Georgia
Institute
of
Technology SCHOOL OF INFORMATION AND COMPUTER SCIENCE / (404) 894-3152 / ATLANTA, GEORGIA 30332

TIME SHARING SYSTEM
USERS MANUAL
for the
BURROUGHS B5700

July 1972

ACKNOWLEDGEMENTS

This manual describes the B5700 Time Sharing System which was developed and supplied by Burroughs Corporation, Detroit, Michigan. It has been previously described in a Burroughs publication entitled the "B5700 Time Sharing System Terminal Users' Guide." The Burroughs supplied software implementing many of the Command and Edit language verbs has been modified by the Rich Electronic Computer Center (RECC) staff. In addition, software generated by the RECC staff has been added to implement new verbs. This manual includes the description of many verbs as contained in the Burroughs publication but with considerable expansion, including much original material and many examples. Georgia Tech is indeed grateful to the Burroughs Corporation for developing and supplying such an excellent Time Sharing System.

This manuscript was prepared by the Staff of the Rich Electronic Computer Center and was released to the School of Information and Computer Science for publication.

Table of Contents

Table of Contents (Cont'd.)

Table of Contents (Cont'd.)

Table of Contents (Cont'd.)

1. FEATURES OF THE TIME SHARING SYSTEM

The Time Sharing System utilizes the most recent operating system software to be developed by Burroughs Corporation for the B5500/B5700 hardware. Both the current hardware and software result from many years of evolutionary development under the Burroughs approach to computer system design. This approach is to totally integrate both hardware and software through a policy of cross-training the design specialists. Hardware engineers learn the intricacies of software architecture, and software specialists learn the subtleties of hardware logical design. These people are then merged into a single system design team.

1.1. Development and Installation History

In the late 1950's, such a team produced the hardware and software design specifications for the Burroughs B5000 computer system. In 1963, one of the first B5000's was installed at Georgia Tech using the original software operating system called the Master Control Program (MCP). At that time the B5000 was used only for batch processing, the only access mode for which it was designed. In 1965, the hardware configuration was expanded. The most significant changes were the addition of a second central processor and the replacement of auxiliary drum storage with disk storage. A revised software operating system, called the Disk File Master Control Program (DFMCP), was supplied to utilize the new storage medium. Batch processing was still the only mode of access. At this time, Burroughs changed the name of the system from the B5000 to the B5500.

In 1966 and 1967, the hardware configuration was again expanded. This time the most significant change was the addition of telephone interface equipment. A revised operating system, called the Data Communications Master Control Program (DCMCP), was supplied by Burroughs to support the remote terminal access mode in addition to the usual batch processing. These events made available for the first time at Georgia Tech the new dimension of remote computing. The features of the original Master Control Program software were broad enough to require only a modest amount of alteration to produce the DCMCP. Although the DCMCP permitted remote users to interact and converse with their programs, it treated remote jobs more or less the same way it treated batch jobs. This caused some remote users to sit idle for extended periods of time

before their jobs were initiated; however, once the job was initiated, the response time was usually satisfactory. Psychologically, this left something to be desired. One solution which developed for this problem was time slicing, or true time sharing.

The B5500 Time Sharing System eliminated this inadequacy and has been in use at Georgia Tech since June 21, 1971. No hardware enhancements were required, nor have any been made since 1967. Again, only the software operating system was modified and called the Time Sharing Master Control Program (TSMCP). Since both the DCMCP and the TSMCP evolved from the same origin, they both contain the same important features of the original design, and they differ only in certain resource management strategies. Currently, the TSMCP is used during certain hours of each day for remote operations with some concurrent background batch processing, and the DCMCP is used during other hours for batch processing only.

In 1970, Burroughs announced the availability of certain additional hardware for the B5500, such as extended core memory and a data communications processor. Newly installed B5500 systems with these hardware features are called B5700 systems. The Georgia Tech system does not contain either of these features. Subsequent to the B5700 announcement, some new Burroughs publications and several revisions to older manuals used the term B5700 in their titles. At the present time, all software and programmer reference manuals with either B5500 or B5700 in their titles are pertinent to the Georgia Tech system.

1.2. Design Features of the Fundamental System

The original B5000 system contained many significant features, primarily as a result of the unified hardware and software design approach. All of these original features are still present in the system as it now exists, although many changes and enhancements have been made in the interests of user convenience and operational efficiency. Some of the more significant features will be described rather briefly.

