will be asked for a *remote users password* which is used to discourage logins by unauthorized outsiders. If you need to log in remotely, ask someone. Sometimes you may find that the system is in *maintenance mode*, which means that the monitor is being debugged and the system is not available for normal use. If you try to log in at such a time you will be asked for the maintenance password.

The further details of LOGIN's operation are controlled by a file named OPTION.TXT which you may have in your directory. If there is one, it is searched for a line of the form

```
LOGIN:opt1,opt2,opt3;comments
```

where opt1...opt3 are the desired options. Spaces may be used in the obvious places. Only the first six letters of an option name are read. Upper or lower case is ok. Several other programs use this OPTION.TXT file, looking for their own option lines. The LOGIN options are as follows:

```
ME          Tells you your fortune.
MESSAG      Types your mail without asking; see below.
LOGRUN      Runs the LOGRUN program, which executes monitor commands from a
            LOGRUN entry in OPTION.TXT; see below for details.
WHO         Starts a WHO line automatically if you are at a display terminal,
            as if you had typed ESC W.
UNHIDE      Makes your Data Disc channel public, like BREAK H.
FULL        If you are at a Teletype, sets the full character set mode switch.
            Equivalent to TTY FULL.
TABS        If you are at a Teletype, clears the tab expand bit; tabs will
            not be converted to spaces on output.  Like TTY TABS.
FILL        If you are at a Teletype, sets the fill switch to insert extra
            carriage returns on output to give the carriage time to get all
            the way back.  Like TTY FILL.
DIGEST      Tells you if there is a new Associated Press news digest that came in after
            the last time you logged in and asks you if it should type it out.
PORNO       Try it and see.  Doesn't work if you also have LOGRUN.
NOMAIL      Avoids being asked questions about message files.  This is for
            people who like to type ahead while LOGIN is running.  The
            exact effect depends on what other options are used; see below.
AUDIO=n     Selects audio switch input channel n to your speaker if you
            are at a display terminal.  "n" is an octal number.
            "AUDIO=-n" inhibits telephone paging interrupts, like BREAK n U.
```

If there is a message addressed to your programmer name or to your PPN, LOGIN normally types

```
THERE'S A NOTE FOR prg                    (or prjprg)
READ IT NOW?
```

If you type Y, the message file is printed, and you are asked if you want to delete it. (You can also type R, which will treat the mail like a system message file, i.e., LOGIN doesn't type header lines, and if you used a slash in the command it only types messages which came since you last logged in. In this case it does not ask if you want to delete the file.) Note: Mail is normally sent to programmer names, not PPNs. Some people send PPN mail to themselves as reminders.

If you have the MESSAG option but not the NOMAIL option, then mail is always typed out without asking first. If you have both MESSAG and NOMAIL, the mail is typed but you are not given the chance to delete it. If you have NOMAIL alone, you get the THERE'S A NOTE... line but not the mail itself. (See the RCV command for another way to process your mail.)

If you have DIGEST and NOMAIL, you are notified of a new A.P. digest but it is not typed out.

(RCV will also do this for you.) It is also possible to get automatic notification of incoming A.P. stories on particular topics from the APE program; these messages are treated like programmer mail with respect to MESSAG and NOMAIL.

The LOGRUN program looks for a line in your OPTION.TXT file starting with LOGRUN:. It then takes that line, and all following lines until a semicolon is seen, and makes them be executed as if you typed them in. (The semicolon is required.)

## 5.2 LOGOUT

The command for terminating a session on the computer is KJOB (kill job), which runs the LOGOUT program. The command may be abbreviated K. It takes an optional switch argument to select various optional features. The OPTION.TXT file is also used to control logout options.

The LOGOUT program normally types out several lines of accounting information, e.g., computer time used. It deletes any files in your directory with extension RPG (these are the files the RPG system uses to remember the commands you gave), and if there are no files in your directory, it deletes the directory itself.

If you have the RPGSAV option (see below), LOGOUT will look for TMPCOR files (simulated files in core storage) used by some processors for remembering RPG commands instead of disk files, and copy them to the disk so they will be remembered the next time you log in. Note that this will not be done if you use the /F switch (below) for fast logout. Note also that LOGOUT does not save TMPCOR files created under an alias, even if you are still aliased when you give the KJOB command. (The monitor resets your disk PPN to your logged-in PPN before running LOGOUT.)

You are told if you have another job logged in with the same PPN, and if you have assigned any private devices (mag tape, etc.) you are reminded to unload them.

This normal processing can be affected by switches in the command itself or by options in the OPTION.TXT file. LOGOUT looks for a line in OPTION.TXT of the form

```
LOGOUT:RPGSAV,ME,FAST;
```

(Of course, you need not use all the options.) These options, if found, have the following effect:

```
RPGSAV      Don't delete RPG files.
ME          Type a friendly message (try it).
FAST        Just log out, don't delete anything, don't print statistics.
```

Switches are single letters preceded by a slash (e.g., KJOB/F). The switches are:

```
/K          Kill RPG files (only necessary if you have the RPGSAV option.)
/S          Save RPG files.
/M          Type a message, like the ME option.
/F          Fast logout.
/Z          Zero the disk.  Deletes all your files, after asking if you
            really mean it.
```

There are two other monitor commands which run the LOGOUT program, for logging in with another PPN, and for attaching to an existing job (see the ATTACH command, Section 4.4.) They are

```
KLOG logout-switches ppn
KATTACH jobnum ppn logout-switches
```

The logout switches are as described above. The other arguments are in the required form for the LOGIN and ATTACH commands.

Finally, there is a LOGOUT command, which is exactly the same as KJOB but no arguments are allowed.

## 5.3 RPG

The RPG commands have already been mentioned. They run the RPG program, which then runs various other system programs. The commands are divided into two basic groups. The first, for text file editing, take a file specifier as argument:

```
CTV         create a file with TV
TV          edit a file with TV
CREATE      create a file with SOS
EDIT        edit a file with SOS
MAKE        create a file with TECO
TECO        edit a file with TECO
```

The other major editor, E, includes the necessary code to carry out the RPG functions itself, e.g., default extensions and remembering the commands. The following commands are, therefore, not strictly RPG commands but function equivalently:

```
CETV        create a file with E
ETV         edit a file with E
```

The second group, for program compilation, take one or more program names, along with various option switches:

```
COMPILE     compile programs
AD--        compile and load programs
EXECUTE     compile, load, and run programs
PREPARE     compile and load with a debugger
DEBUG       compile, load with a debugger, and start the debugger
TRY         compile, load with a debugger, and start the program

PUB         produce a document with the PUB Document Compiler
```

Finally, one more command, used in conjunction with the compile-type commands, which takes no arguments:

```
CREF        produce cross-reference listings from compilations
```

The complete description of these commands, with the command syntax and the processing options provided, is included as Appendix 1.

## 5.4 COPY and SPOOL

The next major group of commands runs the COPY program. Their arguments are generally in the form "new-file ← old-file":

```
COPY        copy a file
TRANSFER    copy and delete the original
RENAME      rename a file or change protection
```

The following COPY commands imply a destination, and take just one or more input file specifications (separated by commas if more than one):

```
DELETE      delete a file
TYPE        copy to the terminal
LIST        copy to the line printer (SPOOL, below, is the preferred technique)
PRINT       copy to the line printer with different formatting (see Appendix 2)
XGPLIST     copy to the Xerox Graphics Printer (XSPOOL is preferred)
DIRECTORY   type file directory
HELP        explain system programs
```

The COPY manual explains these commands fully, along with various options specified in the argument list; see Appendix 2.

Since only one job can use the line printer or XGP at a time, if several people want to make listings at once there is a scheduling problem. To solve this, the system allows listing requests to be stored on disk, and processed one at a time by *spoolers* which actually control the LPT and XGP. The system commands for this function are:

```
SPOOL       request line printer listing of specified file(s)
XSPOOL      request XGP listing of specified file(s)
UNSPOOL     delete a spooler request
QSPOOL      type out spooler status and queues
```

The SPOOL commands also allow editing options to be specified in the argument list; see Appendix 3 for details.

The HELP and QSPOOL commands may be given without logging in; the others in this section require login first.

## 5.5 MAIL

Several commands are provided for sending messages to other users. Messages can be filed on disk, so that the addressee will be notified about them by LOGIN, or they can be sent directly to the terminal of a logged-in user. There is also an automatic reminder system which allows delayed messages. The commands for processing messages are:

```
MAIL      send a message to a user's message file
SEND      send a message to a user's terminal
REMIND    request a delayed message
GRIPE     send a message complaining about a system problem
RCV       edit message files
LATER     request delayed execution of a program
CANCEL    delete REMIND or LATER requests
```

The usual format of the MAIL and SEND commands is

```
MAIL user message
SEND user message
```

where user is a programmer name and message is the text you want sent. If you do not include a message in the command, the MAIL program will ask you to type a message, which may then have more than one line. The complete description of these commands is in Appendix 4.

The RCV command may be given without logging in; the others require login first.

## 5.6 DART

DART is a program to save disk files on magnetic tape and restore them as needed. It also handles mag tape positioning commands. The file dumping commands take disk file names as arguments:

```
DUMP      dump disk files onto tape
RESTORE   restore files from tape to disk
LOCATE    find which tape has dumped copy of files
TLIST     list all files on a tape
```

The tape positioning commands are:

```
REWIND
ADVANCE
BACKSPACE
EOT
```

The complete documentation of these commands is in Appendix 5 of this manual.

## 5.7 DO

The DO program allows automatic execution of an arbitrary sequence of commands. By writing DO command files, you can make your own sequences of program execution analogous to the RPG compiler-loader-execution sequence. The DO command takes a file specifier as argument. The text in the file is simply transferred into your terminal's input buffer, as if you had typed it. Then the DO program exits, and the monitor processes the commands from the file. The DO program cannot be run by detached jobs. The amount of text allowed in the DO file is limited by the capacity of your input buffer.

DO does some conversion of certain characters in the file, to allow things like CONTROL and META characters. The characters which are processed specially are:

```
RETURN      ignored
LINE        ignored
↵           translated to RETURN (the monitor will supply a LINE as well)
↓           translated to LINE
≠           translated to ALT
β           translated to deferred CALL
VT          adds CONTROL bit to the following character
FORM        adds META bit to the following character
∞           translated to ESC
∞-          translated to BREAK
|           separates different DO functions (see below)
≡           quote the next character (do not process it specially)
?           takes the next character (other than RETURN or LINE) as a
            variable name.  Suppose the character is A (i.e., ?A).  If
            this is the first occurrence of ?A in the file, DO types out
            "?A    " and waits for you to type in a text string ending
            with RETURN.  This string is substituted for every occurrence
            of ?A in the file.
```

To allow more than one DO function to be stored in the same file (because small files are inefficient in using disk space), the vertical bar (|) character can be used as a separator. The command

```
DO DOFILE(3)
```

will DO the commands between the second and third | characters in the file.

The DO command remembers its argument in an RPG file, so you can type DO without an argument to repeat the same command file. Also, the DO command is remembered as if it were a COMPILE-type RPG command, so the RPG exit mode commands in the text editors will also repeat the DO command. Thus, users of languages like LISP which are not recognized by RPG can write DO files to run LISP and read in their program.

## 5.8 ZERO

The ZERO command is used to initialize file directories on disk or DECtape. The command takes a device name as argument; the device must be DSK, DTAn, or UDP (user disk pack).

For DECtape, ZERO simply executes the UTPCLR UUO, which tells the monitor to initialize the file directory for the DECtape unit you specified. Make sure you don't get the wrong unit number and zero someone else's DECtape!

If you specify device DSK, you must confirm (by typing Y) that you want to delete all the files in your disk area. ZERO then attempts to delete each file in your area. For each file, it types # if successful, otherwise a message indicating the reason for the failure (write protected, etc.).

For the UDP, you must give the write password when asked. Then you are given the opportunity to initialize a directory on the pack. (Note: the monitor does not maintain a file structure on the UDP. The directory format on UDPs is maintained by the COPY program to simulate a disk directory.) Generally this is only done once. The other thing you may do with this command is change the write password for the UDP. If you answer Y when asked if you want to do this, you are asked for the new password, which is written on the UDP.

## 5.9 WHO

The WHO command (abbreviated W) runs the WHO program, which displays system status information at your terminal. If you are at a display terminal, the information is automatically updated as long as the program continues to run. If you are at a Teletype, the information is only typed once. The WHO command does not require that you be logged in.

The first part of the WHO display is a list of the jobs on the system, with various information about each job. This list is divided into two sections, for jobs belonging to users and for system *phantom* jobs. (More precisely, the second section contains jobs which are detached and have the JLOG bit off in the job status word, i.e., are not logged in.) Jobs attached to pseudo-teletypes are listed under the controlling job, with the line for the controlled job indented. The listing for each job has several parts:

```
JOB         job number
QUEUE       queue name, as in the WHO line, with possibly an extra character
            indicating one of several states (if more than one applies, the
            one which comes first in this list is displayed):
            -   the job is now actually running
            *   the job is locked in core
            →   the job is an upper segment, and is next in line to swap in
            ↑   the job is next in line to be swapped out
            ↓   the job is next in line to be swapped in
            ✲   the job is swapped out
            ↔   the job is next in line to be moved in core
PPN         the job's logged-in PPN
            (This will be 100100 for not-logged-in WHOs, etc.)
LINE        the job's terminal line number, or DET for detached jobs
JOBNAM      the job name
SIZE        the job's core image size in K (1024-word blocks)
TIME        the job's total run time since login
PL          percentage of CPU time spent on this job recently
SL          service level reservation for this programmer
SEG         job number of this job's upper segment
SW10        number of ticks between startups of job's PDP-10 spacewar module
SW6         number of ticks between startups of job's PDP-6 spacewar module
```

After the job display comes a similar display for upper segments, containing the job number, job name, core size, and number of jobs using the segment.

The next part of the WHO display is a summary of overall system statistics. This includes the time since the system was last restarted (UPTIME); short and long term time spent running the null job (NULTIMES); short and long term time spent running the null job when another job wanted to be run but wasn't in core (WASTED); total user core image sizes in core and swapped out (CORE); the largest contiguous free block of core, the total available user core not used by locked-in jobs, and the total available user core (USABLE); the number of jobs in RUNQ and TQ and their combined core image size (RUNNING JOBS); and the total service level reservation for logged-in users (TOTAL SLEVEL).

The final section of the WHO display is a list of all I/O devices in use by programs or assigned by jobs. For each such use of a device (at most one per device except for DSK), a line is displayed containing the logical name, if any; the physical name; the character "*" if the device was assigned by the ASSIGN command; the job number using the device; if a particular file is open, the file name, extension, and directory ppn (for disk files), the number of records in the file, the record currently being read or written, and the read/write status. This last will be W if the file is open for output, R for input, or RA for read/alter mode, followed by E if the end of file has been read. Note: if the file has been closed, there may be no mode flag, and a large number like 1101 in the current-record position. Record numbers are displayed in octal.

On display terminals, only a part of this display can fit on the screen at any time. Single-character commands can be typed to WHO which provide "scrolling" of the display as well as other functions. The following commands are understood:

```
↓       scroll down 1/2 screen size
↑       scroll up 1/2 screen size
W       run forever (otherwise WHO exits after nothing has been typed in
        for two minutes)
R       only display jobs which have recently been Running
        (not in NULQ, STOP, IOWQ, or INTW)
M       only display My jobs (login or alias programmer name must agree
        with user's login or alias programmer name)
N       restore Normal display (all jobs)
E       Exit, leaving information displayed on the screen and with the
        page printer set up to avoid erasing it
1-9     repeat argument for ↓ or ↑
```

Any other character will make WHO restore the page printer to normal and exit.

## 5.10 FIND

The FIND command is used to locate information in a file by searching for a keyword and typing out the "paragraph" in the file containing the key. The main use of this program is to look people up in the AI Lab telephone directory. FIND may be run without logging in first.

The syntax of the FIND command has the following form (brackets denote optional things):

```
FIND [EXACTLY] key [ [IN] [file]]
```

If file=PHONE or is omitted entirely, the AI Lab telephone directory is read. Otherwise, the specified file is read.

If a key contains any spaces or tabs, it must be separated from the file name (if present) with the word IN, or delimited by " characters or the quote pair ' ... '. A ' or ' character may appear in the former, and a " may appear in the latter. If the number of spaces and tabs in the key is significant (in EXACTLY mode), the quoted form must be used. See the examples.

The file is logically divided into *paragraphs*. A paragraph consists of twenty or fewer consecutive non-blank lines on one page. FIND searches the file for paragraphs that contain the key. Each paragraph containing the key is a *hit* and is typed out. Also, the hits are counted and the count is printed at the end.

The EXACTLY qualification causes FIND to search for an *exact match*. Otherwise it performs a *template match*. A template match recognizes two special symbols in the key: *space* (one or more consecutive spaces) and *ellipsis* (three or more consecutive dots). A space in the key matches zero or more spaces, tabs, and carriage returns in the paragraph. An ellipsis in the key matches zero or more arbitrary characters in the paragraph. Furthermore, in a template match, upper and lower case letters are equated. The EXACTLY search is somewhat quicker.

Examples:

```
FIND john...lathrop<cr>

*McCarthy, John (Prof. John)              208       P[MTC*,JMC] 2 800
*       846 Lathrop Dr., Stanford, (9) 321-7580

1 HIT ON KEY = john...lathrop
```

```
FIND BRIAN HARVEY
```
will look up "BRIAN HARVEY" in the phone directory unless you have a file named HARVEY, in which case it looks up "BRIAN" in HARVEY.

```
FIND BRIAN K. HARVEY
```
looks up "BRIAN K. HARVEY" in the phone directory in either case.

```
FIND 'BRIAN K.' HARVEY
```
or
```
FIND BRIAN K. IN HARVEY
```
looks up "BRIAN K." in HARVEY.

```
FIND BRIAN IN
```
looks up "BRIAN IN" in the phone directory unless you have a file named IN, in which case it looks up "BRIAN" in IN.

## 5.11 Other System Information Programs

This section describes other commands which run information programs. All of them are allowed to be run without logging in.

The WHERE command types out information similar to WHO for jobs belonging to a particular user. It takes a PPN or programmer name as argument, and for each job whose login or alias PPN matches the argument types out the job number, PPN, job name, core size, queue, total run time, actual time when the job was last run, terminal, and alias if any. (The terminal is listed by type and unit number, e.g., TTY5, III23, DD56, PTY121. Note that these are not actual physical device names recognized by the system, except for TTYn.)

If you are logged in and run WHERE, you can give the REENTER command and type in a job name. WHERE will then list all jobs with that job name.

The FINGER command also lists jobs belonging to a particular programmer, but provides different information: the job number and job name, the programmer name, the real name of that programmer if known to the system, the number of minutes since the job was last run, the terminal number (and type, as in WHERE), and the actual location of the terminal (room number, if in the lab, and whose office it is). FINGER with no arguments displays all jobs which are logged in.

The HELP command provides information about system programs. It takes a program name as argument, and runs the COPY program, which types out the file NAME[3,2] where NAME is the argument. HELP with no argument lists the available HELP files. HELP files are generally very short descriptions which refer you elsewhere for complete documentation.

The SYSTAT command types out a system status description similar to that of WHO. The WHO version is better, and SYSTAT is rarely used.

## 5.12 Miscellaneous System Programs

The commands in this section run miscellaneous system programs. Except for FIXIMLAC, they require that you be logged in.

RSL runs the RSL program, which is used to reserve service level (i.e., better service). The command takes no arguments. The meaning of service level and the use of RSL are described in Appendix 6.

The LISP command runs the LISP interpreter. It takes no arguments. Note: when the system wizards need a command table slot to patch in a new command, LISP is the first to go. Therefore, it may not always exist. R LISP has the same effect and always works.

The FIXIMLAC command is used to initialize IMLAC terminals. (These are terminals which are treated as Teletypes by the monitor, but are actually small computers with display screens.) With any luck, you will never see an IMLAC. Just in case, the command is explained in Appendix 7.

The XEROX command runs the XEROX program, which does file formatting to produce pretty documents. It has nothing to do with the Xerox Graphics Printer. XEROX has been largely replaced by PUB, a more versatile formatter. The XEROX manual is the file XEROX.LES[UP,DOC].

The TELNET and TN commands (TN is used to abbreviate TELNET) run the T program, for talking to another computer on the ARPA network. The site name of the computer you want to use is the argument. If you end the command line with ALT instead of RETURN, the program will attempt to connect to the remote computer even if the monitor thinks it's down.

The FTP command takes an ARPA site name like TELNET, but runs the FTP program to carry out the File Transfer Protocol for copying files between computers.

Further details on T, FTP, and the ARPA net in general is contained in Appendix 8 of this manual.

# SECTION 6

# PRIVILEGED COMMANDS

Certain command functions can only be used by privileged users. (There are several privileges which may be associated with a PPN.) Most privileged functions are carried out by programs and are documented with those programs, but a few monitor commands themselves take on special meanings when used in a privileged way. Most users do not require any privileges; if you think you need to be assigned any, see a system programmer, who will disagree.

The DETACH command, if given with a device name argument by a user with the DEV privilege, makes the specified device unavailable to users. This is intended to be used for hardware maintenance. The ATTACH command with device name argument, also privileged, assigns a detached device to the job giving the command; it becomes generally available when the job deassigns it. These commands also accept pseudo-device names of the form DDn, to make Data Disc channel n unavailable.

The KILL command can be used to kill any job by a user with the KIL privilege.

The TALK command is always successful, even to a terminal in user mode, if given by a user with the TLK privilege.

The EDDT command can be given only from the PDP-10 console Teletype. It stops timesharing and starts Executive DDT to allow debugging of the monitor.

# APPENDIX I

## RPG

RPG, the Rapid Program Generator, is a program which interprets certain monitor commands for editing, compiling, and loading programs. It simplifies these tasks by expanding concise user commands into the different forms used by editors, language processors, and the LOADER. Without user intervention, RPG will run the correct language processors to translate the user's files. It will run the LOADER to create an executable core image containing the user's program. The monitor provides several commands that run RPG.

This appendix is excerpted from the RPG manual, SAILON 51.1, on the disk as RPG.REG[S,DOC]. It does not include the section on the internal workings of RPG.

RPG knows about certain standard processors. These are the editors: TV, SOS, and TECO; the compilers: SAIL, FAIL, F4 (FORTRAN), MACRO, OSAIL (Old SAIL), NSAIL (New SAIL), and PUB (Document Compiler); the debuggers: RAID and DDT; and the LOADER.

The E editor includes within it the RPG functions, so its "RPG" commands run E directly rather than via the RPG program. Nevertheless, the E commands are included here because they are operationally similar.

The RPG commands are divided into three classes, EDIT, COMPILE, and CREF.

### EDIT-Class Commands

There are eight EDIT-class commands, two for each of the four commonly used editors:

| | Command Name | |
|---|---|---|
| Program | Create File | Edit File |
| SOS | CREATE | EDIT |
| E | CETV | ETV |
| TV | CTV | TV |
| TECO | MAKE | TECO |

CREATE, CTV and MAKE

These commands take a file name as the argument. RPG will call the editor and tell it to initialize the file of that name. For CREATE, SOS is started in line insertion mode. The TV editor will initialize the file and allow the insertion of text on page 2. TECO (the MAKE command) will be initialized to write on the specified file.

## EDIT, TV and TECO

These commands take a file name (with optional project-programmer name) as the argument. RPG remembers the name of the last file that was edited (except that LOGOUT makes RPG forget). If an EDIT-class command is given without an argument, RPG will use the argument that it remembered.

If no extension is specified in the argument and no file with the given name and blank extension can be found, then the user's directory is searched for any file with the given name and one of several standard extensions. The standard extensions are searched in the order: NSA, OSA, SAI, FAI, MAC, F4, LST, WRU, PUB, and CMD.

If a user specifies a disk area other than his own current (logged in or aliased) area, then the edited file will be put on his disk area and the source (from the other area) will not be changed.

The editors have a read-only mode in which files may be examined by using editor commands, but not changed. To get read-only mode type /R after the file name in the edit command (e.g., TV M.SAI/R). Read-only mode in SOS prohibits any commands that would change the file. In the display editors, the mode may be changed from read-only to read-write at any time. In TV, there is a no-directory mode, specified by the switch /N.

## CETV and ETV

These are the create and edit commands for the E editor. They do not run the RPG program; instead, E itself simulates the RPG functions. There are several differences between these and the other editor commands:

1. E saves its edit commands not on the disk, but in a TMPCOR file, i.e., a simulated file kept in core by the monitor until the job is logged out. The LOGOUT program copies these files to the disk of you use the RPGSAV option.

2. The command can include /nP and /nL switches for page and line numbers as well as /R and /N. E saves its position in the file when it exits, so a later ETV command with no argument will continue the edit from the same point in the file.

3. E allows you to edit a file in another dsk area by including the PPN in the command, rather than copying the file to your own area as RPG does for the other editors. Also, since TMPCOR files belong to a job rather than a PPN, changing your alias does not change the saved edit command which E finds.

## COMPILE-Class Commands

The COMPILE-class commands are an abbreviation for a series of commands to compilers (and other language processors) and the LOADER. Here is a brief summary of the commands:

| | REL file | Core Image | Debugger | Execution |
|---|---|---|---|---|
| COMPILE | yes | | | |
| LOAD | yes | yes | | |
| PREPARE | yes | yes | yes | |
| EXECUTE | yes | yes | | starts the user program |
| TRY | yes | yes | yes | starts the user program |
| DEBUG | yes | yes | yes | starts the debugger |
| PUB | | | | document compilation |

The COMPILE-class commands, like the EDIT-class commands, remember their argument list. If the user does not supply explicit arguments, RPG uses the same arguments that were used in the previous COMPILE-class command. Arguments are remembered in the disk files QQSVCM.RPG (COMPILE-class) and QQSVED.RPG (EDIT-class), which are usually deleted by the LOGOUT program. RPG actually remembers the last COMPILE-class command that had explicit arguments; both the command and the arguments are remembered. This is useful because all the editors allow an exit-and-go command (in SOS, the G command; in TV and E, the CONTROL-X G command; in TECO, the EG command). The exit-and-go feature runs RPG in a special way which makes RPG re-execute the command that it remembered.

PUB, the Document Compiler, is a special case: it does not translate computer programs into REL files, but documents into DOC files. The PUB command is used to compile a PUB document; the other COMPILE-class commands do not recognize PUB. The PUB command is remembered in a QQSVCM.RPG file, like the other COMPILE-class commands.

### COMPILE

The argument should be one or more file descriptors, separated by commas. RPG will cause the appropriate translator to be used for each of the files. The translator that is used is determined by the the extension of each file. The standard extensions and translators are:

| Extension | Translator |
|---|---|
| SAI | SAIL |
| F4 | FORTRAN |
| FAI | FAIL |
| blank | FAIL |
| MAC | MACRO |
| OSA | OSAIL (Old SAIL) |
| NSA | NSAIL (New SAIL) |

The decision about whether to compile is made for each file by comparing the source file date (and

time) with the corresponding REL file date. If there is no REL file, or if the text file has been edited since the creation of the REL file, then the source file will be translated again. If the REL file is up to date, no compilation is done. For example, if the files FOO.FAI, GARP.FAI, and BAZ.F4 are to be compiled, the command: COMPILE FOO,GARP,BAZ would be suitable. Only those files that have changed since the creation of their corresponding REL files will be compiled.

Note that the COMPILE command does not produce a core image. To get an executable core image one of the other COMPILE-class commands must be used.

LOAD

LOAD does a COMPILE if necessary and then starts the LOADER. The LOADER takes the REL files that were generated by the compilers and combines them into an executable core image. When the LOADER is done, it exits to the monitor. This command makes a core image but does not start it. If any SAIL program is loaded, the LOADER will be instructed to request the current SAIL segment.

PREPARE

PREPARE is like LOAD, but a debugger (RAID or DDT) is loaded with the user's program.

EXECUTE

This is like LOAD, but instead of exiting, the LOADER starts the program it loaded at its specified starting address.

TRY

TRY is like EXECUTE, but it loads a debugger (RAID at displays, otherwise DDT) with the user program(s). Execution starts at the program's starting address.

DEBUG

DEBUG is like TRY except that rather than starting the user's program it starts the debugger (RAID or DDT). The user may give commands to the debugger to start his program, to plant breakpoints, etc.

## Switches and Project-Programmer Names

Switches are allowed in COMPILE-class commands by including a slash (/) and the switch name, e.g., /LIST. Switches allow the user to select options. In general, a switch placed before a file name is sticky, i.e., it applies to all the terms that follow. Switches placed after a file name are not sticky; they apply only to the preceding term.

Many switches have inverse switches that undo the effect of the switch. For example, /NOLIST is the inverse of /LIST. The inverse setting is the default for all switches.

A project-programmer name (PPN) may be included with each file name in the command. The PPN specifies the disk area from which the file is to be read. The PPN, enclosed in square brackets ([ and ]), usually follows a file name; in this case the PPN applies only to the preceding term. If a PPN appears before a file name, then that PPN is sticky for the rest of the command until another sticky PPN is seen.

A name followed by a colon (:) is a device name (which is sticky).

Switch Function Summary

| | |
|---|---|
| /COMPILE | Force re-compilation without checking the dates of REL files |
| /CREF | Request a cross-reference listing |
| /DUMP | Force the LOADER to make a DMP file of the core image |
| /FAIL | Use FAIL for this term |
| /FORTRAN | Use F4 (FORTRAN) for this term |
| /FORWARD | Prevent RPG from looking for this file to decide if the REL file is up to date |
| /LIBRARY | Force the LOADER to search this term as a library |
| /LIST | Request a listing file |
| /MACRO | Use MACRO to translate this term |
| /MAP | Request a LOADER map of global symbols |
| /NODUMP | Force re-loading without checking the dates of DMP files |
| /NOLOAD | Prevent a loading request for this term |
| /NOSAISEG | Load SAIL programs with the SAIL library instead of the SAIL segment |
| /NONSTANDARD | Specify a nonstandard processor |
| /NSAIL | Use NSAIL (New SAIL) to translate this term |
| /OSAIL | Use OSAIL (Old SAIL - 1971 version) to translate this term |
| /REL | Force loading from the REL file without considering the source files |
| /SAIL | Use SAIL to translate this term |

Switch Names and Abbreviations

| Switch name | Abbreviation | Inverse name | Abbreviation |
|---|---|---|---|
| /COMPILE | /COM | /NOCOMPILE | /NOC |
| /CREF | /C | /NOLIST | /N |
| /DUMP | /D | | |
| /FAIL | /F | | |
| /FORTRAN | /FORT | | |
| /FORWARD | /FORW | /NOTFORWARD | /NOT |
| /LIBRARY | /LIB | /NOSEARCH | /NOSE |
| /LIST | /L | /NOLIST | /N |
| /MACRO | /M | | |
| /MAP | | | |
| /NODUMP | /NOD | | |
| /NOLOAD | /NOLO | /LOAD | /LO |
| /NONSTANDARD | /NON | | |
| /NOSAISEG | /NOSA | | |
| /NSAIL | /NSA | | |
| /OSAIL | /OSA | | |
| /REL | /R | | |
| /SAIL | /S | | |

## Generation of REL Files

The simplest way to translate several programs is to create them with file extensions which define the processor to be used; then give a single COMPILE command with all these names separated by commas. (It is usually a bad idea to have two programs in different languages with the same file name, e.g., FOO.SAI and FOO.FAI.)

RPG does not require file extensions to be typed explicitly. For the command COMPILE FOO, RPG will look for any file named FOO and check its extension. If a file FOO is found that does have one of the standard extensions, RPG will select the corresponding translator.

If a file does not have a standard extension, RPG can be told which translator to use by giving that information in the command string. For example, to translate the file SINE with the SAIL compiler use the command: COMPILE SINE/SAIL. If there is a list of several file names, the processor name may be used after each file name, or before the entire list. For example,

```
COMPILE FOO/SAIL,BAZ/SAIL
```

is the same as

```
COMPILE /SAIL FOO,BAZ
```

The command:

```
COMPILE /SAIL SINE,IOPACK/FAIL,TJ
```

would use SAIL on SINE and TJ, and would use FAIL on IOPACK. The default processor is initially FAIL. Caution: if a file does have a standard (non-null) extension, RPG cannot be convinced to use some other processor.

Every time a switch like /FORTRAN appears in front of a file name, it changes the normal mode. For example,

```
COMPILE /SAIL SINE,COSINE, /FORTRAN TANGNT,COSEC
```

would cause SINE and COSINE to be translated by SAIL, and TANGNT and COSEC to be translated by FORTRAN.

The compilations that are requested will not take place if there already exists a REL file whose creation date is more recent than all of the files used in producing that REL file. If any source file is edited, it will have its creation time and date updated so that it will force re-compilation. A user may force re-compilation, if he wishes, by including /COMPILE in the command line. That is,

```
COMPILE /COMPILE FOO
```

will compile the file FOO regardless of its creation date. Date checking can be forced again by including /NOCOMPILE in the command line. Including /REL will cause RPG to forget about looking for source files at all and just proceed to generating a core image from the existing REL files.

## Concatenation of Source Files

It is often desirable to break up a program into two (or more) separate files which have only one END statement at the end of the last file. One reason for breaking up a program is because it is very long, and it would be inconvenient to keep and edit such a large file. Another reason for separating a program is to allow a portion of the program text to be shared. Several files can be processed together as one big translation by separating their names with plus signs (+) rather than with commas. For example,

```
COMPILE DATA+PROGRM
```

treats the files DATA and PROGRM as if they were one large file and produces one relocatable binary file, PROGRM.REL. A maximum of 40 files may be combined in this way. RPG will not allow concatenation of source files that specify inconsistent processors.

Normally, the REL file will have the name of the last file in the command string; it can be given some other name by putting the desired name and an equals sign (=) at the front of the argument list, e.g.,

```
COMPILE FOO=SYMS+PROGRM+IOPACK
```

would do one translation as if SYMS, PROGRM, and IOPACK were one large file and produce relocatable binary output on the file FOO.REL.

Occasionally, a header file must be assembled with several other files, e.g.,

```
COMPILE ACUMS+SINE,ACUMS+COSINE,ACUMS+TANGNT
```

The brokets (< and >) feature allows this to be done more simply:

```
COMPILE ACUMS+<SINE,COSINE,TANGNT>
```

## Producing Symbolic Listings

Assembly listings have the original code as it appears in the file plus the octal values that the code represents with the relative locations that the octal values go in. Assembly listings can be generated by means of a /LIST or /CREF (cross-reference listing) switch in the command. The /LIST switch will produce a file with the same name as the REL file and extension LST. Such a file may be spooled or typed. Note that COMPILE FOO/LIST will not compile anything if a current FOO.REL exists. This is another instance where the /COMPILE switch is useful. The /LIST may be used as a sticky switch or it may be applied to particular terms. The inverse switch is /NOLIST. (Processors other than the assemblers may also produce listings.)

If the /CREF switch appears, instead of /LIST, it generates an expanded listing that includes a list of all the symbols (identifiers) used in the program and the line numbers on which each symbol

appears. This is called a cross-reference listing. These listing files cannot be printed directly. Instead, they must be processed by the cross-reference lister program, CREF. This is done by the CREF command (see page 54). Note: not all the processors can produce a cross-reference listing. Consult the appropriate manual to find out if a given processor is capable of generating one.

## Command Files

COMPILE-class commands are sometimes too long to fit on one line, or are too complex to be typed correctly every time. To solve this problem, arguments can be read from a file. The text contained in such a file is read by RPG as if it had been typed as the arguments to a command.

The command file may contain any features available in a COMPILE-class command. Also, if the command is too long to fit on one line of the file, a semi-colon (;) placed at the end of a line signifies that the argument list will continue on the next line of the file.

To use a term as a command file write an at sign (@) before the command file name, e.g., LOAD @SYS. The command file SYS might look like the following file which is used to assemble the entire operating system:

```
%S%B SYSTEM=OUTER(XLR)+JOBDAT+ALLDAT+IMPODB;
+LOWCOR+PARSER+SYSINI+COMCSS+CLKSER+UUOCON+SCHEDU+CORE;
+DSKSER+DSKINT+DTCSER+MTCSER+XGPSER+FBPACK;
+IMPCLK+IMPREG+IMPUUO+IMPINT;
+LPTSER+PTPSER+PTRSER;
+DPYSER+TTYSER+LINED+SPWSER+TVSER+ADSER+MAIL;
+PATCH+SYSMAK+DDT+ONCE
```

There are two things in this command file which have not yet been explained, LOADER switches (see page 54) and translator switches (see page 52).

## Generating Core Images and DMP Files

The commands EXECUTE, DEBUG, LOAD, PREPARE, and TRY each generate an executable core image. Core images are made by translating source files (as in the COMPILE command) and then running the LOADER, which takes the REL files and produces a core image from them.

With DEBUG, PREPARE, and TRY, the LOADER is instructed to load a copy of RAID (or DDT if the user is at a Teletype) and the program symbol table with the core image.

If /MAP is included in the command string, the absolute values of all the global symbols are listed in a legible way on the file MAP.MAP.

A disk dump of the core image can be made by including /DUMP in the command string. The dump file will have the same name as the first REL file and extension DMP. The DMP file can be

given some other name by putting the desired name and a left arrow (←) in the command string. For example, the command

```
TRY DUMP←SINE,COSINE/DUMP
```

translates the programs SINE and COSINE, producing SINE.REL and COSINE.REL. The LOADER is run. Because the TRY command requests a debugger, RAID.REL (or DDT.REL) is loaded from the system area. SINE.REL and COSINE.REL are loaded next. Then the LOADER makes a dump copy of the core image in the file DUMP.DMP and starts the program.

If there already exists a DMP file of the right name and if it has a creation date more recent than those of the source files, no loading will be done. Instead, the dump copy itself will be called in. Reloading may be forced by the /NODUMP switch in the command line. Both an equals sign and a left arrow may be included in a command string. For example:

| COMMAND | REL | DMP |
|---|---|---|
| TRY SINE←COS/DUMP | COS.REL | COS.DMP |
| TRY DUMP←SINE←COS/DUMP | COS.REL | DUMP.DMP |
| TRY DUMP=SINE←COS/DUMP | DUMP.REL | DUMP.DMP |
| TRY FOO←DUMP=SINE←COS/DUMP | DUMP.REL | FOO.DMP |

## Library Searches

Several REL files may be combined into a library. A library is a special REL file from which routines may be independently selected. Library files are made by combining REL files together using the FUDGE2 program (which is described in DEC documentation). For example, sine, cosine, and tangent routines might be combined into a trig library, TRIG.REL. Then, if a particular program used just the sine routine, instead of loading the entire trig package it would be able to select only the sine routine. The LOADER will search library files and load only the programs that are needed. The /LIBRARY switch after the name of a library file tells RPG to request this LOADER feature. The /LIBRARY switch may also be used as a sticky switch; its inverse is /NOSEARCH. Suppose there is a program called NLT which uses the sine routine that was included in TRIG.REL. Then the command LOAD NLT,TRIG/LIBRARY will cause NLT to be loaded and TRIG to be searched. The LOADER notices that NLT requires SINE.REL and searches TRIG.REL for it. Suppose another program, XLT, requires routines from two library files, TRIG.REL and IOPACK.REL, and must be loaded with another program LT. Then any one of the following commands could do this:

```
LOAD XLT,LT,/LIBRARY TRIG,IOPACK
LOAD /LIBRARY XLT/NOSEARCH,LT/NOSEARCH,TRIG,IOPACK
LOAD XLT,LT,TRIG/LIBRARY,IOPACK/LIBRARY
```

Programs which use library files must be placed before the library file names in the command. This is because the LOADER cannot know what routines to extract from each library until it has loaded the programs that require library routines.

## Nonstandard Processor Features

Processors other than SAIL, FAIL, FORTRAN, PUB, or MACRO may be specified by using the non-standard processor feature. The switch /NONSTANDARD DEV:NAME specifies a nonstandard processor. In this switch, DEV (assumed to be DSK, if absent) is the device name on which to find NAME.DMP, the core image file of the nonstandard processor. Once the /NONSTANDARD switch is used, RPG will recognize the extension NAM (i.e., the first three letters of the processor name) and the switch /NAME (the processor name) in the command line. A project-programmer name may be used in specifying the non-standard processor.

For example, for a user to translate the program SINE using a processor named PROC on his disk area, he could use the command: COMPILE SINE/NONSTANDARD PROC.

When a standard processor is used, it is known that it will take a text source file and produce a binary REL file, and possibly a text LST file. With nonstandard processors there is no such guarantee. A nonstandard processor may process a text file into a new text file which becomes input to a standard processor. It is possible to specify all these functions in a single COMPILE-class command.

If a nonstandard processor generates a text file instead of a REL file, loading must be inhibited. The /NOLOAD switch accomplishes this. The /NOLOAD switch must be used for each term that does not produce a REL file (except in the COMPILE command, which does no loading anyway). The inverse switch is /LOAD.

Suppose the preprocessor, NSP, translates the file BAZ into a file FOO which is a FAIL program. FOO must be translated when NSP is done. Also, MAIN and SINE must be translated and loaded with FOO. The command:

```
LOAD FOO=BAZ/NONSTANDARD NSP/NOLOAD,MAIN,SINE,FOO/FORWARD
```

will do all of this. The /FORWARD switch is necessary because the file FOO does not exist at the time the command is given. Including the /FORWARD switch prevents RPG from looking for FOO; RPG will assume that FOO magically appears before FAIL is called. The inverse switch is /NOTFORWARD.

The processor DO (see Section 5.7) is another example of a nonstandard processor. The DO program writes QQSVCM.RPG to force the exit-and-go command from the editors to run DO again.

## Translator Switches

Switch names enclosed in parentheses "(" and ")" are passed directly to the translator. The documentation for each of the translators explains what switches are available. The switches passed in this way appear on the source file specification. Sometimes it is necessary to put switches on the binary or listing term. For this purpose, there are two other switch constructions.

The construction (#1,#2), where #1 and #2 are any switch strings, passes #1 to the binary term and #2 to the source term. The construction (#1,#2,#3) passes #1 to the binary term, #2 to the listing term, and #3 to the source term. For example,

```
COMPILE/LIST FOO(AX,BY,CZ)
```

will pass the switch string "AX" to the binary term, "BY" to the listing term, and "CZ" to the source term.

The PUB command will take PUB switches enclosed in parentheses and pass them on to PUB.