1.2.1. The Master Control Program (MCP)

One of the B5000 design objectives was a system capable of controlling its own resources and scheduling work on a dynamic basis. This was accomplished with the operating system software, originally called the Master Control Program, or MCP. Later, as significant capabilities were added to the MCP, it was called the DFMCP, the DCMCP, and now the TSMCP. The MCP (all of them)

is the controlling traffic director of the entire system. It is always in control of the hardware, or has the means of regaining control. Its main function is to permit the hardware to process all the user jobs presented to it by the human operator. Furthermore, it must perform all of the many, many tasks associated with job processing as efficiently as possible. To be efficient it must be cognizant of the resources available, the resources needed by each user job, and be able to allocate and deallocate resources to the job stream as quickly as possible.

1.2.2. Hardware Status Monitoring

In many operating systems the available resources (hardware configuration) are specified as parameters when the software is assembled for loading (system generation). In such systems, if the hardware configuration should change, either by adding more devices such as core memory, tape units, card readers, etc., or by removing some device, possibly for temporary maintenance, the operating system may require partial or complete regeneration. In Burroughs systems, the MCP continually checks the status of the total system. It maintains comprehensive but compact availability tables that indicate the number of tapes, disks, printers, lines, I/O channels, and other devices which are available and ready. Changes in status are easily detected because of the high degree of hardware-software integration.

1.2.3. "Fail-Soft" Capability

This dynamic assessment of current resources permits a fail-soft capability; i.e., malfunctioning devices can be dynamically removed for maintenance and the system continues to function. Of course, there are fewer resources and throughput decreases, but, nevertheless, the machine continues to function. A memory module, an I/O channel, a peripheral device, or even a central processor in a two-processor system, can be taken off-line and the MCP will reroute the work around the missing components; certain devices may even be removed while the system is operating without causing system failure.

1.2.4. Dynamic Resource Management

Provided with up-to-date resource availability tables, the MCP is able to dynamically assign certain resources to a particular user job. Rather than, at job initiation time, assign all the resources that will ever be needed, specific resources are dynamically assigned and deassigned to each job as required. This dynamic assignment is accomplished not only for peripheral devices such as tapes, printers, etc., but also for the floating I/O channels

and main core memory. This means that only that portion of main core memory (the most expensive resource) which is actually required at that instant is ever assigned to a job.

1.2.5. Multiprogramming and Multiprocessing

Because most jobs require only a portion of the total system resources at any given instant (especially core memory), the MCP is able to initiate several jobs concurrently and assign them disjoint sets of resources. With only one or two central processors, only one or two of the jobs "in the mix" may be in execution at any given instant; however, the processor(s) may be assigned to different active jobs during brief, disjoint intervals of time. This ability to have many jobs (or programs) concurrently active is called multiprogramming. If a system contains two or more central processors and is able to execute simultaneously a job in each processor, it is said to be capable of both multiprocessing and multiprogramming. Furthermore, the sets of resources assigned to the active jobs need not necessarily be disjoint. Since assignments are made dynamically, they need only be disjoint with respect to time. This leads directly to the concept of "time-sharing" of resources, although the current discussion pertains to a feature of the batch processing MCP.

1.2.6. Automatic File Recognition

A practical necessity of the multiprogramming MCP is the authority to make specific peripheral device assignments. A desirable consequence is that the programmer need not, in fact can not, make specific unit assignments. The programmer need only specify the type of device and then address all program and data files by name only. For example, if a program needs to create a tape file, the MCP assigns an available tape unit and thereafter associates that unit with the programmer's choice of file name. Similarly, if a program requires some particular tape file for input, it is called by name only. The operator can mount the tape on any available unit and the MCP automatically recognizes the file name and makes the unit assignment.

1.2.7. Peripheral Device Independence

Automatic file recognition also facilitates the realization of true device independence. This permits magnetic tape and disk to be used as backup, or pseudo-devices, for the card readers, line printers, and card punch. Instead of forcing a job to wait for access to a busy peripheral device, the MCP automatically assigns a temporary backup device. This feature provides a considerable increase in efficiency and throughput.

1.2.8 Homogeneous Hardware Architecture

The hardware architecture which makes multiprogramming and dynamic resource allocation practical is centered around two exchanges--the memory exchange and the input-output (I/O) exchange. Each central processor can access any of the eight core memory modules through the memory exchange. Likewise any of the four floating I/O channels can access memory without interfering with the processors. In turn, the I/O channels can access any of the peripheral devices through the I/O exchange. Thus, both processors and all I/O channels could simultaneously access different modules of memory. The exchanges contain the logic necessary to float the access requests to a free I/O channel. Each I/O channel contains the logic and registers necessary to autonomously complete the information transfer once it has been initiated.