## LOADER Switches

Switches to the LOADER may be specified in a COMPILE-class command. A percent sign (%) followed by a letter (or a number and a letter) in the command string of one of the LOAD, TRY, EXECUTE, PREPARE, or DEBUG commands directly specifies that switch to the LOADER. The following are the legal LOADER switches:

| | |
|---|---|
| %A | Causes a listing of all global symbols to be printed |
| %B | BLT symbols down when done loading |
| %C | Chain beginning with Common |
| %D | Load DDT |
| %E | Start up program when done loading |
| %F | Enter library search mode; search all default libraries |
| %G | Finish loading, make final links, and exit |
| %H | Load and start RAID |
| %I | Ignore starting address of this program |
| %J | Use starting address of this program |
| %nK | Adjust to n K of core when done loading |
| %L | Search this file as a library |
| %M | Print storage map |
| %N | Leave library search mode |
| %nO | Set program origin to n, absolute |
| %P | Inhibit automatic library search for undefined globals |
| %Q | Allow automatic library search |
| %R | Chain beginning with resident module |
| %S | Load symbol table also |
| %T | Load and start DDT |
| %U | Print current list of undefined globals |
| %V | Load RAID |
| %W | Load without symbols |
| %X | Do not list all globals |
| %Y | Load SAILOW for 2 segment SAIL programs |
| %Z | Restart LOADER |
| %< | Load high segment |

For a full description of what these switches mean, consult the DEC LOADER documentation. The LOADER has been modified for Stanford, so the DEC documentation does not accurately reflect the state of the software. Consult a wizard if necessary.

## CREF Command

This command causes a cross-reference listing to be made on the line printer. All the files made from COMPILE-class commands which used the /CREF switch are listed. (LOGOUT makes RPG forget which listing files to process.) The CREF command activates RPG which in turn activates the CREF program in a special way. The CREF program when activated in this way reads a file that RPG wrote which contains the names of the files to process.

## Error Messages

### Obvious errors

COMMAND ERROR
TOO MANY SWITCHES
UNRECOGNIZABLE SWITCH
DEVICE NOT AVAILABLE
TOO MANY NON-STANDARD PROCESSORS

### Subtle errors

UFD HASHING ERROR INCITED BY DCS
: Probably you have too many files on one disk area.

TOO MANY NAMES
: Your command included too many file names or non-standard processor names.

PROCESSOR CONFLICT
: A + construction specified files with different extensions, e.g., COMPILE FOO.SAI+BAZ.FAI.

NESTING TOO DEEP
: Too many levels of command files.

NO SUCH FILE
: Some LOOKUP failed. A file named in the command does not exist.

### Horrible errors (A wizard should be consulted)

DISK NOT AVAILABLE
INPUT ERROR READING UFD
INPUT ERROR
NOT ENOUGH CORE

OUTPUT ERROR
: Some file could not be written correctly. Possibly a file is write protected. Otherwise you may need help.

UFD LOOKUP FAILURE
: You are aliased to a non-existent disk area. Otherwise, you need a wizard.

# APPENDIX 2

# COPY

This appendix is taken from COPY.RPH[S,DOC], SAILON 61.1, the COPY manual. The manual first describes the basic COPY command with all possible options, and then lists the other monitor commands which run COPY and describes their special functions.

## Syntax

In this section the following rules are observed. Anything in curly brackets is optional. Anything enclosed in brokets (e.g., <.....>) has a syntactic definition of its own and is described elsewhere. Upper and lower case letters are legal anywhere. A break character is something which is not a letter, a number, or a "$". This will become important when you try to figure out some of the error messages. In this manual the term *sticky* means that the sticky item is applicable until changed or until the end of the current command.

The basic syntax for all COPY commands is:

COPY {{<destination term>}{,<list term>}←}<source term>

"=" may be substituted for "←".

```
<destination term>::=       <switch term>
<source term>::=            <switch term>{,<source term>}
<switch term>::=            {<switch list>space}<term>{<switch list>}
<list term>::=              <term>
<term>::=                   {dev:}{filnam}.ext}}{[p,pn]}
                            | {<switch list>}{dev:}{[p,pn]}{<switch list>}
                                @<spec term>
                            | \<filehack selector>
                            | ∂{ <pn> | * }
<spec term>::=              {<spec switch list>space}{dev:}{filnam}.ext}}
                                {[p,pn]}{<spec switch list>}
<switch list>::=            {/<switch>}{(<page list>)}{<switch list>}
<spec switch list>::=       {/<switch>}{<spec switch list>}
<page list>::=              <page term>{,<page list>}
<page term>::=              N{:M}
```

## General Operation

COPY reads data from the logical file(s) indicated by the *source term*, performs any operations requested by the *switch term(s)*, and writes the data on the logical file indicated by the *destination term*. Any listing information goes to the *list term*.

Dev and p,pn are sticky in the source; that is, if they are absent, the last one specified is used. The default device is DSK and the default p,pn is blank (those of the current job).

A single * may be substituted for any of the following:

```
filnam
ext
p
pn
```

A * in the source term means "all." A * in the destination term means "same as in source file being transferred."

If filnam.ext is absent in the destination term, *.* is assumed. If the destination filnam.ext are specified (no *'s), all source files will be concatenated into one big file with that name.

For example:

```
COPY DING←FOO,BAZ
```

would cause a file named DING to be created on your disk area comprised of the files FOO and BAZ from your area. If filnam.* or *.ext is used in the destination term, the source files will be copied onto the destination one by one with the new filename or extension. <destination term>← may be left out and DSK:*.*← will be assumed. For source terms with directory devices you must specify the filnam.ext.

If you try to write on top of a file which already exists, you will be given the option of deleting it or not. If any of the source files you have specified do not exist, you will be informed and given the proper option.

If a string of source descriptors is specified, the string is executed from left to right.

You cannot write a file in two different modes (e.g. FIL1←TTY:,DSK:FIL2). However FIL1←TTY:,DSK:FIL2/ASCII is quite acceptable (/ASCII causes the data mode to be 0). Data modes are explained in the *UUO Manual.*

The <spec term> construct causes the scanner to continue scanning in the file specified by the term. The term must indicate a unique file. When end of file occurs the scanner again reads from the terminal. No spec terms may appear in the file. Switches preceding the "@" are treated as sticky switches. Switches following the "@" are treated as sticky switches only while reading from the indirect file. In a sense the indirect file is a "program block." Upon leaving the "block" all defaults are reset to those in force just before the "@". A device name and p,pn may precede the "@", in which case the defaults are changed to those, but no operation is done. The device and p,pn used in the spec term do not affect the current defaults.

The \<filehack selector> construct for a term is a special shorthand feature to allow commonly referenced files which would normally require a large amount of typing to be entered quickly. The following is a list of legal filehack selectors and what they are short for (<pn> stands for the current programmer name right adjusted):

| Filehack Selector | File selected | What for? |
|---|---|---|
| MAIL | <pn>.MSG[2,2] | User <pn>'s mail file |
| MSG | <pn>.MSG[2,2] | " |
| MAINT | MAINT.TXT[2,2] | 5 day computer maintenance forecast |
| NOTICE | NOTICE.TXT[2,2] | Public mail file |
| RPG | *.RPG[*,<pn>] | All of current user's RPG files |
| NAP | <pn>.NAP[2,2] | AP news notification file |
| OPTION | OPTION.TXT[*,<pn>] | All of current user's option files |
| DAY | DAY.TXT[2,2] | Daily birthday or holiday message |

The ∂[ <pn> | * ] construct is another form of shorthand for referencing MAIL files (see Appendix 4). If no argument follows the ∂ then the current user's mail file is indicated. If <pn> is used, it means that user's mail file. And lastly, if * is used the file NOTICE.TXT[2,2] is selected (this is the public mail file).

There is a page counter in COPY which monitors ASCII output for form feeds. When a form feed is encountered, the counter is incremented. If a page list is used in a term, N is taken as the starting page of this output and M (or N if M is absent) as the ending page. Output is only active when the counter is within the range of the page list (e.g., LIST COPY(21) would list page 21 of the file COPY). When the end of the first term is exhausted, control is transferred to the next term, and so on until the right parenthesis is reached, at which point output ceases. If more than one file is indicated, the page list is re-scanned for each one. Page lists are never sticky.

Switches will be described in a section of their own. See /LIST for an explanation of "list term."

Logical device names may be used, but all switches apply to the physical device. The following are legal physical devices for the copy command:

```
DSK
UDP
DTAn
MTAn
TTYn
TTY
LPT
XGP
PTR
PTP
```

When the special character ↓ (↑A on Teletypes) appears, the scanner will read the line character-by-character without interpretation until the next ↓ (or ↑A). This is useful for allowing non-alphameric characters to be included in filenames, e.g., editor or spooler temporary files.

For numeric input there is always a default radix (either decimal or octal). Sometimes, however, it is desirable to force one or the other. A number preceded by a single quote is always interpreted in octal. A number preceded by a double quote is always interpreted in decimal.

## Device Peculiarities

Non-Directory Devices:

If the source device doesn't have a directory and the filename and/or extension is specified by a *, then a generated filename and/or extension is used. The generated filename starts out as AAAAAA and the generated extension starts as COP. Either of these may be changed by specifying a filnam.ext in the source term. If a star is used, the old name is retained; otherwise it is replaced by the new one. At the beginning of each term the filename is either changed to that which was specified or incremented by one letter (i.e. AAAAAZ.COP is followed by AAAABA.COP). The following example may clear things up:

```
COPY DSK:*.*←MTA0:,,FOO.*,,*.BAZ
```

This would read 5 files from MTA0 and give them the following names:

```
AAAAAA.COP
AAAAAB.COP
FOO.COP
FOOA.COP
FOOB.BAZ
```

Disk:

If you attempt to write on an area which is protected by a password, you will be asked the password. If you answer <cr> to the request for a password COPY will assume that you don't know it and do the appropriate thing. You may delete files belonging to other users (please exercise extreme care). However, if you attempt to delete a file on a user file directory whose programmer name is different from the one you used when logging in, you will be asked if you are sure. All normal file protection applies to deletions.

User Disk Pack:

COPY has all the routines to make access on a user disk pack look like access on the regular disk. However, one must be sure to format his user pack before trying to use it (see Section 5.8).

DECtapes:

DECtapes are assumed to be in PDP-6 format (which is the current format our system uses). See the *UUO Manual* for an explanation of DECtape formats. If the directory becomes full (more than 22 files), you will be told which was the first file not transferred. If the DECtape becomes full, the file currently being written may be incomplete. An attempt is made to detect this condition before the transfer is initiated, but this is not always possible. In order to write on a DECtape the OFF-WRITELOCK-WRITE switch must be in the WRITE position. If not, you will be told to set it.

Magnetic tapes:

Mag tapes may be made to look like directory devices by use of the /SAVE switch which is

described later. All mag tape switches are separate for input and output (see rules about switches in switch section). This means that mag tape switches in the destination term don't apply to the source terms.

Paper tape:

Paper tape I/O is done in binary non-checksummed mode unless otherwise specified (see /ASCII). You should turn off the paper tape reader when you are done. A paper tape with a jagged end may cause spurious data to be read. It is best to tear it evenly at a fold.

Teletypes:

Since some terminals cannot reproduce the full Stanford character set, COPY provides the standard SOS conversion for output of some special characters on model 33 and model 35 Teletypes which are not in full character set mode. See Appendix 13 for the SOS representation.

On display terminals, COPY displays the number of the page being read (if >2) if reading in mode 0, i.e., the number of form feeds seen +1.

## Switches

The term *sticky* means "applies until the end of the line or until changed temporarily or permanently by some means." Most switches appearing in the destination term are carried over as sticky switches in the source. Those switches which are separate for destination and source terms are indicated by a star preceding the switch name. Switches appearing in the source term are sticky only if they precede the term with which they appear. Otherwise, they apply only to the current term and must follow it. Only the first six characters of a switch are significant. You are only required to use enough characters to uniquely specify the switch. The following switches are now available:

| SWITCH | ABBREV | MEANING |
|---|---|---|
| ASCII | A | This forces transfers to be in ASCII. Beware! This switch will cause SOS line numbers to become part of the text. |
| ASK | ASK | This causes dev:filnam.ext[p,pn] to be printed for each file under consideration followed by a question mark. If you answer "Y" the operation will proceed, if you answer "G" the operation will proceed but you will not be asked the for the rest of the term, otherwise the file will be skipped. |
| BINARY | B | This causes the data mode to be 13 and forces word by word transfer of data. |
| BLOCKED | BL | This causes records to be ended on output whenever they are ended on input. |
| CONVERT | C | This causes copy to treat the character following a line feed and the first character of a file as a FORTRAN control character. Be careful, this happens on output. |
| *DENSITY=n | DE | This sets the density in bpi to n where n equals 800, 556, or 200. This switch is ignored for devices other than mag tapes. The system default density is 556. |
| *OSPOOL | OS | This switch, when used with destination or list terms, causes the output file to be spooled with the /DELETE switch. Output device must be disk. This switch is illegal in source terms. |

| | | |
|---|---|---|
| DUMP | DU | This causes each 36 bit word from the input device to be converted to the ASCII representation of the octal number it represents and to then be sent to the output device. The following format is used. A "word number" is sent to the output device which indicates which word in the file is the first word in this row. Then 8 data words are transferred. Then a <cr><lf> is inserted. If desired, a "page list" may be used for /DUMP; however, the values are those of word addresses in the file instead of pages. |
| *EVEN | EV | This sets mag tape transfers to even parity. |
| EXTRA=n | EX | Whenever a line feed is seen n extra line feeds are put out; n must be an unsigned decimal integer. If n=0 line feeds are converted to a 177 followed by a 21; this inhibits form ejection at page boundaries on the line printer. |
| EXTRA≡n | EX | (Note "≡" not "=".) This switch, used with the XGP only, sets the inter-line spacing to n raster units. |
| FAST | F | All the names specified by the source term are listed (no size or other info). This switch is therefore much faster than /SEARCH. |
| FONT#n=file | FON | This switch is used when copying to the XGP to select a font. Font number n (default is 0) is taken from the specified file. The file specification can include any of filename, extension, and PPN; the defaults are FIX25.FNT[XGP,SYS]. |
| FOONLY | FOO | This switch is used to produce a file directory listing meant to be read by various programs rather than by a human being. The switch name derives from the fact that the switch was created for the benefit of layout and drawing programs used in the design of the late, lamented SUPER FOONLY computer. |
| FULL | FU | This forces all the directory information to be printed under /SEARCH even if no list term was specified. This switch does not automatically imply /SEARCH. |
| GTOTAL | G | At the beginning of the execution of a term the term is printed. At the end of the term, a line is typed of the form<br><br>TOTAL= x.x n BLK pp%<br><br>where x.x is the size of the file in K and tenths (if the file is smaller than 1K, the size in words is typed instead), n is the number of disk blocks allocated to the file, and pp is the percent of the allocated space which is actually used. This switch implies /SEARCH. |
| HEADER | H | This causes a header to be put out at what would be the top of each page on the line printer. The header looks like this:<br><br>DATE TIME FILNAM.EXT P,PN PAGE N-M<br><br>The current date and time are used, N is the logical page number (number of form feeds seen plus one), and M is the physical page of that logical page. |
| IGNI | IGNI | This causes input errors to be ignored (no message). |
| IGNO | IGNO | This causes output errors to be ignored. |
| KILL | K | The input file is deleted after the transfer is finished, even if the output filename is the same as the input filename. If the 200 bit in the file's protection code is on, COPY asks before deleting the file. |
| LIST | L | This causes the names of the files transferred during that term to be listed. If the input device is the disk the [p,pn] are also listed. Non-directory devices will obviously list nothing. The output will go to the "list term". If the list term is absent, the output will go to your terminal. |
| *MLENGTH=nnnn | M | This allows the user to specify nonstandard length records for mag tape I/O, where nnnn is a 1 to 4 digit number interpreted in octal which specifies the number of data words per record. |
| NONUMBERS | N | This deletes line numbers from files that have them. This forces an ASCII mode transfer; in fact, /NONUMBERS can be used on any ASCII transfer. Do not use this switch on anything but text files or you will be sorry. |
| ODD | OD | This sets mag tape transfers to odd parity. This mode is standard and is the default option. |
| OPTIMIZE | OP | This forces COPY to treat the file as one produced by SOS, and to pack it by allowing lines to span a record break as long as line numbers are |

not the last word of a record. SOS will read an OPTIMIZEd file, but will not write a record ending with a partial line. A file which cannot be read by SOS because of line numbers or page marks at the ends of records will be "fixed" by this switch.

| | | |
|---|---|---|
| PROTECTION=nnn | P | This causes the output file to have the protection nnn, where n is a digit from 0 to 7. If no protection switch is specified, the protection of the input file is used (for non-disk input 000 is used). All three n's must be present. This switch applies only to disk output. When concatenating files, the protection applicable to the first file transferred is used. |
| QUIET | Q | This switch prevents you from getting the message "FILE ALREADY EXISTS...." If the output file already exists, it is deleted without a sound. Also, if this switch appears with /SEARCH, then only the p,pn and the size on that p,pn are printed. This switch also prevents being asked for confirmation when trying to delete a file with the 200 bit on in its protection code. |
| RENAME | R | A rename is done on the input file with the output file information. /QUIET applies when the output filnam.ext already exists. |
| *SAVE | SA | If the output device is a mag tape, four words of directory information are written. If the input device is a mag tape, four words of directory information are read and compared with the source name specified. If they don't match, that file is skipped and the next examined until a match is found or the end of tape is reached. |
| SEARCH | SE | This switch overrides all other switches. It gives you more information than /LIST and suppresses any transfers. Added to the /LIST information, you get the size of the file in 1K records (or in words if less than 1K), the date and time it was last written, the protection, and the offset (see the UUO Manual). This is preceded by a suitable heading. The whole smear is preceded by the current date and time. If the information does not exist, it is not listed. If you try to SEARCH a non-directory device, you will get a strange answer. The output goes to the same place as with /LIST, with the following variation: If the "list term" is omitted, the listing goes to your terminal and the information and heading for disk input will be truncated to FILNAM.EXT, [P,PN], and SIZE. If /QUIET is used with this switch and no list term was specified, only the p,pn and the total size of each disk area is printed. If the input device was the disk and the file has been dumped by DART, the date and tape ID of the last dump are printed, followed by ">" if it was dumped twice or more. |
| *SPOOL | SP | This is the same as /DSPOOL except that the /DELETE switch is not used in spooling the file. |
| TITLE | T | This causes a "title page" to be put out in large block letters followed by a form feed. The title is of this form: |

FILNAM.EXT

P,PN

TIME

DATE

The date and time are when the file was last written. For non-directory devices, <device name>: is printed in place of filnam.ext and p,pn.

## Monitor Commands

The following is a list of the monitor commands which use COPY and how they work. The syntax is the same as for COPY.

| COMMAND | EFFECT |
|---|---|
| DELETE | This deletes all the files specified by the source term, and types the space reclaimed as with the /GTOTAL switch. If the 200 bit is on in the file protection key, you will be asked if you really want to delete the file (/QUIET overrides this feature). The destination term is interpreted as the list term. The list term is therefore illegal. |
| DIRECTORY | This gives you all or part of the file information for the source term(s) (or *.* if no source term is used). If /L appears anywhere in the command string the default list device will be set to the line printer. The destination term is treated the same as for DELETE. |
| HELP *name* | This types the file *name*[3,2] on the terminal. If no argument is given or confusion arises, DIR *[3,2]/FAST is done. This command is intended to give information about system programs. You need not be logged in to use HELP. |
| LIST | This sets the default output device to the line printer and turns on the /HEADER switch. |
| PRINT | This sets the default output device to the line printer and turns on the /NONUMBERS and /TITLE switches. |
| RENAME | This allows you to change the filename, extension, p,pn, and protection of a file. The format is newname←oldname. Use /PROTECTION= to change protection. |
| TRANSFER | This copies files from one place to another and deletes the source files. |
| TYPE | This sets the default output device to your terminal. |
| XGPLIST | This sets the default output device to the XGP. |

## Examples

```
COPY DTA4:←*.*
```

This would cause all the files on your disk area to be written on DTA4.

```
COPY ←DTA4:*.*
```

This would cause all the files on DTA4 to be written on your disk area.

```
COPY <filnam.ext>/NONUMBERS
```

This would delete sequence numbers (if any) from the file indicated.

```
COPY [2,RPH]←*.*[1,RPH]
```

This would cause all the files on 1,RPH to be copied onto 2,RPH no matter who you are logged in under (subject to file protection failure).

```
TYPE <filnam.ext>
```

This would cause the specified file to be listed on the terminal you are logged in on.

```
LIST <filnam.ext>(2)/NONUMBERS
```

This would cause page 2 of the specified file to be listed on the line printer with headers and without line numbers.

```
DIR LPT:←*.RPG[*,*]
```

This would cause the filnam.ext, [p,pn], size, etc. for all files in the world with the extension RPG to be listed on the line printer.

```
COPY FOO←TTY:,FOO/ASCII
```

This would allow you to add text at the beginning of the file FOO (/ASCII is mandatory).

```
RENAME FOO/PROTECTION=077
```

This would change the protection of the file FOO to 077 without doing a transfer.

```
PRINT FOO
```

This would cause the file called FOO to be listed on the line printer without line numbers and with a title page.

```
DIR[*,RPH]/QUIET
```

This will tell RPH the size of each of his areas without printing the filenames and other information.

```
COPY TEMP←↓$ED$09↓.TMP
```

This copies the file $ED$09.TMP into TEMP so that you can edit it. (These names are created by SOS.)

```
DELETE @DEL
```

This would use the file DEL as a list of files to be deleted. This method is very good for "cleaning" your area.

# APPENDIX 3

## SPOOL

This description of the SPOOL commands is taken from the SPOOL manual, SPOOL.REG[UP,DOC].

Spooling provides a method of producing line printer listings of files without the usual disadvantage of tying up a user's job while the listing is actually being done.

To use the spooler, a user must create commands for it. A command is the name of a file to list, and control switches. Spooler commands are created by the monitor commands SPOOL and XSPOOL.

As an example of spooling, the monitor command

```
SPOOL SPOOL.REG[UP,DOC]
```

will cause the LPT spooler to list the file SPOOL.REG[UP,DOC].