1.2.9. The Georgia Tech B5500 Configuration

The Georgia Tech configuration is as follows:

2	Central processing units
8	Core memory modules (4,096 words each)
4	Floating I/O channels
10	Magnetic tape units, 7-track, 200 and 556 BPI
3	Disk storage modules (1,200,000 words each)
2	Line printers (1100 lines per minute)
2	Card readers (1400 cards per minute)
1	Card punch (300 cards per minute)
1	Operator console (called SPO)
12	Telephone line adaptors and buffers

1.2.10. Hardware Stack

The machine language instructions of most computer systems consist of a command part and an address part--the command specifies the action and the address specifies the location of the operand. Instead of the usual format, the Burroughs hardware executes instructions expressed in "Polish notation," and utilizes a push-down stack. The stack consists of two registers and a contiguous area of core memory. Some instructions cause new operand values to be placed in the top of the stack (the registers) causing older values to be pushed down (into core memory). Other instructions remove the value at the top causing lower values to be automatically raised, thus keeping the two registers at the top effectively always full. Other instructions specify an arithmetic operation to be performed, where the operands are by definition the values in the top of the stack.

1.2.11. Polish Notation Machine Language

For example, the ALGOL assignment statement (just like FORTRAN except for the assignment operator :=)

D := (A + B)/C

when expressed in Polish notation becomes

AB+C/D:=

Polish notation eliminates the need for conventional rules of arithmetic precedence and bracket grouping of values within expressions. The rule that applies is: Follow two arithmetic values (the operands) with the operation (the command) that is to use those values. Thus, by definition, every mathematical operator works on the most recently obtained pair of operands which are always in the two registers at the top of the stack. The instructions needed to execute the assignment statement shown above are as follows:

Push the value of A into the top of the stack.

Push the value of B into the top of the stack.

Add, destroying the values of A and B and leaving (A+B) in the top of the stack.

Push the value of C into the top of the stack.

Divide, destroying the values of (A+B) and C and leaving the quotient in the top of the stack.

Store the top of the stack as the new value of D, leaving the stack empty.

1.2.12. Program Reference Table (PRT) and Reentrant Code

The B5500/B5700 machine language instructions may be placed anywhere in core memory and are fetched sequentially by a central processor for execution. The values of simple variables, pointers to the locations of multidimensional data arrays, pointers to subroutines, etc., are kept separately from the instructions and the stack in a contiguous area of core called the Program Reference Table (PRT). Notice that the instructions need only reference certain relative positions in the PRT (and the stack). This means that the same set of instructions could produce different results depending on the contents of the PRT. Such instructions are said to be reentrant, meaning that two or more jobs, each with its own separate PRT and stack, could be concurrently executing the same set of instructions. B5500 instructions, or code, are always reentrant, yielding great economy in the use of core memory.

1.2.13. Hardware Presence Bit Checking

A further economy in the use of core memory is accomplished by a unique hardware feature called the "presence bit." A pointer (descriptor) to each data array (and code segment) is kept in each job's PRT. When an instruction attempts to call a value from a data array into the stack (or branch to a new code segment), the central processor examines the presence bit in the descriptor. If the bit indicates the data array is currently located in core, the value is obtained and execution continues. If the presence bit indicates it is not present, the program is suspended and the MCP is automatically called. From the descriptor the MCP determines where the array is located on disk, copies the array into some available core area, places the address of this area in the descriptor, and turns the presence bit on. The same instruction is then reexecuted. By this means, all data arrays (and all code segments) for a job need not be present in core at the same time. This permits the MCP considerable latitude in allocating and deallocating core space as a function of time, thus minimizing the total core memory resource requirements.

1.2.14. Virtual Memory

The hardware presence bit and the MCP's ability to dynamically manage core space provide the programmer with a luxury called "virtual memory." This means that the total core requirements for a program may be far in excess of the actual core installed (32,768 words) and the program will still run. Furthermore, this is accomplished completely automatically by the system without programmer awareness. The MCP allocates and deallocates core space in exact area sizes demanded by the programs, and the hardware guarantees the data or code is actually present before it is referenced. The only remaining requirement is automatic program segmentation which is accomplished by the compilers, or language processors.

1.2.15. One-Pass Compilers

The function of a language processor is to convert a program written in some high-level language, such as ALGOL, BASIC, COBOL, FORTRAN, or GTL into machine language instructions and at the same time organize the data to be used as operands for those instructions. Depending on the nature of the machine language instructions and hardware organization, the compilation process can sometimes be quite tedious and time consuming, requiring many passes through the source language statements to fill in all the core

addresses and organize the data. The resulting object code is often not as efficient as it might have been if the programmer had used some lower level language, such as an assembly language (similar to machine language). However, the integrated hardware-software design of the B5500 permits extremely fast, one-pass compilers to usually produce the most efficient object code possible.