Spooled listings are preceded by a title page (except see /NOTITLE below), used to make it easy to separate different listings. This title includes the file identifier, the name of the requestor and his project-programmer name, the current date and time, and the date and time when the file was last written. Programmer names are read from the file FACT.TXT[SPL,SYS]. Project names (of which there are few) are read from the file XFACT.TXT[SPL,SYS].

In the discussion below, "SPOOL" is used to mean either the command SPOOL or the command XSPOOL, except as noted. SPOOL makes requests for the LPT spooler; XSPOOL makes requests for the XGP spooler.

To have the spooler make a listing the user must tell the spooler what file(s) to print. This easiest way to tell the spooler what to do is by the monitor command SPOOL. This command consists of the word SPOOL followed by the name(s) of the file(s) to be listed. File names are separated by commas. Each file name may be followed by switches, explained below. The program SPOOL will send commands to the spooler.

Files to be spooled are specified in the conventional way, by file name, extension, and optionally project-programmer name.

Asterisk, *, may be used to allow matching of any file name, extension, project name, or programmer name. (RPG, DMP, and REL files will not be matched by * in the extension).

Switches are allowed either preceding or following the file name. Switches that precede the name are sticky and will be applied to subsequent names. A switch consists of a slash, the switch name, and sometimes, a parameter. For example, /REPEAT=5.

The following switches are available. Note that most switches occur in two senses (e.g., *delete* and *nodelete*). This allows an explicit switch in some term to override a sticky switch.

| Switch | Action |
|---|---|
| DELETE<br>NODELETE | DELETE this file after printing. The file protection of this file must be low enough so that the spooler can delete it. The file must belong to the requestor (alias not allowed). |
| NONUMBER<br>NUMBER | Omit SOS line numbers from the listing. "N" is an accepted abbreviation for NONUMBER. |
| FORTRAN<br>NOFORTRAN | The first character of each line is interpreted as a FORTRAN carriage control character. |
| LPT100<br>LPT0 | The spooler will open the LPT in mode 100, instead of the usual mode 0. (This changes certain character codes.) |
| HEADING<br>NOHEADING | Print a page heading at the top of each page. "H" is accepted as HEADING. HEADING is the default for all files, except:<br>LST or DOC extensions,<br>any file from any area with programmer = DOC.<br>The heading includes date, time, filename, and page numbers. |
| TITLE<br>NOTITLE | Precede the listing with a title page. TITLE is the default for all files. |
| NARROW<br>NONARROW | Center the title page to fit an 8.5 inch wide page. |
| RECOPY<br>NOCOPY | Copy the file to [SPL,SYS] and list the copy. This allows the file to be edited while the listing is in progress. |
| REPEAT=n | Make n copies of the listing. (REPEAT is NOT sticky). |
| EXTRA=n | Insert n extra line feeds in the listing after every line feed in the file. (EXTRA is NOT sticky). |
| DUMP<br>NODUMP | Octal listing. Data will be printed as octal full words. (OCTAL is an equivalent switch.) |
| NOFF<br>FF | Each line feed in the file is translated to '177&'21 which causes single spacing with no form-feed between pages. |
| NOWARN<br>WARN | Suppress "spooling done" message when listing is done. |
| ASK | Before making the spool command, SPOOL will type the file name and wait for the user to respond with "Y" to spool the file. (This is useful with *.* terms). |
| HOLD=hhmm | The listing will be held in the queue until the time specified by hh hours, mm minutes. (24 - hour time is assumed.) If hhmm is prior to the current time, hhmm tomorrow is assumed. |
| MODE=n | Open the file in mode n (n is octal). |

## Delayed Listing

Usually requests for spooled listings will be processed as rapidly as possible. Under some circumstances this may not be desired. For example, a long listing impacts the normal operation of the system by making the line printer unavailable during its duration. The spooler allows users to delay such listings until a time of minimum loading (e.g., 0500) when the unavailability of the line printer is less likely to interfere with other users.

If the construction /HOLD=hhmm appears in a command line then hhmm is interpreted as the (24 hour) time of day when the listing should actually be sent to the line printer. In the example below, 1730 will be interpreted as 5:30 PM.

Example:

```
SPOOL SYSTEM.LST[S,SYS]/HOLD=1730
```

If the time specified is smaller than the current time, it is interpreted to mean, "list the file tomorrow at the time specified." That is, if at 10PM you give the command:

```
SPOOL BAZ.FAI/HOLD=400
```

then the listing will occur in six hours.

## Page Selection

You may follow a file name by a list of ranges of pages that you want listed. The list must be parenthesized. For example:

```
SPOOL FOO.SAI (1,7,5:8)
```

will cause page 1, page 7, and pages 5 through 8 inclusive to be listed. Page ranges are never sticky.

In octal dump mode, this page range list is used to specify a word range to list.

## Multiple Copies of a File

The REPEAT switch (/REPEAT=n) is accepted by SPOOL. The effect is to interpret n as a decimal number and to list that many copies of the file. Multiple copies are spooled as usual, except that the two pages of heading that normally precede a file (except if the /NOTITLE switch is used) are suppressed on the second and subsequent copies of the file.

## XSPOOL Special Features

The XSPOOL command is quite similar to the SPOOL command. The XGP has greater flexibility than the line printer, so more parameters can be specified in the XSPOOL command.

The following switches are legal only in the XSPOOL command:

| | |
|---|---|
| FONT#n=f | n is a number from 0 to 15 decimal. f is a file name. The XGP spooler will use the font named f as font n when spooling. If #n is omitted, n=0 is assumed. Up to 16 font select switches are permitted. The defaults for filename, extension, and PPN are FIX25.FNT[XGP,SYS]. |
| LMAR=n | Set left margin to column n. Columns in the XGP are numbered from 0 to 1699 (approximately). The left margin is the column which the carriage return character selects. |
| RMAR=n | Set right margin to column n. If the XGP is going to write a character that exceeds this margin, a new line will be started. |
| TMAR=n | Set the number of blank scan lines between the top edge of the page and the first line of text. |
| BMAR=n | Set the number of blank scan lines between the bottom of the text and the bottom edge of the paper. (The monitor interprets BMAR=0 to mean no paper cutting.) |
| PMAR=n | Set the number of scan lines in the page body. Text is written inside this area. (The monitor interprets PMAR=0 to mean that the end of a page body is signalled only by a form feed in the file, for variable-length pages. This is true only if BMAR≠0.) |
| XGP | The first page of the file is taken as font and margin commands, preceded by slashes as in the command line. Only the special XGP spooler switches (the ones in this table) are allowed. Carriage returns are ignored. If the first page of the file is a TV/E directory, the second page is used for the commands. The command page is not printed. A file extension of .XGP also invokes this feature. |
| XLINE=n | Set the minimum interline spacing to n scan lines. |

More peculiarities:

If any FONT select switch specifies some font number other than 0, no /HEADING switch will be assumed unless it is explicit.

Page headings will occur in font#0. Whatever font was in effect before the heading line will be reinvoked before any further processing. The title page, if present, will be made in font#0.

## QSPOOL Command

The QSPOOL command starts the SPOOL program in a special way. QSPOOL reads the directory of [SPL,SYS] and interprets the contents of the SPX and XSP (spooler command) files. Essentially, it displays the queue of files to be printed.

The queue is based on a priority scheme. The priority of a spool request is determined by
(time in minutes since spool request, +1) / (file size in records)

Requests are processed in the order of decreasing priority.

The QSPOOL command may be given without logging in.

## UNSPOOL Command

UNSPOOL is a monitor level command that invokes the program SPOOL. If you run UNSPOOL, it will display all the spooler requests that you (your PPN) have requested, and it will allow you to delete any of them. If you delete a request which hasn't been seen by the spooler, then nothing special happens since the spooler is not yet aware of the command, so it won't be missed. If it happens that the spooler is already processing your commands, then the listing in progress will be purged.

If you log in as SPL,SYS and run UNSPOOL, then not only are you allowed to purge any command file that is either in progress or pending but you also can change the spooler behavior in the following ways:

1. You can cause the present listing to be purged.
2. You can cause the present listing to be started over.
3. You can cause the spooler to hold the queue and release the LPT immediately before starting the next file.
4. You can release the queue from the state induced by holding it.

# APPENDIX 4

# MAIL

This appendix describes the use of the MAIL and SEND monitor commands, which are used to send messages to users. The SEND command sends the message to the user's terminal, if he is logged in, and optionally may also (or instead, if he isn't logged in) send it to him as MAIL, so it will be seen on login. The next section describes the REMIND command, for automatic delayed messages, and the LATER command, for automatic delayed program execution. Finally, the RCV command for editing mail files is described.

The information in this appendix is taken from the files MAIL.BH[UP,DOC] and RCV.BH[UP,DOC], the manuals for the MAIL and RCV programs.

## MAIL, SEND, and GRIPE Commands

The basic format of the MAIL command is

```
MAIL <switches> <destination list> <message>
```

and that of the SEND command is

```
SEND <switches> <destination list> <message>
```

Another command, for complaining about system bugs, has the format

```
GRIPE <message>
```

This is a version of MAIL which is directed to a special system area.

The only switches allowed with the MAIL command are /D and /U. The /D switch causes a list of the *destinations* specified by the command to be added to the message as its last line, preceded by "CC: ". If a distribution list file is used (see below), both the file specifier and the contents are included. Note: all specified destinations are listed, even if invalid.

The /U switch is used if for some reason you need to send a message to a user who is *unknown* to the system (has no UFD). The switch allows this but gives a warning message and asks you for each unknown addressee whether you really mean it. These switches may also be used with SEND and REMIND, in addition to the ones listed below.

There are three mutually exclusive switches which may optionally be used with the SEND command. All have to do with the option of mailing messages. Normally, SEND will just send messages to terminals of logged-in users (at all terminals where logged in), but if a destination of

the form PRG (see the next paragraph) is not logged in anywhere, the user will be given the option of mailing the message to that prg instead. The switches /Y for *yes* or /N for *no* will assume that response whenever the situation arises. The switch /M for *mail* will always mail to all destinations on the list as well as sending to their terminals, even if they are logged in.

Another switch used only with SEND is /W, for *where*, which lists on the user's terminal the ppn, job number, job name, queue, and terminal number of all (logged in) recipients of the message.

The destination list for MAIL can be just * , which will send the message to the file NOTICE.TXT[2,2] so all users will see it on login. Otherwise, it is one or more project and/or programmer names separated by commas. The elements of the list should be in the form PRG or [PRJ,*] or [PRJ,PRG] . Any PRJ or PRG can be a dot (.) to mean the same as the user's project or programmer name (real ppn, not alias). List elements may also take the form @<file specification> (default device DSK, default extension DIS as in DIStribution list); the specified file will be read and will replace the @ field in the command. (Note: if the file specification does not give an explicit extension, the given file name with no extension will be tried before the default extension given above.) If such a file contains an @ field, the newly specified file will be read and the remainder of the old file, if any, will be ignored. The DIS file may have a TV/E directory and/or SOS line numbers, which will be ignored. The DIS file may have any number of lines; destinations are separated by any combination of returns, linefeeds, spaces, and zero or one comma. Lines in the DIS file may contain comments started by semicolons, which will be ignored up to a return; note that this is not true of the command line typed at the terminal! In the @ file specification, the notation [PRJ] may be used to mean [PRJ,.]. The notation ↓chars↓ may be used to include non-alphamerics in a file name. To make it easier to refer to mail files as messages (see below), the filename scanner will also accept "filenames" of the form ∂ or ∂PRG or ∂PRJ,PRG or ∂PRJ, or ∂PRJ,*. The first case uses your own logged in programmer name, as does the ∂PRJ, form. These imply device DSK, and default extension MSG and ppn [2,2], although extension or ppn can be specified explicitly.

For the SEND command, there are two special forms of destinations: a job number, and a device name followed by a colon. These special forms can only be used alone, i.e., with no other destinations. The form SEND * will send the message to all logged-in users.

The message part of the command may be a one-line message, which will be sent as is; an @<file specification> (default extension TXT); or null, in which case the program will request a multi-line message from the user. Any TV/E directory, SOS line numbers, and form feeds in a message file will be ignored. If you start typing a short message on the command line, and decide you need more than one line after all, you can end the command line with <lf> instead of <cr> and you will be allowed to continue the message.

Messages can only be sent to real terminals, or to pseudo-teletypes which are connected to the IMP (ARPA users). This excludes detached jobs and non-ARPA PTYs. SEND * sends only to those terminals at which someone is logged in.

When a message is sent to a display terminal (III or Data Disc), the MAIL program tries to determine whether the recipient is running some program, such as E, which changes the normal page printer positioning, thereby making it possible that the message will be lost. If it finds anything unusual about the page printer, it warns the sender of this in a message including the ppn, job name, etc. of the recipient as in the /W output. The sender may choose at this point to

REENTER and change the SEND to MAIL. (The REENTER facility is explained below.) SEND * does not type out these warnings.

Mail can only be sent to defined users, i.e., there must be a UFD matching the destination prj and/or prg names (unless the /U switch is used, as explained above, to permit it). The mail is put in the front of the file <ppn>.MSG[2,2]. If the mail is sent to all users via MAIL *, the program also scans the old NOTICE.TXT[2,2] file as it copies it, and deletes any messages more than two weeks old. Also, TV directories and SOS line numbers are deleted from the old messages in this case.

Another special feature of MAIL * is that an expiration date may be specified for the message. The format is

```
MAIL *← <date> <message>
```

where <date> can be in any of the date formats allowed for the REMIND command (see next section), except that wildcard dates are not allowed and only the date (no time) may be specified. The date given is the one on which the message will be deleted, i.e., it will be seen last on the previous date. This expiration feature is implemented by including the expiration date in the message header, and entering (as with the LATER command described in the next section) a request to run the program EXPIRE.DMP[RMD,SYS] on the specified date. This latter program deletes any expired messages. This implementation is described because the CANCEL command (see next section) will list the request for the EXPIRE program, so you should expect that if you use this feature. If no expiration date is specified, the message is still eligible for deletion after two weeks as described above.

The check for a file directory for the specified user(s) is made before the program asks for the message if the multi-line option is used. If there are no valid destinations, the program exits without doing anything else. If any destinations are valid, however, the program continues as if only the valid destinations were specified. (Exception: the list provided by /D includes all specified destinations, valid or invalid.) Note: in the case of SEND, the only check made before the message is typed in is the file directory check. The check for the user(s) being logged in is not made until later.

There is actually a way not-logged-in users can get messages sent by SEND: it is possible to declare a particular terminal to be your "home" terminal for SEND purposes. If you have a file called OPTION.TXT[1,prg] which includes a line of the form SEND:number then if someone sends you a message with SEND and you are not logged in, a message will be typed on the tty specified by the above number indicating that the message is coming, and it will be mailed as if the sender had said SEND/Y. You can read the message by saying RCV prg. You need not be logged in to do this, but if you are it destroys your core image. If the message is sent to [prj,prg] rather than prg, it looks for OPTION.TXT in [prj,prg] rather than [1,prg].

When a message is MAILed by the MAIL command (not SEND/M, etc.) to a user who is logged in, a one-line notice is sent to his terminal indicating that mail has arrived. This also applies to mail-only reminders (see REMIND/M, next section).

A sample command is

```
MAIL BH,@FOONLY,[S,*] TEXT OF MESSAGE
```

which will send the message TEXT OF MESSAGE to ↓ BH↓.MSG[2,2], ↓ S ↓.MSG[2,2], and whatever users are listed in FOONLY.DIS in the user's disk area.

Spaces may be used in all reasonable places in the command. For the benefit of any weird people who want to start messages with a comma, there is another syntax in which the destination list is enclosed in parentheses. In this case spaces may separate destinations instead of commas if you want. This syntax is also handy for SENDing to a numeric programmer name (otherwise it would be considered a job number) and similarly useless things.

There is an error recovery feature for people who type in long messages, only to discover that because of a typing error they are trying to send it to a non-existent destination or something; to wit: If, at the end of a multi-line message, you type <ctrl><meta><altmode> instead of <ctrl><meta><linefeed>, the message will not be transmitted. Instead, the original command will be loaded into your line editor, and you can correct and re-enter it, but if it does not include a message, the old message will be used instead of asking for a new one. Also, the same effect can be gotten by giving a monitor REENTER command after the program exits or you <call> out of it. From a non-display terminal, of course, the line editor feature is not available, but you can still REENTER and retype the command line. Note: this reenter facility only works after the message has been completely entered; do not <call> in the middle of typing the message and reenter, finish the message and use <ctrl><meta><altmode> instead.

## REMIND, LATER, and CANCEL Commands

The REMIND command is used to tell the MAIL program to send and mail your message at a specified later date and time, possibly repeatedly. The basic syntax is

```
REMIND <switches> <destination list> <datime> <count> <message>
```

The allowable switches are /S for *send only* or /M for *mail only*. If neither is used, the message is both sent and mailed. /D and /U may also be used, as in MAIL or SEND.

The destination list and message fields function like those of the MAIL command, as described in the previous section. Reminder recipients must be users known to the system, i.e., they must have a UFD (unless you use the /U switch as explained in the previous section). The destination field may be omitted, in which case your own programmer name will be used as if the destination field were the character "."; NOTE, however, that some forms of datime (those which start with a letter) could be confused with destinations, and if you want to use such a form there must be an explicit destination.

The datime field may specify a date, a time, or both. The formats allowed for dates are illustrated here:

```
4/28/75         28-APR-75       APR 28, 75
4/28/1975       28-APR-1975     APR 28, 1975
4/28            28-APR          APR 28
4/28/*          28-APR-*        APR 28, *
*/28            28-*            +3
WED             WED*            W               W*
```

The forms without a year imply this year unless the date is today or earlier, in which case next year is used. Letters may be upper or lower case; all after the first three are ignored. The days of the week have, for convenience, the single-letter names familiar to *Farmers' Almanac* readers and former MIT students: M,T,W,R,F,S,D. Asterisk as a month or year is wildcard; note that forms like */28 expire at the end of the current year. WED* and W* mean every Wednesday. The +3 form is the number of days later than today, i.e., three days from now. Spaces may generally be used or omitted as desired except within a word or number, except for the APR 28 forms. Certain mixed notations such as 4/28, 1973 may work, or they may not. If only a time field is used without a date, today is assumed unless the time is already past, in which case tomorrow is used. The weekday forms always imply a day between one and seven days from now as their first or only instance, never today. Here are the formats for times:

| | | | |
|---|---|---|---|
| 1423 | 223pm | 223p | |
| 14:23 | 2:23pm | 2:23p | |
| 0223 | 223am | 223a | 223 |
| 02:23 | 2:23am | 2:23a | 2:23 |
| 2 | 2am | 2pm | 2a |
| +2:23 | +2: | +:23 | 2p |

A date without a time means midnight. The relative forms may only be used without a date. Letters may be upper or lower case. There may be spaces on either side of a colon, but NOT between the time and the am or pm! To use both a date and a time, they must be enclosed in parentheses and separated by spaces: (WED 3am).

The optional count field is used to specify the number of instances of a repeated request. The format is #<number>. If none is used, #50 is assumed (the number is decimal). The form #fff may be used for requests which are to live forever, but please don't leave these around after you go away....

When a reminder which is sent more than once is sent for the last time because its count expires, a warning to that effect is appended to the text of the message. This does not apply to a message which is not rescheduled because of the specified datime, such as a date of */10 which expires at the end of the year.

Here are sample commands:

```
REMIND . (WED* 2:45P) #30 Go to 3:00 class!!
Remind me, sir, to monitor your computer usage.
REMIND * APR 15,* LAST DAY TO FILE INCOME TAX RETURNS
```

The messages will be both sent and mailed at the target time, unless /S or /M is used to specify send-only or mail-only. The message header will say "(reminder)" to distinguish it from an ordinary message. All the features of MAIL work here, like @ fields for destinations or messages, REENTER, etc. There are the usual syntactic ambiguities which are not expected to affect anyone in practice: if the first nonblank after the command name is a digit, it is taken as part of the datime field in REMIND, as opposed to a programmer name in MAIL or a job number in SEND.

The LATER command is used to cause delayed execution of an arbitrary program. The format is:

```
LATER <filespec> <core> <datime> <count>
```

in which datime and count are as in the REMIND command. The filespec determines the program to be run; the default device is DSK (the only other device allowed is SYS), and the default extension is DMP. Note: if no ppn is specified, the ppn used is your disk ppn (alias) at the time the command is entered! The delayed job is given your logged-in ppn as its logged-in ppn regardless of the ppn of the dump file. It is run with JLOG off, i.e., it is killed if any error condition arises. The optional core argument, if used, is enclosed in angle brackets. It may include an initial core allocation and/or a starting address offset, in the form <3K,+1>. The subarguments may be in either order, and must be separated by a comma if both are used. The core allocation argument is in decimal units of 1024 words, and the starting address offset is in octal words. If no job slots are available at the time the job is to be run, it is not done; therefore, if you are depending on the result you should schedule the job for non-prime time. (If there is no job slot for the REMIND phantom itself, it will try to schedule your job as soon after the given time as it is itself run, but it will only try once.)

The monitor command CANCEL will run a program which will list and selectively delete reminder requests from or to you, or all requests if you are logged in as [RMD,SYS]. This program can also be used to list requests without asking whether you want to delete them: after the first request is listed, type the letter L in answer to the deletion question and any remaining requests will be listed without interruption. LATER requests are also listed and deleted by CANCEL.

## RCV Command

The RCV program allows MAIL messages to be selectively deleted, listed, or saved as desired. The program is called by simply entering the monitor command RCV. All mail files which can be construed as being for you are offered for your perusal; if you are logged in as [PRJ,PRG], mail files for PRG or [PRJ,PRG] are automatically listed; mail files for [PRJ,*] or [otherproj,PRG] are announced if they exist and you are asked whether or not you wish to edit them. It is also possible to edit other users' mail files with RCV by giving an argument, e.g., RCV BH; this argument can be PRG or PRJ,PRG or PRJ, or any of those inside brackets. In this case, mail for PRG, [PRJ,PRG], or [PRJ,*] respectively will be listed automatically, and you will be asked about [anyproj,PRG], [otherproj,PRG] (or just PRG), or [PRJ,anyprog] respectively. If there is a message file exactly matching the argument (or your prg if no argument), it will be edited before any other files. RCV * will edit NOTICE.TXT[2,2].

RCV also allows editing of the files created by the A.P. news service automatic notification feature; you are asked whether or not you want to edit such a file addressed to the user whose mail you are examining if it exists. The command RCV \ (or RCV \arg) will examine the A.P. notices first.

It is also possible to use RCV on files not in [2,2] (such as saved message files previously created with RCV) with a command like:

```
RCV #filespec
```

with default of DSK:SAVED.MSG in your (alias) disk area. The conventions given below for output file specs also apply here.

*** NOTE: ONCE RCV HAS OPENED A MESSAGE FILE, YOU SHOULD EXIT ONLY ***
*** BY TYPING "E" TO AN OPTION REQUEST, AS EXPLAINED BELOW. TYPING ***
*** <CALL> TO EXIT MAY RESULT IN LOSING SOME OF YOUR MESSAGE TEXT! ***

When a mail file is opened, messages are typed out one at a time and you are asked for a processing option. The available options are

```
S       Saves the message in the mail file
D       Deletes the message
C       Copies it to another file, keeping it in the mail file too
T       Transfers it from the mail file to another file
L       spools it for the Lpt and deletes it from the mail file
X       spools it for the Xgp and deletes it from the mail file
Z       allows line-at-a-time editing from display terminals
E       saves it like S and Exits, saving all remaining messages
F       sets the File name for C and T (see below)
!       saves the rest of this file and goes on to the next
?       lists these options
```

The option letter must be followed by a <cr> *except for* ? which activates itself. <cr> without an option letter means S. (Yes or no questions must also be answered by a Y followed by a <cr>.)

If you use an argument in the RCV command to edit another user's mail, the first time you specify any option which would remove a message from the mail file or alter its contents you are asked to confirm it. Once you have confirmed such an option, however, you are not asked again.

The Z option, for line-by-line editing, is allowed only from display terminals (III or Data Disc). The message lines are loaded into your terminal line editor one at a time, not including the header line or the space at the beginning of each text line. You can edit the line with the usual line editor commands, and type <cr> when done. If you want to cancel the changes to a line while it is still in the line editor, you can type <altmode>, which will load the original line into the line editor again. If you want to stop editing, leaving the remaining lines as they were, type <control><cr> instead of <cr> after the last line's edits (like E's line insert mode). Lines which are blank (like between messages) are not edited. To delete a line, type <control><meta>D when it is loaded in the line editor. To insert new lines before the line in the line editor, type <control><meta>I or <control><meta><cr>; then type the new lines, ending each with <cr> except the last, which should be ended with <control><cr>. You can also get out of line insert mode by typing an altmode, in which case (unlike <control><cr>) the line with the altmode is not inserted. (Also like E line insert.) Typing <control>D at the end of a line will concatenate it with the next line and allow you to edit the new line. Typing <meta><cr> in the middle of a line will break the line in two pieces, storing the first piece as one line and leaving you editing the second line. The only way to add new lines after the end of the message is to add some spaces or something to the end of the last nonblank line, then <control><bs> over them and <meta><cr>. This crock illustrates my advice that if you have major changes to a message you will be better off using the T option to move it to a separate file, editing with TV, and re-mailing it.

The first time you specify C or T, you are asked to specify a file name. You can reply with a standard file spec, "?", or <cr>. <cr> uses the default file name, initially DSK:SAVED.MSG in your area. "?" will print a helpful message and let you try again. Thereafter, whatever file you specified will be used for all C and T messages (the file is not closed until you exit or select another one) until you specify option F, which asks for a new file name and then for the processing option again. The file which was open for C and T messages provides default extension, etc. for the new request,

e.g., if the first time around you selected MAIL.FOO[S,SYS], if you then type F and specify filename MOBY you will get MOBY.FOO[S,SYS] rather than MOBY.MSG in your area. Note: if the specified file already exists, the new messages are appended to it.

The notation ↓chars↓ may be used to get non-alphameric characters in a filename. In addition, a shorthand notation is provided for entering names of mail files: if the character ∂ is the first character of the file spec, then the device becomes DSK, the default extension MSG, and the default ppn [2,2]. (The extension and ppn can be changed later in the filespec.) If the next non-blank character is not alphameric, the filename used is ↓ prg↓, i.e., your programmer name right adjusted. (Note that this is your own login name, not the name whose mail you are editing.) You can specify a different name with the formats ∂prg , ∂prj,prg , ∂prj,* , or ∂prj, (the latter uses your own programmer name along with the specified project name). Any of these may be followed by .ext and/or [ppn] to change the default values as described above.

Spooling may only be done to one device (XGP or LPT); once you give an L or X command, the other one becomes illegal. Messages so processed are transferred to a file called QQSPL.TMP in your area, and when you exit from RCV a SPOOL or XSPOOL command is executed for that file. This is one of the reasons that you should exit from RCV by typing E, not <call>! If you <call> out, files may not get closed.

It is possible, although hopefully unlikely, that a message may be so long as to overflow RCV's message buffer. In that case, you will see as much as fits, followed by an overflow notice and an option request. After you enter the desired option, the remainder of the message will be typed as it is being processed, so you may safely use the Delete option and you will still see the rest of the message. If you want to avoid this continued typeout, type the letter Q (for Quiet) before your option choice, e.g., QD for Quiet Delete. Q is only recognized when a message overflows.

RCV may be run when not logged in, with an argument specifying the mail files to be edited; however, the messages may only be examined, not modified—the message files are simply listed without asking for processing options. In this case it is safe to use <call> to stop reading messages, but please remember that you must kill the job if you <call> out of it lest it stay around forever!

There are two special options available in RCV, for reading the A.P. news digest and for editing messages sent with the GRIPE command. These options are provided when the RCV command is given without an argument, if you have an OPTION.TXT file with a line of the form

```
RCV:DIGEST,GRIPE;        (either order, or either one alone, is ok).
```

The GRIPE option is intended for system wizards, while the DIGEST option is good for people with both DIGEST and NOMAIL in their LOGIN options. You are asked READ DIGEST? or EDIT GRIPES? if the appropriate files exist.

# APPENDIX 5

# DART

The DART (Dump and Restore Technique) program is used to save disk files on magnetic tape. It also includes tape positioning commands. This appendix is excerpted from the file DART.REG[UP,DOC], which includes information on DART tape formats, complete disk dumps, etc., as well as the following facilities provided for individual users.

In the description below, braces, { and }, are used to denote optional items. Vertical bar, | , denotes an exclusive-or choice. Pointed brackets, < and >, are used to enclose syntactic items that are defined below.

DART accepts the following monitor commands:

```
DUMP        {<dest>←}{<source>}
RESTORE     {<dest>←}{<source>}
REWIND      {<dev>{:}}
EOT         {<dev>{:}}
ADVANCE     {FILE|RECORD}  {<dev>{:}}  {<count>}
BACKSPACE   {FILE|RECORD}  {<dev>{:}}  {<count>}
LOCATE      {<source>}
TLIST       {<dest>}←{<source>}

<dest> ::=          {<dev>:}<file>{.<ext>}{[<prj>,<prg>]}
<source> ::=        {{<dev>:}{[<prj>,<prg>]}@}
                        {<dev>:}<file>{.ext}{[<prj>,<prg>]}{,<source>}

<dev> is "any" legal device name.
<file> is any file name or *
<ext> is any file extension or *
<prj> is any project code or *
<prg> is any programmer name or *
<count> is any string of decimal digits.
```

Caution: The only devices that are appropriate here are disk, mag tape, and UDP. See the semantics section, below, for further explanations.

## REWIND

This command will cause the device named to rewind to load point. MTA0 is the default if no device argument is used. The device named should be a magnetic tape unit.

## EOT

This command will cause ADVANCE FILE to be repeated until either two adjacent file marks are seen (logical end of tape) or until physical end of tape is reached.

## ADVANCE and ADVANCE FILE

This command will cause the tape to advance past the next file mark on the tape. If a repeat factor is given, then the command will be repeated that number of times.

Caution: DART often records more than one disk file on a mag tape file! Therefore, advance file will (sometimes) skip more than one disk file.

ADVANCE RECORD

Same as ADVANCE FILE except that instead of file marks, record marks are used. This command leaves you at the front of a record.

BACKSPACE and BACKSPACE FILE

This command will cause the tape to move backwards until a file mark is seen. DART then does one ADVANCE FILE operation to position the tape at the front of a file (immediately after the file mark just read). If a repeat argument is given, then that argument is used to repeat the backspace operation. Only one ADVANCE is used, after all backspace operations.

Caution: BACKSPACE or BACKSPACE 1 will position the tape at the front of the current mag tape file. BACKSPACE 2 will position one previous, etc.

BACKSPACE RECORD

Same as BACKSPACE FILE except that record marks are used to stop the operation instead of file marks. After all backspaces are completed, DART does an ADVANCE RECORD command.

LOCATE

For each file named in the argument list, this command prints the tape numbers where this file was dumped and the corresponding creation dates of the file.

TLIST

This command will list on the destination device the names of the files that are read from the source device.

DUMP

This command will write on the destination device those files that are specified by the source term. If there is no source argument, *.*[current area] is used.

RESTORE

This command will restore to the destination the files that are described by the source term. A null source or destination means *.*[current area]. The command RESTORE [*,REG]←[*,REG] will restore all of REG's files to the areas that they were dumped from.

# APPENDIX 6

## SERVICE LEVEL SYSTEM (RSL)

The RSL monitor command runs the RSL program to reserve service level. The monitor command takes no arguments; the RSL program itself accepts commands which control its operation. The first section of this appendix explains the concept of service level.

The timesharing scheduler gives different priorities to each of three user classes: interactive users, reserved users, and all others (scavengers). The first goal of the scheduler is to provide good service to anyone doing interactive work such as editing. The system decides whether a given job is currently interactive on the basis of keyboard input activity.

When you log in at a time for which you have made a service level reservation, you are assigned that service level. Whenever you are running, the scheduler will then attempt to give you a processing level (% of CPU time) given by

```
PL = SL - C * (B - 1)/10
```

where SL is the service level, C is your current core size (in K), and B is the price of service level. The system will not let more than a certain total service level to be allocated in this way (currently 80%).

If you are neither interactive nor reserved, you get some of what is left, which may be pitifully little. The system is supposedly rigged so that reserved users always get service at least as good as scavengers. Note that if you are editing and execute a long-running string search, the system may decide you are a scavenger and take forever to finish.

Each authorized user has an allowance of two kinds of money, called *whams* and *bams*. Whams may be used only to purchase reservations for peripheral devices, including III terminals, while bams are good only for CPU service. The allocations are as follows.

| | Whams | Bams |
|---|---|---|
| Half time user | 4 | 100 |
| Full time user | 8 | 200 |
| Panic user | 16 | 400 |

You are *half time* or *full time* in accordance with the portion of your time devoted to A.I. activities. In an emergency situation, you may be given a short term appointment as a Panic user.

Your "money" allocations are actually revolving funds in that if you reserve machine time and use it, you may then re-use the reservation money. In effect, then, there is a limitation on how much you can reserve at a given time. There is no conversion between whams and bams, nor can funds be transferred between persons.

The display costs W (whams/hour) vary with time of day as given in the table below. The price P (bams/hour) of buying any given service level SL (in percent) is

P = SL*B

where B is the rate (bams/hour) given in the same table.

| Time | 0000-0900 W & B | 0900-1300 W & B | 1300-1800 W & B | 1800-2400 W & B |
|---|---|---|---|---|
| Monday-Friday | 1 | 2 | 3 | 2 |
| Sat., Sun., holidays | 1 | 1 | 2 | 1 |

You may reserve a display without a service level or vice versa, but reservations may be made only for integral hours beginning on the hour. Other peripheral devices may be reserved only if you have reserved a service level. The minimum service level purchase is 5%.

The time from 8AM to 9AM and from 5PM to 7PM on weekdays is reserved for maintenance. No service level or device reservations may be made in those time periods until 18 hours before the period. Maintenance reservations for the entire machine may be made, however.

## Using the RSL Program

The monitor command RSL with no arguments runs the RSL program. You must be logged in first.

RSL will prompt you with an asterisk ("*") when it's ready to accept commands. Command format is discussed in detail below. Commands are RESERVE, CALENDAR, AVAILABLE, DISPLAY, MAINTENANCE, HELP, and EXIT.

For example, to reserve 15% SL at 1400 on the 31st of December, any of the following command strings will serve:

```
RESERVE 15 SL AT 1400 HOURS ON 31-DEC
r 15 on 31 decemb at 2 p.m.
RE 2 PM ON 31 DECEMBER 15
R 15 14 31 12
```

If you wish to change a service level reservation, just do another reservation for the new desired amount. Releasing a service level reservation may be done by "reserving" 0% SL.

## IMPORTANT!

When you are through making reservations, you should exit from RSL by typing E. If you exit instead by typing CALL, then there is a possibility that your reservations may get lost. The E command causes reservations to be written out onto the disk.

## How to Reserve Devices

The six III displays, the user disk pack, plotter, and MTA0 may also be reserved. RSL merely makes and retains device reservations; unlike service level, enforcement is left up to the users. To reserve III24, 5% service level, and the user disk pack, type RESERVE 5 SL III 24 UDP or R III4UDP5. A device may be unreserved by preceding it with a minus sign: R -III PLTR -UDP MTA will simultaneously reserve MTA0 and the plotter, and unreserve the UDP and any and all IIIs. If you don't care which III you get, you may say III instead of III4 or III23 etc. Note that the plotter is considered by the computer to be the same device as the paper tape punch, so reserving PLTR also makes the PTP unavailable.

## Other Commands Available

To get a list of your current reservations, type DISPLAY or D for short. The command string D XYZ will get a list of programmer XYZ's current reservations.

The CALENDAR command will show you which programmers have what reservations on a given time and date. For example, C FOR 3 will show you the reservations for the current hour and the following 2 hours also.

AVAILABLE is a short form of CALENDAR. A 0000 HOURS FOR 24 ON 1-JAN-88 will yield 24 lines of output telling how much service level is still available at each time slot on the first day of 1988, should the system last so long.

HELP or H will type for you a summary of command string structure in condensed form, not necessarily comprehensible.

The MAINTENANCE or M command reserves the entire (bare) machine for hardware or software purposes, e.g., M 4 pm for 2 will request the machine for maintenance from 4 to 6 pm. This command can also be used to cancel a bare machine reservation. Just type a minus sign immediately in front of the date specification, e.g. M -25 nov 73 for 2. Do not use this command without prior administrative approval.

Proper use of the EXIT command can be very important, and requires an understanding of its function. The data base for all current and future reservations is kept internally in the *-SL-* program at all times. Many of the commands you give to RSL cause a change in this data base. However, these changes are not necessarily made in the permanent data base, which lives on the disk. The EXIT (or E) command causes the internal data base to be written out on the disk, and it is only this copy which can survive system crashes or system maintenance. *It is possible to lose reservations by leaving RSL with CALL instead of* E.

## RSL Command Syntax

Anything in curly brackets is optional.

A slash ("/") between two terms represents one or more delimiters (space, slash, apostrophe, minus sign, comma).

A double arrow ("↔") between two terms indicates that the order of the terms is not always important. If the syntax of a term distinguishes it, then it may appear anywhere in the command line, provided no nondistinguishable term which might be confused with it precedes it. (It's simpler than it sounds.)

```
<RSL command> ≡ <Rcom>|<Acom>|<Ccom>|<Dcom>|<Hcom>|<Mcom>|<Ecom>
<Rcom> ≡ RESERVE {<sl-arg>}↔{<hr-arg>}↔{date-arg>}↔{<for-arg>}↔{<device>}
<Acom> ≡ AVAILABLE {<hr-arg>}↔{date-arg>}↔{<for-arg>}
<Ccom> ≡ CALENDAR  {<hr-arg>}↔{date-arg>}↔{<for-arg>}
<Dcom> ≡ DISPLAY {<programmer initials>}
<Hcom> ≡ HELP
<Mcom> ≡ MAINTENANCE {-}{<hr-arg>}↔{<date-arg>}↔{<for-arg>}
<Ecom> ≡ EXIT
<sl-arg> ≡ <integer> | <integer> SL
<hr-arg> ≡ {AT} <military hour> | {AT} <civilian hour>
<civilian hour> ≡ { 1 | 2 | ... | 12 } | AM | A.M. | PM | P.M.|
<military hour> ≡ {0 | 1 | ... | 23}{00}{HRS|HOURS}
<date-arg> ≡ {ON} <people day> | {ON} <day-arg>/<month-arg>{/<year-arg>}
<people day> ≡ SUNDAY | MONDAY | ... | SATURDAY
<day-arg> ≡ 0 | 1 | 2 | ... | 31
<month-arg> ≡ <people month> | <computer month>
<people month> ≡ JANUARY | FEBRUARY | ... | DECEMBER
<computer month> ≡ 1 | 2 | ... | 12
<year arg> ≡ {19}71 | {19}72 | ... | {19}88
<for-arg> ≡ <integer> | FOR <integer>
<device>  ≡ MTA | PLTR | UDP | <iii>
<iii> ≡ III | III0 | IIII | ... | III5 | III20 | III21 | ... | III25
```

## Semantics

Names of commands, weekdays and months may be abbreviated by truncation, and may be capitalized or not. Device names may not be abbreviated.

The FOR term (e.g., FOR 3) specifies a consecutive number of hours, starting with the hour and date specified.

When the syntax of a command does not allow a left-to-right scanner to distinguish terms, then the ambiguous terms will be assumed to be in the order listed in the syntax.

If a term is missing from an argument, RSL will supply one it deems appropriate. RSL always has a time and date "in mind," called the default date. The default time and date are usually the last time and date you typed. If a command does not specify a time (date), then the default time (date) will be used.

Commands terminated with a line feed (altmode) will advance (backup) the default time and date before executing the command. The amount of advance (backup) is 1 hour, 1 day, 1 week, 1 month depending as plain, control, meta, or control-meta were used with the line feed (altmode).

A date specified by giving the name of a day, is the earliest future or present date falling on the given weekday. (Notice that at 0830 on Wednesday, "WEDNESDAY" specifies either the current date, or the date one week hence, depending on the hour specification.)

# APPENDIX 7

## FIXIMLAC

The FIXIMLAC command is used from IMLAC terminals to reload the program which controls the terminal into its minicomputer. The command runs a program on the PDP-10, and may be used without logging in first. The use of the command is shown below as part of the startup procedure.

IMLAC RELOADING PROCEDURE

IF THE IMLAC IS DEAD:

1. SHIFT LOCK off (up).
2. Start at 40 (Push STOP, then hold the AT 40 key while pushing START).
3. Type C (upper case) several times to send ↑C's.
4. Type `fiximIM` (upper case M goes out as a <CR>).
5. Wait about 10 seconds to give the system time to log you in.
6. Hit <CONTROL><TOP><SHIFT>S.

IF THE IMLAC IS ALIVE:

1. At monitor level, type FIXIML.
2. When the program puts out a *, type

<CONTROL><TOP>B (Sends the IMLAC to its loader)
<CONTROL><TOP><SHIFT>S (Starts the loader)

# APPENDIX 8

# ARPA NETWORK

The ARPA network is a facility organized by the Advanced Research Projects Agency of the Department of Defense to connect computers at various research centers funded by ARPA, allowing people at one site to use the resources of another site. The device which provides the interface between our computer and the network, called an IMP *(Interface Message Processor)*, can be used by user programs like any other I/O device. Two main system programs are provided for connecting to other computers by console commands: the user TELNET program, called T, and the File Transfer Protocol, FTP. The former allows you to use your terminal as if it were a terminal of the remote host computer; the latter provides high-speed transmission of data between hosts. This appendix explains the use of these programs. It contains excerpts from the file NET.JAM[UP,DOC], which also describes the UUOs for user programming of the IMP. The FTP command description here is taken from FTP.DCS[UP,DOC].

The TELNET or TN commands, which are identical in effect, are used to run T. The FTP command runs FTP. Both require that you be logged in. They take a remote host name as argument. (The host names are listed in a later section of this appendix.) Both programs try to establish a connection to the remote site. Once this connection is established, the T program simply allows you to type at the remote computer as if you were using one of its own terminals. (T also handles special command characters which control local echoing of typein, etc.) FTP, however, accepts commands which allow sending and receiving files. The special commands for these programs will be described below, after a glossary of ARPA network jargon.

## Glossary

NCP — This is the software in the monitor that services the IMP.

TELNET — The thing you type into that sends your characters off to a foreign site is a user TELNET program. The thing that receives them at the other end and passes them on to the timesharing monitor there is the server TELNET. These are generally user-level programs.

SOCKET — When you log in on another system, there are four socket numbers involved. There is a local send socket number, a local receive socket number, a foreign send socket number, and a foreign receive socket number. These numbers are internal connection indexes and are used to keep different connections separate. Each connection has a unique set of 4 socket numbers. (Actually, there are two connections involved, one in each direction. Thus, in ARPA net terminology, each connection has two socket numbers, but you have four sockets altogether.)