1.2.16. Automatic Program Segmentation

The compiler-oriented hardware only requires the compilers to produce machine language instructions in Polish notation, which is, itself, a high level language. The hardware push-down stack to automatically hold intermediate results, the hardware indirect addressing through the PRT, the hardware presence-bit checking, and the software in the MCP to manage core space, all contribute to further simplify the compilers' tasks. Each compiler contributes to core management efficiency by automatically segmenting each program on the logical boundaries of the language; for example ALGOL according to blocks, COBOL according to paragraphs, and FORTRAN according to subroutines.

1.2.17. System Software Written in ALGOL

With fast, efficient compilers there is no need for programmers to use anything but high level languages, such as ALGOL, GTL, FORTRAN, COBOL, or BASIC. Each different language has its own unique features and its own group of devotees. However, some features of ALGOL, such as the ability of a procedure (subroutine) to call itself recursively, coupled with the ability of the stack to hold a large number of intermediate results, make it an extremely attractive language in which problem solutions may be stated very concisely. For this reason, Burroughs chose ALGOL as the language in which to write all the system software which has evolved, for all practical purposes, without an assembly language. Not only is the MCP written in ESPOL (a special dialect of ALGOL) but all of the compilers, including the ALGOL compiler itself, are written in ALGOL. Thus, all the system software is quite concisely documented, but, nevertheless, quite efficient.

1.2.18. Comprehensive Accounting

A very necessary feature of the multiprogramming MCP is the ability to account for the resources used by each individual job that is processed. A very comprehensive logging system provides detailed information concerning processor time, I/O channel time, peripheral device usage, and disk space occupied. This detailed information is summarized and reported to Georgia Tech users in weekly Status Reports. The Status Reports show itemized charges incurred by each user for the current week plus the cumulative charges for the fiscal year.

1.3. Additional Features of the Time Sharing System

In addition to the features already described, the Time Sharing System contains two additional ones. The first is the use of the Time Sharing Master Control Program (TSMCP) instead of the Data Communications Master Control Program (DCMCP). The second is the Command and Edit (CANDE, pronounced candy) language processor.

1.3.1. Predictable Response Time of the TSMCP

The most serious disadvantage of the DCMCP is the fact that it processes remote jobs in the same manner as batch jobs. Its batch processing capabilities are excellent, providing high utilization of all resources and good throughput via multiprogramming. When the core memory requirements of the active batch jobs reach a certain threshold, the DCMCP refuses to initiate additional jobs in order to avoid an excessive overlay situation. A certain amount of core memory overlay is healthy, but an excessive amount is highly inefficient. For this reason the DCMCP is quite efficient over long periods of time and gives good response to the active jobs, but may force both batch and remote jobs to remain uninitiated for uncertain periods of time. Even a few minutes of complete inactivity can be most frustrating to a remote user. Thus, the unpredictable response time of the DCMCP to remote users' demands is its most serious deficiency. The TSMCP overcomes this deficiency by providing a predictable response time.

1.3.2. Time Slicing in the TSMCP

Each remote user desires fast and predictable responses. In fact, each remote user desires to have the exact kind and quantity of system resources made available to him whenever he needs them; i.e., to have sole use of the entire system. However ideal this might be from the viewpoint of the remote user, it is highly inefficient and impractical. The TSMCP attempts to create this sole-user illusion to many remote users by providing each user with reasonably fast, predictable, and equitable responses. Each remote job is executed for a short period, or slice, of time. It is then rolled out (swapped) from core memory to disk and the next job is rolled in and executed during its time slice. Therefore, each remote user receives his fair share of time, and within a brief cycle of time, all users receive a response from the system. All of the time required for the TSMCP to roll jobs in and out of core is, in many cases, wasted time. However, this inefficiency is the price that must be paid for predictable response time.

1.3.3. The Time Slicing Algorithm

The number of successive time slices required to complete a given task depends on the nature and magnitude of the task. Simple tasks, such as listing a source language statement or entering a data value to a program, can usually be completed during one time slice. More complex tasks, such as inverting a large matrix, may require many, many time slices. The total time required to cycle through all remote jobs and give each user a response depends on the number of users and whether each job utilizes its full time slice. When a job is first initiated (enters the mix), it is assigned to an area in core which minimizes conflict with other active jobs. It is then given an immediate time slice. If necessary, jobs in core are rolled out to make room for the new job, unless they are also getting their first time slice. In that case, the new job is placed at the head of the queue of jobs waiting for a time slice. There are five main conditions which cause a remote program to be rolled out:

(1) The program is input limited; i.e., it is waiting for data from the remote terminal.