LOGGER — Each site is supposed to provide a program that does nothing but sit around and listen to socket 1. When someone connects to socket 1, the LOGGER is supposed to send it back a socket number that the foreign host can use to connect to the LOGGER's host on. Only the serving host knows what socket numbers it can service, and it is the job of the LOGGER to pass out these numbers. (The LOGGER also listens to socket 3 for FTP requests.)

RFC — Request for connection. This is how a pair of hosts establish a connection. The originating host sends an RFC to the destination host with a local socket number and a foreign socket number as arguments. The destination host can complete the connection by returning an RFC with the same socket numbers, or can refuse the connection by returning a close code.

LINK — Once a connection is established, an 8-bit link number is assigned such that the 8-bit host number concatenated with the 8-bit link number is unique at both ends of the connection. During the lifetime of the connection, the link number is used to separate the connections, being as how the socket numbers are 32 bits long. The IMP itself is programmed in such a way that only one message may be in transit on a particular link at a time. The IMP signals the sending host that the message has arrived by returning a RFNM *(request for new message)* with that link number as an argument. All of this is invisible to the user.

CONTROL MESSAGES — Data is sent between hosts on a non-zero link number. Link zero is defined as the control link, on which hosts communicate regarding the connections. All host to host protocol messages are exchanged on link zero.

ALLOCATION — The host to host protocol defines a kind of flow control on a higher level than the RFNM control. This is the allocation system. When a connection is first established, the receiving host sends a control message telling the sending host how many bits and messages can conveniently be buffered. The sending host must then not send more than that many bits or messages without waiting for more allocation from the receiving host. Although much of this is unseen to the user, the user has control over how much allocation the system will give the remote host. For data and file transfer purposes, it is convenient to increase the allocation over the system default amount.

## The User TELNET

Our user TELNET program is called T. It is run by the TELNET or TN commands, which take a site name as argument. When it is run, it attempts to connect to the specified host. It will type

either Connection established if successful, or an error message otherwise. The various possible connection errors are described later.

Foreign sites do not use the Stanford character set. Instead, they accept standard ASCII codes. Letters, digits, and most of the punctuation characters available in ASCII are the same in the two codes, however. The main difference is that codes 1 to 37 (octal), used for special printing characters at Stanford, are control characters in ASCII. The precise use of these control characters is defined by the remote host. TENEX sites, for example, use ASCII control-A to mean "delete one character," like our BS. Our T program handles this difference by interpreting the CONTROL key on a Stanford keyboard to mean ASCII control; i.e., CONTROL on a letter makes T convert the letter to upper case and subtract 100 from the character code. T uses META and CONTROL-META characters as commands to itself, as listed below. In some cases, META and CONTROL-META are equivalent. In others, a processing switch is set by META and cleared by CONTROL-META. In the list below, βX represents META-X, αβX is CONTROL-META-X, and ⊛X means either one.

There are certain differences between the two character sets in non-control characters. These are handled by automatic translation in our TELNET and FTP user and server programs. Specifically, the following translations are done:

| char | Stanford | ↔ | external |
|---|---|---|---|
| ~ | 32 | ↔ | 176 |
| } | 176 | ↔ | 175 |
| ALT | 175 | ↔ | 33 |
| ≠ | 33 | → | 33 |

The Stanford not-equal has no external representation and cannot be entered from a remote site. All other codes are transmitted unchanged. Note that our BS (octal 177) is the ASCII delete; there is an ASCII backspace (10), which is our λ. Typing λ or CONTROL-H to T will produce that code. The other possible ambiguities are our characters ↑ ( 136) and ← (137). These codes have two different meanings: the DEC PDP-10 version of ASCII agrees with our use, but the official version uses those codes for ^ and _. Our network programs support the arrow version, and our ∧ and _ characters are transmitted unmodified as 4 and 30.

It is possible to simulate the CONTROL and META functions when using T from a Teletype. To allow this, T recognizes the character control-E (↑E) as an escape character if it is run at a Teletype. One ↑E before a character means CONTROL, two ↑Es means META, and three means CONTROL-META. For example, ↑E↑EQ from a Teletype is like META-Q at a display.

### Control Commands to T

⊛<number>An octal argument is assembled from digits typed this way.

⊛H Sends the previously typed octal argument as a single ASCII character. For instance, ⊛1⊛0⊛1⊛H is the hard way to send an "A".

αβ<BS> Sends 177 (ASCII delete). The line editor will not send <BS> to programs without CONTROL-META.

βL Enter line mode. Characters you type are sent to your line editor as usual, and not sent to the remote site until activated by RETURN, LINE, ALT, or some CONTROL or META character not taken as a line editor command.

αβL Enter single character mode. In this mode, each character is an activation character, and is sent individually to the foreign host when it is typed. This is useful for using DDT at a remote host.

⊛S Send. If T is in line mode, this command causes all the characters typed so far to be sent to the foreign host. Normally, the characters are sent only when CR, LF, or ALT are typed when in line mode. This command is not useful for Teletypes, because typing ↑E↑ES does not cause an activation in line mode.

βY Enter Datapoint simulation mode. This is often used when talking to the three ITS systems at MIT. If you also convince MIT that you are a Datapoint, their display programs will work properly from our display terminals.

αβY Leave Datapoint simulation mode.

βE Begin local echoing. Different sites have different echoing conventions. Some sites expect to send back echoing, some do not. If you find that what you type is not being echoed in a reasonable time, you may use this command to let you see what you are typing.

αβE Terminate local echoing. One does this when each key one types appears twice on the console. Turning off local echoing will often eliminate one copy of the key.

βR Inhibit sending of LF after CR. Normally, when CR is typed, a LF is invented by the system and CRLF is sent to the foreign host. This command inhibits sending the LF after the CR.

αβR Enable sending LF after CR. The inverse of the above command.

βK Inhibit duplexing LF after CR. The system normally types a LF out when you type CR. This command inhibits that. This command is different in effect from βR, because it alters only what appears on your screen, not what goes out over the line.

αβK Enable duplexing LF after CR. The inverse of the above command.

⊛X Set escape character. The escape character is normally set to ↑E, but may be changed by this command. The next character typed is taken to be the new escape character. This is useful for Teletypes only.

βI Open input file. This command asks for a file name, then proceeds to send said file to the foreign host.

αβI Close input file.

βD Open output file. This command asks for a file name, then proceeds to write everything that comes from the foreign host on this file. Note that characters you type, or characters from any input file that is opened, do not go into the output file unless the foreign host is duplexing them.

αβD Close output file.

⊛Q Quit. Terminates connection and closes any input or output files that may be open.

⊛C Send interrupt. The host to host protocol defines an interrupt that may be sent. This command sends said interrupt.

T normally sets the echoing and activation conventions to those used by the particular site in question, but anomalies do occur. For instance, the foreign host's program may instruct the foreign system to inhibit duplexing, but that will not inhibit local duplexing. For example, if local echoing is in effect, an attempt by the remote computer to prevent echoing of a password will not work.

## The File Transfer Protocol

The FTP command is used to set up a connection to a remote File Transfer Protocol Server. It takes a site argument like TELNET, and the connection process is identical. Once the connection is set up, you can enter the following commands to the ⊛FTP← program. Each command is a four-letter name followed by arguments as described below. Note that file specifications for remote sites must follow the format conventions of that site; they are passed on verbatim by the FTP. In particular, the case of letters may be significant at some sites.

Note: FTP command strings sent over the control link to the remote host are subject to character conversion to conform with standard ASCII, as described in the discussion of the TELNET program. However, the files transferred over the data link are *not* modified by FTP.

BYTE n n is a (decimal) number indicating the *byte size* of the network data connection. Data will be sent from one site to another in bytes of this size.

TYPE x x is A, I, L, P, or E, and specifies the *representation type* for the data. A means the data is in (8-bit) *ASCII*. I is *image* type. The other three are more obscure: *local byte*, *print file (ASCII)*, and *EBCDIC print file*. At the moment, SAIL supports ASCII and image types only. The default is image. If you must know about unusual types and modes, read the official FTP document, NIC 10596.

MODE x x is S, B, T, or H, meaning *stream*, *block*, *text*, or *hasp transmission modes* respectively. At the moment, SAIL supports stream and text modes only. The default is stream.

USER x x is a string which the foreign site will recognize as a valid user description, user name, or ppn. Not all sites require a user name.

PASS foo foo is a password. Some sites may restrict access to those who can supply the magic foo.

ACCT x  x is an account number. Some sites may require this for their billing/accounting purposes.

RETR x←y  This command *retrieves* a file from the foreign site. x is a local file specifier, and y is a foreign file specifier. The foreign file y is copied to the local file x. Current settings (or default values) are used for byte size, representation type, and transmission mode. The x← string is eaten locally, and the RETR y part is transmitted to the foreign site. If you haven't typed any MODE, TYPE, or BYTE commands, the defaults will be sent before the first RETR or STOR.

STOR x→y  STOR means *store*, and this command is the inverse of the RETR command. A local file is copied to a foreign site.

APPE x→y  This command will *append* the local file to the end of the foreign file.

MAIL x  x is a user ID (like our PPN). Following this command you may type in a message which will be mailed to the specified user at the foreign site. The message must be terminated with a line containing only a period.

MLFL x→y  In this case x is a local file specifier, and y is a foreign user ID. The text of the specified file will be mailed to the specified user.

HELP  This command asks the foreign FTP server to send back a message indicating what commands it takes, etc.

STAT x  If x is omitted, this command asks the FTP server to send back information regarding the FTP connection, e.g., socket numbers. If x is a directory name at their site (like our PPN), it lists that file directory.

DELE x  x is a foreign file specifier. The specified file is deleted at the foreign site.

RNFR x  x is a foreign file specifier. This command is short for *rename from*; it must be immediately followed by a RNTO.

RNTO x  x is a foreign file specifier. This command completes the file rename operation started by RNFR. (Some sites accept * for wildcard specification in these commands.)

RSTR  RSTR means *restore*—included to help remedy a hack in the current FTP, wherein sometimes an attempted transfer fails, but our end is unaware of it and leaves the connection open. The RSTR command causes our end to forget everything it knows about data connections to the other end. Any command should be legal after that.

QUOT x  x is any character string, which should be an FTP protocol command with arguments. The string is sent as is over the FTP control link (see FTP protocol, NIC 10596 for terms). QUOT was included to allow you to execute commands which are as yet unimplemented here, or which are nonstandard, specific to some serving site. It will be of little use for those commands which require some special action by the FTP program at this end.

QUIT  Terminate connections with the foreign site and quit.

## Table of Host Mnemonics

Note: A decimal site number may be used with TELNET and FTP instead of a site name. Also, <site>#<socket number> will select a particular socket. (Some hosts provide special services at particular sockets.)

| LONG NAME | SHORT | SITE | LONG NAME | SHORT | SITE |
|---|---|---|---|---|---|
| ABERDEEN | ABER | 29 | MIT-ML | ML | 198 |
| AMES-67 | AME67 | 16 | MIT-MULTICS | MLTX | 6 |
| AMES-TIP | AMET | 144 | MITRE-TIP | MTRT | 145 |
| ARPA-TIP | ARPT | 156 | MOFFETT | MOFF | 45 |
| ARPA | ARPA | 28 | NBS-TIP | NBST | 147 |
| BBN-NCC | NCC | 5 | NBS | NBS | 19 |
| BBN-PDP1 | BBN1 | 197 | NORSAR-TIP | NORT | 169 |
| BBN-TENEXB | BBNB | 133 | PARC-MAXC | MAXC | 32 |
| BBN-TENEX | BBN | 69 | RADC-TIP | RADT | 146 |
| BBN-TIP | BBNT | 158 | RADC | RADC | 18 |
| BELVOIR | BELV | 27 | RAND-RCC | RAND | 7 |
| BOSTON-TIP | BOST | 168 | RML-TIP | RMLT | 165 |
| CASE-10 | CASE | 13 | RUTGERS-TIP | RTGT | 174 |
| CCA-TENEX | CCA | 31 | SCRL | SCRL | 67 |
| CCA-TIP | CCAT | 159 | SDAC-TIP | SDAT | 154 |
| CHI | CHI | 131 | SDAC | SDAC | 26 |
| CMU-10A | CMUA | 78 | SDC-ADEPT | SDC | 8 |
| CMU-10B | CMU | 14 | SRI-AI | SRAI | 66 |
| DOCB-TIP | DOCB | 153 | SRI-ARC | NIC | 2 |
| ETAC-TIP | ETAT | 148 | SU-AI | SAIL | 11 |
| FNWC-TIP | FNWT | 161 | SU-HP | SUHP | 75 |
| FNWC | FNWC | 33 | TINKER | OCAF | 21 |
| GWC-TIP | GWCT | 152 | TYMSHARE | TYMS | 43 |
| HARV-10 | HARV | 9 | UCLA-CCBS | UCL10 | 129 |
| HARV-11 | HRV2 | 137 | UCLA-CCN | CCN | 65 |
| HARV-1 | HRV1 | 73 | UCLA-NMC | NMC | 1 |
| HAWAII | ALOT | 164 | UCSB-MOD75 | UCSB | 3 |
| I4-TENEX | AMES | 15 | UCSD-CC | UCSD | 35 |
| ILL-11 | ILL11 | 12 | UKICS-360 | UK | 42 |
| LBL | LBL | 34 | UNIVAC | ILL | 76 |
| LL-67 | LL67 | 10 | USC-44 | USC | 23 |
| LL-TSP | TSP | 138 | USC-ISI | ISI | 86 |
| LL-TX2 | TX2 | 74 | USC-TIP | USCT | 151 |
| LONDON-TIP | UKT | 170 | UTAH-10 | UTAH | 4 |
| MIT-6180 | MIT | 44 | UTAH-TIP | UTAT | 132 |
| MIT-AI | AI | 134 | XEROX-11 | XER11 | 160 |
| MIT-DMS | DMS | 70 | XEROX-VTS | VTS | 96 |

## Connection Error Messages

CONNECTION HAS BEEN CLOSED

This means just what it says. The host has voluntarily broken the connection for reasons known only to himself.

RESET RECEIVED FROM HOST

The host has sent us a reset command which directs our NCP to break all connections to this host and clear our tables of everything it knows (regarding previous connections) about this host. Some sites will send a reset the first time you connect to them as standard procedure. In this case, trying again will succeed. In the more normal case, this means the host has crashed and just been brought up again.

HOST DEAD

This means that the RFC got to the host's IMP, but was not read from the IMP within 90 seconds, and the IMP timed out. This generally means the host is really down. When this happens, the NCP remembers that the host in question was down so that it does not have to wait for the 90 second timeout. In this case a connection is not even attempted. If you want the system to attempt the connection even though it thinks the host is dead, end the host name with altmode instead of carriage return.

DATA QUOTA OVERFLOW

The host is not conforming to protocol and has sent us more data than we allocated him. We are breaking the connection.

END OF FILE

Doesn't occur in normal circumstances.

SOCKET IN USE

Generally happens after an aborted attempt to connect to someone else. This means the foreign host suddenly sprang to life after T timed out and sent us an RFC from a different socket than he told us he has now. This cures itself in a few minutes.

CAN'T CHANGE SOCKETS
SYSTEM ERROR
NO LINKS AVAILABLE
ILLEGAL BYTE SIZE

These all indicate horrible system errors of one form or another and should not happen.

IMP DEAD

The interface has begun to malfunction, the network control center has brought the IMP down, or someone here has given the magic UUO that brings the IMP software down.

HOST NOT RESPONDING

One of the various timeouts inside T or the system went off. T times out generally when a message is sent to the foreign host which demands a response and no response was received within a "reasonable" time.

The following messages appear only in conjunction with one of the above messages and indicates only the point in T where the error occurred.

CAN'T CONNECT TO LOGGER

The error occurred on the initial RFC to get the foreign host's LOGGER.

DIDN'T GET SOCKET NUMBER FROM LOGGER

We exchanged RFC's with the foreign host's LOGGER, but somehow he did not send us a socket number.

CAN'T CONNECT TO RECEIVE SIDE

We got a socket number from the LOGGER and tried to open a connection on that socket when the error occurred.

CAN'T CONNECT TO SEND SIDE

We got the socket number and successfully sent an RFC to the receive socket, but we hit an error when trying to send an RFC to the send socket.

ERROR WHILE WAITING FOR RECEIVE SIDE

This means that we successfully sent RFC's to the send and receive sockets, but we got an error while waiting for the return RFC on the receive side.

ERROR ON OUTPUT
ERROR ON INPUT

These are given after the connection is established and refer to errors that occurred while doing output or input.

# APPENDIX 9

# CARE AND FEEDING OF DEVICES

## The Line Printer

On the front of the line printer (LPT) there are several buttons (STOP, START, TOP OF FORM, MANUAL PRINT, TEST PRINT, OFF, and ON) and several indicators. The normal state of the line printer is with the START button lit and all red indicators off. (TOP OF FORM is always lit.)

If the system or the spooler reports that the line printer is hung, check the following things. If the START button is not lit and there are no red indicators, push the START button. If there are red indicators, NO PAPER or PAPER LOW ALERT lit, then you have to put in more paper (or fix the paper that has gone astray).

To reload the paper, find someone who knows how to do it and watch him, or do it yourself. Reloading the paper is fairly obvious. The front cover (which includes a transparent section) is hinged at the top; lift it. The yoke is the assembly which carries the ribbon, the printing drum (which you can't see) and a rotating disk at the left side. Open the yoke by finding two toggle switches (one with each hand) and pushing them both down. These switches are located to the left and right of the yoke and below it. They are each labeled CLOSE - OPEN. The yoke will stop when it is fully open. Find four paper tractors, two above the printing hammers and two below. Each tractor has a sprocket wheel (which pokes through the holes in the edge of the paper) and a retainer which holds the paper against the sprocket. Open all the tractors. If the paper is perforated for 8.5 inch wide pages, be sure that the perforation is closer to the right side. Push the TOP OF FORM button before loading the new paper. Use the black arrows to align the "concave" fold. The concave fold is the one where the two adjacent sheets are front to front when folded. Close the four paper tractors so the sprockets fit into the holes in the paper. Close the yoke by finding those two switches and pressing up on both of them. Push START.

Always restart the fan-fold at the back of the LPT so the paper will stack properly.

The line printer logic can become hung. This condition is identified by the MANUAL PRINT light being on continuously. Push MANUAL PRINT and then START to clear this condition. If that fails, open the door on the right side and push the black button labeled RESET. If that fails, find a wizard.

If the ALARM STATUS light comes on or if the printer starts making a loud buzzing noise, shut off the printer (push OFF) and find a wizard. (This light may come on briefly when the printer is being turned on.)

The YOKE OPEN light comes on when the yoke is open while the paper is being changed. When the yoke is closed it should go off.

The printer has a ribbon that can be changed too. It is best to watch someone do it before trying it yourself.

## DECtapes

To mount a DECtape, first assign an available drive. Drives are identified to the system by a rotary switch labeled 1,2,...,8 corresponding to DTA1, DTA2, etc. Although there are eight numbers on the switches, there are only four drives, and the system will not recognize DTA5 and up. Generally you should have no reason to change the normal setting of these switches. Place your tape on the left reel of the drive and thread the tape over the top of the tape guide and head assembly. Wind several turns of the tape onto the takeup reel (turn the reel clockwise). Finally flip the OFF-WRITELOCK-WRITE switch to WRITE LOCK, or if you want to write, to the WRITE position.

To unload the tape, wind the tape off the takeup reel by the direction switch (push it to the left and hold it). Turn the drive off when all the tape is on the left reel. Grasp the reel firmly and pry it off the hub.

## Magnetic Tapes

First, decide whether you want to write on the tape or not. If you intend to write, place one of the plastic write-enable rings in your tape. Place your tape on the top hub and twist the handle in the hub clockwise to tighten the reel to the hub. (You can really lose if you don't tighten it all the way!)

Between the two reels there is a three-position switch labeled START-BRAKES. Move the switch to the right and hold it there to release the brakes on the hubs. Unwind the tape until the end of the tape is about a foot from the floor. Carefully thread the tape into the slot between the reels to the left of the brake release switch. Wind the tape onto the bottom reel (wind the reel clockwise). You must be holding the brake release switch to the right to wind the tape. Wind several feet of tape onto the takeup reel until you see the metal foil load point marker go by. Then push the brake release switch to the left (the START position). The tape will hiss and spin. Hold the switch in START until all motion stops.

There are some control buttons on the top panel. One is labeled (and lighted) REMOTE-LOCAL; this button toggles. In LOCAL mode, push the REWIND button. If you're not sure that you have the load point marker on the takeup reel, push the FORWARD button for several seconds to wind the tape forward onto the takeup reel until it passes the load point marker, then REWIND. REWIND positions the tape at load point. Set the drive in REMOTE and it is all set to use.

Unloading the tape is simple. In LOCAL mode, push the REWIND button. When the tape finished rewinding, push the brake release switch and manually wind the tape onto the top reel. Unscrew the hub and remove the reel.

The magnetic tapes are quite finicky and the system software is somewhat flaky. There are several things that you can do with mag tapes that will upset the timesharing system and require the intervention of a wizard who will be angry at you for disturbing his slumber. Never stop your job when it is operating the tape by typing CALL. Instead, the job can be stopped by causing the magnetic tape to appear hung. This is done by switching back and forth between REMOTE and LOCAL until the system stops the job.

There is a reset button for each tape drive located behind the small front panel below the drive mechanism. This button clears all tape motion functions in a reasonable way.

## Xerox Graphics Printer

The normal condition of the XGP is indicated by the green ON light and the orange READY light being on. If all indicators are off, the XGP main power has been shut off inside the cabinet, possibly for some reason.

If the red OFF light is on, push the ON light. This will turn the XGP on, unless some abnormal condition obtains, such as a paper jam near the cutter.

If the orange STANDBY light is on, look inside the panel above these indicators. The following is the normal state of the indicators and switches within:

```
Knobs:  VERTICAL POSITION:      C
        TEST PATTERN:           OFF

Lights that should be on:
        All POWER SUPPLIES lights
                (except 10kv and 2.5kv lights will
                be off any time there is some other
                problem)

        All PROCESS INTERLOCK lights

        All CONTROL LOGIC (except IN SYNC)

Lights that should be off:
        All SUPERVISORY SIGNALS
        FAULT
        IN SYNC
```

Certain conditions of the XGP are indicated by lights being OFF. (If any of the conditions listed below obtains, the 10kv and 2.5kv power supplies will shut off.)

FWT
: Fixed wait timer has not run long enough after being turned on. Wait 2 minutes.

FUSER
: The fuser is not hot enough. If the XGP has just been turned on, wait 5 minutes; otherwise, reset the over-temperature sensor. The over-temperature sensor is reset by a small black button located somewhere in the middle of the fuser wick assembly.

WEB CLEAN
: The drum cleaning web must be replaced.

PAPER OUT
: Load more paper.

DRUM
: The drum assembly has been disengaged.

SWEEP PRESENT
: The interface is not providing a sweep signal. This may mean the connector is unplugged or the interface turned off.

To load a new roll of paper, open the left door of the XGP. Unscrew the roll retainer and remove the old roll. Unwind a foot of paper from the old roll and then cut the paper at the roll. Mount the new roll and tape the front end of the new roll to paper separated from the old roll. Make the leading edge of the joint as smooth as possible. Using the MOTOR MANUAL DRIVE button, advance the splice through the machine until it has passed the cutter.

If any problems develop in this sequence, find a wizard. Beware: certain parts of the XGP are hot; you can be painfully burned if you touch the wrong things.

There is a drum cleaning web that runs out at inconvenient times. There is a small diagram inside the XGP explaining how to change the web.

Toner is icky black stuff that makes the marks on the paper. It gets used up and runs out too. Someone is supposed to check the toner every day and add more if it gets too low. If the XGP runs out of toner, the pages will start getting lighter, there will be occasional black splotches on the paper, and frequent sand-like particles will be found embedded in the paper. Toner is checked through the access panel in the back of the XGP. There is a lever there that adjusts the rate at which toner is used on the paper. This lever must be placed either at the lightest (inner) setting or one setting darker. Never, never set it anywhere else.

If you don't like the copy quality, complain to a wizard. Don't adjust things yourself.

## User Disk Pack

There is one drive on the IBM 3330 disk system which is reserved for private disk packs. This drive is labeled USER PACK. Use the command ASSIGN UDP to assign the drive before you mount your pack.

Assuming there is no pack mounted already, you can mount your pack by moving the CLOSE-OPEN switch to OPEN. The drive will open up. Remove the bottom cover from the pack by squeezing the two handles on the bottom together. Place the pack bottom on the top of the cabinet. Place the pack (still in the top cover) in the drive unit. Turn the handle clockwise. It will be somewhat hard to turn. Turn until it stops. Carefully remove the top cover and place it on the bottom cover. Make sure the two covers are aligned to keep dust out of the enclosed space. Turn the top cover so the label on it is visible. Flip the CLOSE-OPEN switch to CLOSE and the START-STOP switch to START. The drive will close and start spinning the pack. When the green ready light comes on, the pack can be used. There is a R/W-READ switch for the pack. If you do not intend to write on the pack, it is a good idea to put the drive in READ mode.

Unloading the pack reverses the steps above. Turn the START-STOP switch to STOP, and the CLOSE-OPEN switch to OPEN. The drive will open as soon as the pack slows down. Take the top cover whose label corresponds to the label on the pack and place it carefully over the pack. Turn the handle counter-clockwise until it moves freely. Lift the pack out of the drive. Place the bottom cover under the pack and press it on until it latches. The covers help keep dust off the packs. Return the pack to the storage rack. Leave the label facing outwards so you can find it again.

Occasionally, the UDP may not be available to anyone. If this happens, a wizard will explain why. Never touch a disk drive unit unless you have it assigned! Never touch anything but the drive which is marked USER PACK! If you are confused by something you see, ask about it before you touch!

## Core Storage

We have the most outlandish collection of junk that anyone ever called a memory system. Occasionally it works. At the other times, it produces parity errors and sometimes it decides not to work at all.

Memory errors have never, never been fixed by software means. In particular, reloading the system never solves the underlying problem. If the system becomes unusable because of these errors, a wizard should be found.

Memories stop working entirely due to hardware logic bugs. There is a MEMORY STOP light on the PDP-10 console, which usually indicates this problem. Usually the memory can be reset and the system continued. If possible, find a wizard to fix it. Otherwise, read on. A hung memory can be identified by:

```
DEC core:   AW RQ light off
new Ampex:  UA light off
old Ampex:  UA light off
```

Push stop on the PDP-6 and the PDP-10. Reset the memory by:

```
DEC core:   Push RESTART (top panel) and RESET
            (inside the front panel) simultaneously.
new Ampex:  Push RESET (the big green-blue button)
            on Core 0 and Core 1 simultaneously, or
            Core 2 and Core 3 simultaneously.
old Ampex:  Push the black CLEAR button inside the
            cabinet and the white RESET button on
            the AMPEX INTERFACE box.
```

Once the memory is reset, the computers have to be convinced that nothing bad happened. Set 700200 010003 in the data switches and push the console EXECUTE key. Then push CONTINUE.

# APPENDIX 10

## RELOADING THE SYSTEM

If the system crashes or refuses to run jobs normally, it must be reloaded. If possible, find a system programmer, who will try to find out what the trouble is. If that fails, check the MEMORY STOP light on the PDP-10 console. If this light is on, look up the memory stop procedure in Appendix 9. Otherwise, read on.

First, write in the log your name and the reason that the system is being reloaded.

Find the paper tape labeled SYSTEM LOADERS and mount it in the paper tape reader. (It may be mounted already, or it may be an unlabeled blue mylar tape loop.) Push the console keys RESET and READ IN.

If you're lucky, the system will load itself without further hassle. If the PDP-6 is stopped, it must be started by setting 204 in its address switches and pushing the keys INSTRUCTION STOP, IO RESET, and START.

If the system is being loaded without the PDP-6, then the current date and time must be typed in at the CTY during the reloading sequence. Be sure you supply the correct date and time!

Sometimes the system loader tape doesn't work. In this case, find the DECtape with the current system. The DECtape will be labeled with the system name and date. Mount it on some DECtape drive. Find the paper tape labeled 71.5K RIM10B DECDMP and load it into the paper tape reader (on the PDP-10). Stop the PDP-6. Press RESET and READ IN on the PDP-10. The tape will be read, and the CTY will type carriage return and line feed. Type nL, where n is the number of the DECtape drive which has the system tape. The tape will spin for a while. Eventually, the CTY will type crlf again. Type „206G to start the system. (If the Librascope disk is down, type „200G instead.)

# APPENDIX 11

# MONITOR ERROR MESSAGES

This appendix lists the error messages typed by the monitor itself (not by other system programs) with explanations for some of them. Many have to do with errors induced by UUOs in your program, and these are explained in the *UUO Manual*, in the writeup of the UUO involved. The list is alphabetical; messages which start with a variable part are sorted under the first fixed word.

Note: Several of these messages are of the form XXX AT USER <address>. It is possible in some of these cases to get XXX AT EXEC <address> instead, which means that the offending instruction was not in your program, but in the monitor, which was trying to do something on your behalf when it happened. If this happens repeatably, or if the message is shortly followed by a system crash, tell a system programmer.

ADDRESS CHECK FOR DEVICE <dev>

> Your program was trying to use the indicated device, and supplied a buffer address, dump mode command address, etc., outside the bounds of your core image.

ADDRESS OUT OF BOUNDS, UUO AT USER <address>

> Your program executed a UUO which takes as an argument an address in your core image, and the address was out of bounds. The address given is that of the offending UUO.

ALREADY ASSIGNED TO JOB <job number>

> You typed an ASSIGN command, and the device you wanted is in use.

<terminal> ALREADY ATTACHED

> You typed an ATTACH command, and the job you are trying to attach is already attached to a terminal.

?AMBIGUOUS JOB NAME

> Your program gave a MAIL UUO which refers to another job by name, and there are two or more jobs with that name.

ATTEMPT TO ENABLE INTERRUPT ON PDP-6

> Your program tried to enable a user interrupt routine to run on the PDP-6. This feature is not implemented.

ATTEMPT TO GENERATE NON-ENABLED INTERRUPT

Your program tried to send itself or another job an interrupt for which the job was not enabled.

ATTEMPT TO INITIATE SPW JOB WITH ONE ALREADY RUNNING

Your program has a spacewar module active and is trying to start another one on the same processor.

ATTEMPT TO SCHEDULE 1-LEVEL PROCESS, UUO AT USER <address>

Your program's interrupt routine executed a UUO which would put it in a wait state. User interrupt routines aren't allowed to do that. See the *UUO Manual.*

BAD DIRECTORY FOR DEVICE <dev>

The device is a DECtape. This might mean that the tape is in a format not recognized by the monitor. (Our system uses a different directory format from the PDP-10 standard.) It can also happen if you try to write on a write-locked DECtape.

BAD RETRIEVAL

The pointers to your file on the disk are invalid. See a system programmer to try to fix it.

BUFFER TOO LARGE, UUO AT USER <address>

Your program is trying to use an I/O device for which there is a limit on the allowable user buffer size, which you exceeded. The *UUO Manual* explains the rules for each device.

BUSY

You gave a TALK command addressed to a terminal which is in user mode or has characters in its input buffer.

CAN'T ATT DEV

You gave the privileged ATTACH <device> command and don't have the DEV privilege.

<dev> CAN'T BE REASSIGNED

You gave a REASSIGN command in which the device to be reassigned was your own terminal. This is illegal.

CAN'T CONTINUE

You typed CONTINUE after one of the other error messages came out, and you are not allowed to continue the job after that error. You also cannot continue a job after a SAVE or SSAVE.

CAN'T DET DEV

You gave the privileged DETACH <device> command and don't have the DEV privilege.

CAN'T ENTER-RENAME MFD

If you get this message, see a system programmer, who will eat you.

CAN'T LOCK WITH SEGMENT

Your program, which has an upper segment, gave a LOCK UUO. This is illegal.

CAN'T TERMINATE SOMEONE ELSE'S SOCKET!

Your program is using the ARPA network and tried to terminate a connection which does not belong to it.

CONFLICT BETWEEN INTERRUPT ENABLINGS

Your program tried to enable both old-style and new-style interrupts in a conflicting manner.

CONS TRAP AT USER <address>

You can only get this one from a program running on the PDP-6, which means it must be a spacewar module. The CONS instruction is a nonstandard addition to our PDP-6 installed for the benefit of LISP people. It sometimes gives this message. The address is that of the CONS which failed.

CORE DEADLOCK.

The monitor's free storage area has expanded enough so that user core is no longer big enough for your core image. This might fix itself if you wait a while, but if you must run immense programs your best bet is to come back at 3am.

COULDN'T GET YOU A SEGMENT. WILL TRY TO LET YOU WIN WITH SETPR2.

You tried running a program with an upper segment. The program was loaded into core, but there is no room in the monitor's job tables to make an entry for the upper segment. The monitor will try to simulate an upper segment for you by leaving the upper segment code in your lower segment, but simulating upper-segment relocation by the SETPR2 UUO mechanism. This ought to work ok for most two-segment programs, but just in case it doesn't the monitor types this warning. If you are writing a two-segment program, consult the *UUO Manual* or a system programmer for advice.

COULDN'T UNPURIFY UPPER. CONTINUE TO TRY ANYWAY.

You typed a DDT command. JOBDDT in your core image points to an address in your upper segment as the DDT starting address, and your upper segment is write protected. DDT and RAID both contain instructions which modify themselves. The monitor tries to solve this by unprotecting your upper segment. If you are sharing the segment with other users, it has to load a new copy for you so the other users still have a protected one. This might fail, for example, because there are no job slots available. All highly unlikely.

<dev> DEASSIGNED, BUT STILL INITED!

You gave a DEASSIGN command for a device which is in use by your program. This is all OK; the monitor is merely reminding you that you still control the device. The FINISH command will release it, if that is what you want.

DEVICE <dev> NOT AVAILABLE

The device specified in a RUN, R, GET, SAVE, or SSAVE command was unavailable to your job.

DEVICE <dev> OK?

The device is somehow disabled. If it's a mag tape, it may be in LOCAL mode. The line printer could be out of paper. You can fix the problem and type CONTINUE.

DISK IS FULL!

Your program is trying to write a file on the disk, and there is no room. If you delete some files (using another job!), you can type CONTINUE and the file will be written.

DISK TRANSMISSION ERROR

This is a disk failure. See a system programmer.

ENTER FAILED

The filename you typed to ENTER NEEDED (see below) didn't work. Most likely this is a protection failure.

ENTER FAILED!

The file specified in a SAVE or SSAVE command cannot be written. This probably means that the file already exists and is write protected.

ENTER NEEDED. PLEASE TYPE FILE NAME.

Your program tried to write on the disk or a DECtape without doing an ENTER UUO to specify the filename. You are given the opportunity to specify the file to be used.

ERROR IN DSKSER

Horrible error in the monitor disk service routine. See a system programmer.

ERROR IN JOB <job number>

This message will appear along with one of the others. The only reason you might need it is if you are running more than one job at once (through a PTY, for example).

ERROR IN MONITOR

This message is usually followed closely by a system crash. Call a systems programmer.

FILE ALREADY EXISTS. DELETE?

The filename you typed in response to ENTER NEEDED (see above) already exists. If you type Y (and RETURN), it will be replaced with the new output; otherwise, you are asked for another name.

FINISH WHAT?

You typed a FINISH command with an argument and weren't using that device (or there is no such device).

GOTCHA!

This is not an error. You get it when you are in a device wait queue and your turn for the device comes.

HALT AT USER <address>

Your program executed a JRST 4, instruction. You are allowed to CONTINUE the job after this message. The address typed is the address containing the HALT instruction, not its effective address.

HUNG DEVICE <dev>

The device did not respond properly to your attempt to use it in some predetermined time. This probably means that the device is unusable for some hardware reason.

HUNG DEVICE: PDP-6

Your program is trying to run a spacewar module on the PDP-6, which is not responding. Get a system programmer to fix it.

I-LEVEL TIME-OUT

Your program's user interrupt routine has run longer than the maximum time allowed. (No other user can run while an interrupt routine is in progress, which is why they are limited in time. See the *UUO Manual* about interrupts.)

I-LEVEL UUO GIVEN WHEN NOT AT I-LEVEL

Your program tried to execute a UUO which is only allowed in a user interrupt routine, when it was not servicing an interrupt.

ILL INST. AT USER <address>

Your program executed an undefined instruction or one which is not allowed in user mode.

ILL MEM REF AT USER <address>

Your program tried to read or write an address greater than the size of its core image. The offending instruction is at the address typed.

ILLEGAL ATTACH LOOP

Your program is typing into a pseudo-teletype and gave an ATTACH command trying to attach its controlling job. In particular, people using the system via the ARPA network get this if they try to attach to their TELNET server job. (You also get it if you try to attach the job controlling the PTY controlling the job controlling you, etc.) Typed on the PTY.

ILLEGAL DATA MODE FOR DEVICE <dev>

Your program has tried to use an I/O device in an undefined way, e.g., binary I/O to a terminal. The *UUO Manual* discusses data modes for each device.

ILLEGAL DD CHANNEL.

Your program is trying to use a Data Disc channel to which it is not allowed access.

ILLEGAL FORMAT DUMP MODE COMMAND LIST

Your program is trying to do dump mode I/O incorrectly. See the *UUO Manual.*

ILLEGAL INSTR. ADDRESS.

This message refers to instructions in a display program your job is running.

ILLEGAL PAGE SIZE SPEC.

Your program is trying to adjust the page printer geometry incorrectly on a display terminal.

ILLEGAL PIECE OF PAPER

Your program gave a PPIOT UUO which specified a piece of paper number greater than 17 (octal). See the *UUO Manual.*

ILLEGAL UUO AT USER <address>

Your program executed a UUO which the monitor does not recognize. (Probably you are trying to execute data.)

IMP IS DOWN

You are trying to use the ARPA network, and the IMP has been declared dead by the monitor. This may be a real IMP failure or only a temporary problem which can be fixed. Ask a system programmer.

INPUT DEVICE <dev> CANNOT DO OUTPUT

Your program has done something silly like trying to write on the paper tape reader.

INPUT DEVICE <dev> HAS NO INPUT BUFFER HEADER, UUO AT USER <address>

Your program executed an input UUO in buffered mode, and had not provided an input buffer header when it opened the device.

IN USE.

You gave the FLUSH command and the specified terminal is in use.

INTERRUPT ADDRESS OUT OF BOUNDS

Your program has enabled user interrupts and specified an interrupt address which is not in your core image.

IO TO UNASIGNED CHANNEL AT USER <address>

Your program executed an I/O UUO specifying an I/O channel which was not opened. See the *UUO Manual* about channels and I/O.

<dev> IS BUSY, WILL YOU WAIT?

Your program tried to open a device which another job is using. If you say Y (then RETURN), your job is placed in a queue of jobs waiting for the device, and will be continued automatically when it's your turn. If you say N, you are then asked DIRECT IO TO DISK? and can again say Y or N. If you say Y, you may be asked for a filename (if your program does input or output on that channel without specifying a name). If you say N again, your program gets a failure return on the INIT or OPEN UUO.

JMS NOT LEGAL; USE JSR.

Your program set up a III display program with the forbidden JMS instruction in it. See the *UUO Manual.*

JOB CAPACITY EXCEEDED

The maximum allowable number of users are already logged in. Try again later.

<n>K OF CORE NEEDED

The maximum amount of core storage available to user jobs is not enough for the core image you are trying to run. The number in the message is the decimal number of 1024-word blocks needed for the program. (Note: if you give a RUN or GET with a core size argument which is too big, the message will tell you the minimum amount of core necessary to fit the dump file, not the amount you asked for. The amount in the message might well fit even though what you asked for didn't.)

LOGICAL NAME ALREADY IN USE, DEVICE <dev> ASSIGNED

You typed an ASSIGN command with a logical device name argument which you were already using for another device. The device you requested is assigned to your job, but the logical name still refers to the old device.

LOGIN PLEASE

You typed a command which requires that you log in first.

LOOKUP FAILED

The filename you typed in response to LOOKUP NEEDED (see below) could not be read. (It doesn't exist or is read protected against you.) You get to type in another name.

LOOKUP NEEDED. PLEASE TYPE FILE NAME.

Your program tried to read from the disk or a DECtape without specifying a file by executing a LOOKUP UUO. You are given the chance to specify the file to read.

NEED ENTER BEFORE OUTPUT
NEED LOOKUP BEFORE INPUT

You get these messages if your program opens a directory device, closes a file, and then tries to do input or output on that channel without another LOOKUP or ENTER.

NO CORE ASSIGNED

You typed a command which refers to your core image (e.g., DE or START), and you don't have a core image.

NO DDT

You typed a DDT command, and your core image does not contain DDT or RAID (JOBDDT zero).

NO REENTER ADDRESS

You gave a REENTER command and your program has no REENTER address (JOBREN is zero).

NO STARTUP ADDRESS!

You gave a START command and your program has no start address (JOBSA zero).

NO SUCH DEVICE

You typed something the monitor did not recognize when a device name was required.

NO SUCH JOB

You gave an ATTACH command with a nonexistent job number.

NON-EX JOB NAME OR NUMBER

Your program gave a MAIL UUO addressed to a nonexistent job.

NON EX MEM AT USER <address>

This can't happen, it says here. It means that your program tried to address a word of core storage which does not exist. However, we have the maximum possible amount of core, so there shouldn't be any nonexistent addresses. Possibly a hardware failure could cause this. (If you are running a spacewar module which resets its relocation and protection registers when for some reason we are running with less than the full amount of core, you can get this legitimately.)

NOT A DUMP FILE

The file specified in a RUN, R, or GET command is not in dump file format.

NOT A TTY.

You gave the FLUSH command with an argument which is not the name of a terminal.

NOT ENOUGH CORE!

You typed a CORE command with an argument greater than the maximum core available to a user job.

<filename> NOT FOUND

The file specified in a RUN, R, or GET command was not found. If the problem is not a spelling error, make sure you are not aliased.

OUT OF BOUNDS

Some numeric argument to a monitor command wasn't right. This could be an address not in your core image in a DE or E command, or an invalid job number in a KILL command.

OUTPUT DEVICE <dev> CANNOT DO INPUT

Your program has tried to do something silly like read the line printer.

OUTPUT DEVICE <dev> HAS NO OUTPUT BUFFER HEADER, UUO AT USER <address>

Your program executed an output UUO in buffered mode and did not provide a buffer header when it opened the device.

PARITY ERROR IN YOUR CORE IMAGE. LOC = <address>
PARITY ERROR IN YOUR UPPER SEGMENT. LOC = <address>

A hardware failure has invalidated the contents of the indicated address in your core image. The safest thing to do in this case is to restart with a fresh copy of your program. If you were running for a long time and are willing to risk continuing, you can type CONTINUE. You can also try to fix the bad location with DDT or RAID, if present, or with the DE and E commands.

PC EXCEEDS MEM BOUND AT USER <address>

Your program tried to jump to an address outside the range of your core image. The message, unfortunately, tells you the illegal address rather than the address of the jump instruction.

PDL OV AT USER <address>

Your program had a pushdown list overflow. The address typed is the effective address of a PUSHJ, the return address for POPJ, or the address following a PUSH or POP.

PIECE OF GLASS TOO BIG

Your program is trying to run a display program which is too big for the monitor to handle. Sorry.

PLEASE KJOB OR DETACH

You gave a LOGIN command when you were already logged in.

PLEASE TYPE ↑C FIRST

You started your program with CSTART or CCONTINUE, leaving your terminal in monitor mode, and then typed a command which affects your core image. Such commands cannot be processed while the program is running. Type CALL and try again.

PROJECT-PROGRAMMER NUMBER MISMATCH

You typed an ATTACH command, and the job number and PPN arguments did not agree.

RESET CALLED FROM I-LEVEL, OR SPACEWAR LEVEL

Your job's user interrupt routine or spacewar module executed a RESET UUO.

SAVE/GET IO ERROR

Try again. If it still fails, ask a wizard for help.

SORRY, NO <dev>'S AVAILABLE

You typed an ASSIGN command with a generic device name (e.g., MTA), and all of them are in use.

SPACEWAR LOSSAGE - <error>

The error indicated can be ILL MEM REF, PC EXCEEDS MEM BOUNDS, NON EX MEM, PDL OV, or CONS TRAP. It means that your job started a spacewar module (see the *UUO Manual*), which caused the error. See the particular error message in this list for an explanation.

SWAP CALLED FROM I-LEVEL

Your job's user interrupt routine executed a SWAP UUO.

SWAP READ ERROR

Your job was swapped out of core, and when the monitor tried to read it back in, there was a hardware error on the swapping disk. Your core image is probably gone forever. If you really need it badly, you might be able to persuade a system programmer to try to recover it for you, but even then there's not much chance. Start over.

THIS FORM OF MTAPE IS NO LONGER SUPPORTED!

You are trying to run an old program which uses old-style disk MTAPE UUOs. See the *UUO Manual* for the new disk MTAPEs.

TOO FEW ARGUMENTS

This error message can be typed by several different monitor commands and should be self-explanatory.

TOTAL DPY BUFFER SPACE EXCEEDED.

Your program is trying to run a III display program, and there is no more room in your allocation of display buffer space.

**$$** USER DPY ERROR AT <address>

This message comes along with another which explains the error in your display program. The address is that of the offending instruction.

USER I-LEVEL <error>

The error can be ILL. MEM. REF., NON-EX. MEM., or PDL OV. See the associated message in this list for an explanation. The error happened in your program's interrupt routine. If it says CH3 ERROR instead of one of the above, see a system programmer.

UUO AT USER <address>

This line appears with certain other messages. It tells you the location in your core image of the instruction which caused the error.

<dev> WASN'T ASSIGNED

You gave a DEASSIGN or REASSIGN command and hadn't assigned the device.

WASN'T DET

You gave the privileged ATTACH <device> command and have the privilege, but the device wasn't detached.

YOU ARE LOCKED OUT OF CORE!

The amount of user core available has been reduced by other jobs starting spacewar modules (thereby being locked in core). There is not enough room to fit your core image. This condition may fix itself quickly; the message does not return you to monitor mode, but will keep appearing every so often until you fit again. If you have a really huge program, run it late at night.

# APPENDIX 12

# BIBLIOGRAPHY

This appendix lists various other sources of information about system programs. Some of these are available on the disk, some are printed. Some programs, which were not written here but came from DEC, are documented in DEC manuals. Other programs, documented on the disk, are not included here; only the major processors are listed. The files AIMS[BIB,DOC], AIMS.OLD[BIB,DOC], SAILON[BIB,DOC], and PRUNE.DAT[UP,DOC] list other program documentation which may be of interest. A convention used in naming [S,DOC] and [UP,DOC] files is that the filename is the name of the program being documented, and the extension is the programmer name of the author.

SAILONs (Stanford Artificial Intelligence Laboratory Operating Notes) and AIMs (Artificial Intelligence Memos) are generally available in printed form from the project secretaries. Try this before making your own listing.

EDITORS:

| | |
|---|---|
| TV: | TVED.DCS[UP,DOC] is the reference manual for the TV editor. |
| E: | TV2E.FW[UP,DOC] describes the differences between E and TV. Read TVED.DCS also. |
| SOS: | SOS.LES[S,DOC], SAILON 50.3, is the reference manual for SOS. |
| TECO: | This is a DEC program. See the *DECsystem 10 Users Handbook.* |

LANGUAGE PROCESSORS:

| | |
|---|---|
| FAIL: | FAIL.PMP[S,DOC], SAILON 26.2, is the reference manual. (A new manual is in preparation.) |
| SAIL: | SAIL.KVL[AIM,DOC], AIM 204. This is a long manual; try to get the printed version instead of spooling a copy. |
| MACRO: | This is a DEC program. See the *DECsystem 10 Assembly Language Handbook.* |
| F4: | This is the DEC FORTRAN. See the *DECsystem 10 Mathematical Languages Handbook.* |
| LISP: | Our version of LISP, LISP 1.6, is described in LISP.WD[S,DOC], SAILON 28.7. |
| PUB: | PUBNET.TES[S,DOC], SAILON 70, describes the PUB Document Compiler. |

DEBUGGERS:

| | |
|---|---|
| RAID: | The display terminal debugger is described in RAID.PMP[S,DOC], SAILON 58.1. |
| DDT: | The Teletype debugger is a DEC program; see the *DECsystem 10 Assembly Language Handbook.* |

## APPENDIX 13

## STANFORD CHARACTER SET

The Stanford ASCII character set is displayed in the following table. The three digit octal code for a character is composed of the number at the left of its row plus the digit at the top of its column. For example, the code for "A" is 100+1 or 101.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 000 | NUL | ↓ | α | β | ∧ | ¬ | ∈ | π |
| 010 | λ | TAB | LF | VT | FF | CR | ∞ | ∂ |
| 020 | ⊂ | ⊃ | ∩ | ∪ | ∀ | ∃ | ⊗ | ↔ |
| 030 | _ | → | ~ | ≠ | ≤ | ≥ | ≡ | ∨ |
| 040 | SP | ! | " | # | $ | % | & | ' |
| 050 | ( | ) | * | + | , | - | . | / |
| 060 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 070 | 8 | 9 | : | ; | < | = | > | ? |
| 100 | @ | A | B | C | D | E | F | G |
| 110 | H | I | J | K | L | M | N | O |
| 120 | P | Q | R | S | T | U | V | W |
| 130 | X | Y | Z | [ | \ | ] | ↑ | ← |
| 140 | ‘ | a | b | c | d | e | f | g |
| 150 | h | i | j | k | l | m | n | o |
| 160 | p | q | r | s | t | u | v | w |
| 170 | x | y | z | { | \| | ALT | } | BS |

The tables below display the standard ASCII codes, and the *SOS representation* used at Stanford for entering the full Stanford ASCII character set from Teletypes or similar terminals with restricted character sets. The obscure names for the ASCII codes below 40 are listed just for confusion. Notes: "DEL" (177) is the ASCII delete. "ESC" (33) is their alt mode. Codes 136 and 137 have two different interpretations, as shown below. The SOS representation is so called because it is provided by SOS, the Teletype editor. Certain other programs also know about this representation, but it is not built into the monitor in any way.

Standard ASCII

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 000 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL |
| 010 | BS | TAB | LF | VT | FF | CR | SO | SI |
| 020 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB |
| 030 | CAN | EM | SUB | ESC | FS | GS | RS | US |
| 040 | SP | ! | " | # | $ | % | & | ' |
| 050 | ( | ) | * | + | , | - | . | / |
| 060 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 070 | 8 | 9 | : | ; | < | = | > | ? |
| 100 | @ | A | B | C | D | E | F | G |
| 110 | H | I | J | K | L | M | N | O |
| 120 | P | Q | R | S | T | U | V | W |
| 130 | X | Y | Z | [ | \ | ] | ^↑ | _← |
| 140 | ‘ | a | b | c | d | e | f | g |
| 150 | h | i | j | k | l | m | n | o |
| 160 | p | q | r | s | t | u | v | w |
| 170 | x | y | z | { | \| | } | ~ | DEL |

SOS Representation

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 000 | --- | ?! | ?" | ?# | ?$ | ?% | ?& | ?’ |
| 010 | ?( | TAB | LF | VT | FF | CR | ?) | ?* |
| 020 | ?+ | ?, | ?- | ?. | ?/ | ?0 | ?1 | ?2 |
| 030 | ?9 | ?6 | ?3 | ?= | ?< | ?> | ?7 | ?8 |
| 040 | SP | ! | " | # | $ | % | & | ’ |
| 050 | ( | ) | * | + | , | - | . | / |
| 060 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 070 | 8 | 9 | : | ; | < | = | > | ?? |
| 100 | @ | A | B | C | D | E | F | G |
| 110 | H | I | J | K | L | M | N | O |
| 120 | P | Q | R | S | T | U | V | W |
| 130 | X | Y | Z | [ | \ | ] | ↑ | ← |
| 140 | ?@ | ?A | ?B | ?C | ?D | ?E | ?F | ?G |
| 150 | ?H | ?I | ?J | ?K | ?L | ?M | ?N | ?O |
| 160 | ?P | ?Q | ?R | ?S | ?T | ?U | ?V | ?W |
| 170 | ?X | ?Y | ?Z | ?[ | ?: | ALT | ?] | BS |

# APPENDIX 14

## MONITOR COMMAND SUMMARY

On the following pages, the monitor commands are listed first alphabetically and then by function. In the alphabetical lists, the command name is printed with the minimum necessary unique abbreviation capitalized and the rest of the name in lower case. One list provides a brief description of the purpose of each command, and another displays the arguments for some commands and gives a page reference to the complete discussion in this manual. If the command loads a program into your core image, its name is listed. The following abbreviations are used:

| | |
|---|---|
| phydv | physical device name |
| dv | logical or physical device name |
| prj | a project name |
| prg | a programmer name |
| sa | octal starting address<br>(if omitted use program starting address) |
| cor | decimal number of 1024 word blocks<br>to assign to this program |
| lh | octal left half word |
| rh | octal right half word |
| adr | octal address |
| jn | decimal job number |
| fn | a file name (may include project-programmer name) |
| site | an ARPA network site name |
| c...ɔ | enclose privileged version of command |
| → | first program listed automatically loads the next |
| ⊗ | program to be loaded is specified as argument |
| {...} | enclose optional argument |
| \| | separates alternative arguments |
| - | no arguments for this command |

| NAME | DESCRIPTION |
|---|---|
| A | see ASSIGN |
| ADvance | mag tape positioning |
| ALias | set disk PPN |
| ASsign | assign device to job |
| ATtach | attach job to TTY |
| | c attach device to system ↄ |
| Backspace | mag tape positioning |
| C | see CORE |
| CAncel | delete reminders |
| CContinue | continue job, TTY in monitor mode |
| CDetach | continue job, detach TTY |
| CEtv | create file with E |
| CFork | continue job, detach TTY, make new job |
| COMpile | compile programs |
| CONtinue | continue job, TTY in user mode |
| COPy | copy files |
| CORe | set or type core size |
| CREAte | create file with SOS |
| CREF | make cross-reference listings |
| CStart | start job, TTY in monitor mode |
| CTv | create file with TV |
| D | see DEASSIGN |
| DAytime | type time of day or job times |
| DDt | enter DDT or RAID |
| DE | deposit into core image |
| DEAssign | deassign device from job |
| DEBug | compile and load programs, start debugger |
| DELete | delete files |
| DETach | detach job from TTY |
| | c detach device from system ↄ |
| DIrectory | type file statistics |
| DO | execute commands from a file |
| DUmp | dump disk files to mag tape |
| E | examine core image |
| ED | see EDIT |
| EDDt | c stop system, enter Exec DDT ↄ |
| EDIt | edit file with SOS |
| EOt | mag tape positioning |
| ETv | edit file with E |
| EXecute | compile, load, and run programs |
| F | see FINISH |
| FILes | type status of files in use |
| FIND | find character string in a file |
| FINGer | type name and location of users |
| FINIsh | close and release device |
| FIXimlac | initialize IMLAC terminal |
| FLush | clear TTY buffers |
| FOrk | detach, make new job |
| FTp | ARPA net file transfer protocol |
| G | see GET |
| GEt | load core image from file |
| GRipe | document system bugs |
| HAlt | stop job (↑C) |
| HELLo | type name of current monitor version |
| HELP | explain system program |
| K | see KJOB |
| KAttach | kill this job, attach another |
| KIll | kill another job |

| NAME | DESCRIPTION |
|---|---|
| KJob | kill this job |
| KLog | kill this job, make another |
| L | see LOGIN |
| LAter | run a program later |
| LISP | run LISP |
| LIST | list file on line printer |
| LOAd | compile and load programs |
| LOCate | tell which dump tapes have file |
| LOGIn | make a new job |
| LOGOut | kill this job |
| MAIl | send mail to a user |
| MAKe | create file with TECO |
| PJob | type job using device or this TTY |
| PPpn | print PPN of job |
| PREpare | compile, load programs with debugger |
| PRInt | list file on line printer |
| PTty | type TTY attached to job |
| PUb | compile document with PUB |
| Qspool | type spooler status and queues |
| R | run program from [1,3] disk area |
| RCv | receive mail for user |
| REAssign | assign device to another job |
| REEnter | start program at reenter address |
| REMind | create a reminder message |
| REName | rename files or change protection |
| RESOurces | type available system resources |
| RESTore | restore files from dump tape |
| REWInd | mag tape positioning |
| RSI | reserve service level or devices |
| RUn | run a program from core image file |
| S | see START |
| SAve | save core image in a file |
| SEnd | send message to user's TTY |
| SLevel | type service level assignment |
| SPool | request line printer listing |
| SSave | save two-segment core image |
| STart | start job, TTY in user mode |
| SYstat | type system status |
| TAlk | talk to another TTY |
| TECo | edit file with TECO |
| TELnet | talk to ARPA net computer |
| TIme | type runtime for job or system |
| TLIst | list files on a dump tape |
| TN | see TELNET |
| TRAnsfer | copy files, delete source |
| TRY | compile, load with debugger, run |
| TTy | set TTY parameters |
| TV | edit file with TV |
| TYpe | type a file |
| Unspool | delete spooler request |
| W | see WHO |
| WHEre | type job information for PPN or job |
| WHO | display system status |
| XErox | compile XEROX documents (not XGP!) |
| XGplist | list file on XGP |
| XSpool | request XGP listing |
| Zero | clear DTA, UDP, or DSK directory |

| NAME | PROGRAM | ARGUMENTS | PAGE |
|---|---|---|---|
| A | see ASSIGN | | |
| ADvance | DART | | 78 |
| ALias | | prj],prg] | 29 |
| ASsign | | phydv [dv] | 26 |
| ATtach | | jn[(prj,prg]] | 26 |
| | | c dv ↄ | 42 |
| Backspace | DART | | 79 |
| C | see CORE | | |
| CAncel | FORGET | - | 75 |
| CContinue | | - | 25 |
| COetach | | - | 26 |
| CEtv | E | fn | 43 |
| CFork | | - | 26 |
| COMpile | RPG→⊛ | | 45 |
| CONtinue | | - | 25 |
| COPy | COPY | | 56 |
| CORe | | [cor] | 24 |
| CREAte | RPG→SOS | fn | 43 |
| CREF | RPG→CREF | - | 54 |
| CStart | | [sa] | 25 |
| CTv | RPG→TV | fn | 43 |
| D | see DEASSIGN | | |
| DAytime | | [jn] | 28 |
| DDt | | - | 25 |
| DE (deposit) | | [h rh [adr] | 24 |
| DEAssign | | [dv] | 27 |
| DEBug | RPG→⊛ | | 46 |
| DELete | COPY | | 63 |
| DETach | | - | 26 |
| | | c dv ↄ | 42 |
| DIrectory | COPY | | 63 |
| DO | DO | [fn] | 36 |
| DUmp | DART | | 79 |
| E (examine) | | [adr] | 24 |
| ED | see EDIT | | |
| EDDt c CTY only ↄ | | - | 42 |
| EDIt | RPG→SOS | [fn] | 44 |
| EOt | DART | | 78 |
| ETv | E | [fn] | 44 |
| EXecute | RPG→⊛ | | 46 |
| F | see FINISH | | |
| FILes | | [jn][fn] | 28 |
| FIND | FIND | | 38 |
| FINGer | FINGER | [prg] | 40 |
| FINIsh | | [dv] | 27 |
| FIXimiac | FIXIML | - | 85 |
| FLush | | dv | 27 |
| FOrk | | - | 26 |
| FTp | FTP | site | 90 |
| G | see GET | | |
| GEt | ⊛ | [dv] fn [cor] | 23 |
| GRipe | MAIL | | 70 |
| HAlt (or ↑C) | | - | 25 |
| HELLo | | - | 28 |
| HELP | COPY | [fn] | 40 |
| K | see KJOB | | |
| KAttach | LOGOUT | jn[(prj,prg]] | 33 |
| KIll | | jn | 29 |
| KJob | LOGOUT | | 32 |
| KLog | LOGOUT→LOGIN | prj,prg | 33 |
| L | see LOGIN | | |
| LAter | MAIL | | 74 |
| LISP | LISP | - | 40 |
| LIST | COPY | | 63 |
| LOAd | RPG→⊛ | | 46 |
| LOCate | DART | | 79 |
| LOGIn | LOGIN | prj,prg | 30 |
| LOGOut | LOGOUT | - | 33 |
| MAIl | MAIL | | 70 |
| MAKe | RPG→TECO | fn | 43 |
| PJob | | [dv] | 27 |
| PPpn | | [jn] | 28 |
| PREpare | RPG→⊛ | | 46 |
| PRInt | COPY | | 63 |
| PTty | | [jn] | 28 |
| PUb | RPG→PUB | [fn] | 45 |
| Qspool | SPOOL | - | 69 |
| R | ⊛ | fn [cor] | 23 |
| RCv | RCV | | 75 |
| REAssign | | dv jn | 27 |
| REEnter | | - | 25 |
| REMind | MAIL | | 73 |
| REName | COPY | | 63 |
| RESOurces | | - | 28 |
| RESTore | DART | | 79 |
| REWind | DART | | 78 |
| RSI | RSL | - | 80 |
| RUn | ⊛ | [dv] fn [cor] | 23 |
| S | see START | | |
| SAve | | [dv] fn [cor] | 24 |
| SEnd | MAIL | | 70 |
| SLevel | | - | 29 |
| SPool | SPOOL | | 65 |
| SSave | | [dv] fn [cor] | 24 |
| STart | | [aa] | 25 |
| SYstat | SYSTAT | - | 40 |
| TAlk | | dv | 29 |
| TECo | RPG→TECO | [fn] | 44 |
| TELnet | T | site | 87 |
| TIme | | [jn] | 28 |
| TList | DART | | 79 |
| TN | see TELNET | | |
| TRAnsfer | COPY | | 63 |
| TRY | RPG→⊛ | | 46 |
| TTy | | | 29 |
| TV | RPG→TV | [fn] | 44 |
| TYpe | COPY | | 63 |
| Unspool | SPOOL | - | 69 |
| W | see WHO | | |
| WHEre | WHERE | [prj,] [prg] | 40 |
| WHO | WHO | - | 37 |
| XErox | XEROX | | 41 |
| XGplist | COPY | | 63 |
| XSpool | SPOOL | | 68 |
| Zero | ZERO | dv | 36 |

Commands that run programs:

| LOGIN Class | COPY Class | DART Class | MAIL Class |
|---|---|---|---|
| KATTACH | COPY | ADVANCE | CANCEL |
| KJOB | DELETE | BACKSPACE | GRIPE |
| KLOG | DIRECTORY | DUMP | LATER |
| LOGIN | HELP | EOT | MAIL |
| LOGOUT | LIST | LOCATE | RCV |
| | PRINT | POSITION | REMIND |
| | RENAME | RESTORE | SEND |
| | TRANSFER | REWIND | |
| | TYPE | TLIST | |
| | XGPLIST | | |

| RPG Class | SPOOL Class | Information | Others |
|---|---|---|---|
| CETV | QSPOOL | FIND | DO |
| COMPILE | SPOOL | FINGER | FIXIMLAC |
| CREATE | UNSPOOL | HELP | FTP |
| CREF | XSPOOL | SYSTAT | LISP |
| CTV | | WHERE | RSL |
| DEBUG | | WHO | TELNET |
| EDIT | | | TEST |
| ETV | | | TN |
| EXECUTE | | | XEROX |
| LOAD | | | ZERO |
| MAKE | | | |
| PREPARE | | | |
| PUB | | | |
| TECO | | | |
| TRY | | | |
| TV | | | |

Other system commands:

| SAVEGET Class | Device Control | ATTACH Class | Information |
|---|---|---|---|
| CORE | ASSIGN | ATTACH | DAYTIME |
| GET | DEASSIGN | CDETACH | FILES |
| R | FINISH | CFORK | HELLO |
| RUN | FLUSH | DETACH | PJOB |
| SAVE | REASSIGN | FORK | PPPN |
| SSAVE | | | PTTY |
| | | | RESOURCES |
| START Class | Other | | SLEVEL |
| | | | TIME |
| CCONTINUE | ALIAS | | |
| CONTINUE | DE | | |
| CSTART | E | | |
| DDT | EDDT | | |
| HALT | KILL | | |
| REENTER | TALK | | |
| START | TTY | | |

The end of the *Monitor* off Cape Hatteras.