(2) The program is output limited; i.e., it has generated enough data to fill the disk buffer area assigned to it.

(3) The program has used its time slice without becoming input or output limited.

(4) The program has reached completion.

(5) The program is forced out to make room for an entering or re-entering job.

1.3.4. Processing Philosophies of the DCMCP and TSMCP

This swapping feature of the TSMCP is the major difference between it and the DCMCP. The job processing philosophy of the DCMCP may be summarized as follows:

(1) Don't initiate a new job unless its initial core requirement can be satisfied, but initiate as many as possible.

(2) Once initiated, don't interrupt a job unless it needs assistance, such as an input/output operation, more core space, or job completion.

(3) Once interrupted, don't reassign all of a job's core space but only those areas that can be easily restored.

The job processing philosophy of the TSMCP may be summarized as follows:

(1) Always initiate a new remote job as soon as it is presented.

(2) Always interrupt a job at the end of its time slice.

(3) Always give each active job its time slice in cyclic order.

(4) Always make core space available for a new or reentering job, even if all areas assigned to older jobs need to be swapped.

(5) If none of the remote jobs can utilize their time slices during a cycle, then devote a time slice to a batch job.

From these two descriptions, it should be clear that the TSMCP, or any other time sharing system, is inherently inefficient. The TSMCP attempts to overcome some of this inefficiency by dividing core memory into two parts separated by a "fence." The area below the fence, in which only certain kinds of tasks are run, is managed according to the DCMCP philosophy, while the area above the fence is managed according to the TSMCP philosophy. Nevertheless, inefficiency does exist, but the compensation is the predictable response time required in any satisfactory time sharing system.

1.3.5. The Command and Edit Language Processor (CANDE)

The primary function of any operating system, such as any version of the MCP, is to process the jobs as they are presented. The MCP must be provided with certain information about each job, such as the name of a program to be executed, or which compiler is required, etc. Such information is supplied on control records in batch mode, and takes the form of commands to the MCP. Under the Time Sharing System, remote users converse with a program called CANDE, which accepts users' commands. Some of these commands are acted upon by CANDE, itself; others are passed to the TSMCP along with additional information supplied by CANDE. CANDE acts as the interface between the remote user and the rest of the system. One of CANDE's most useful functions is to assist in creating and editing source language and data files, prior to passing the file name to a compiler or program. Each CANDE command takes the form of a special verb which is usually followed by one or more parameters, some of which may be optional. The order in which various commands are entered by the user, and the options of particular commands, depend on what the user desires to accomplish. An overview of CANDE capabilities, along with the corresponding verbs, is presented in Section 2. The syntax of each verb is presented in alphabetical order in Section 4, accompanied by all the optional parameters and their meanings.

2. USING THE TIME SHARING SYSTEM

2.1 Overview of a Session

The B5700 Time Sharing System (TSS) can be accessed during specified hours of the day from a Teletype, or other similar terminal. Each remote user should consult his Departmental Computer Coordinator for the most current information on the hours of availability of TSS and the departmental procedures for using a terminal. The Departmental Computer Coordinators are also responsible for obtaining RECC Reference Numbers (account numbers), user-codes, and passwords and issuing them to each individual user.

A user begins a TSS session by dialing the proper B5700 telephone number and then, when requested, supplying his own personal user-code and password. If these are recognized as valid, he is logged on and may start to work. His session may terminate in a variety of ways, but usually by his entering BYE, in which case he is logged off and his terminal is disconnected. These attachment and detachment procedures are explained in detail in subsequent paragraphs.

Once a user is logged on, he is in communication with the Command and Edit Language Processor, called CANDE. CANDE is a conversational, multi-user program that acts as an interface between the user and the rest of the system. A user's specific requests are expressed in the form of CANDE commands. Each command takes the form of an English language verb which is usually followed by one or more parameters, some of which may be optional. There are about fifty CANDE verbs, each of which results in an action closely resembling its English language connotation. This resemblance, and the fact that some pairs of verbs are synonymous, make the CANDE language quite easy to learn. The many possible combinations of optional parameters add to its richness. The sequence in which a user enters commands and the optional parameters he specifies depend solely on what he desires to accomplish. The same end results can usually be achieved in a variety of ways; the particular choice is left to the knowledge, expertise, and style of the user.

2.2 Input Messages

Input to the system consists of messages entered from a remote terminal, usually from a keyboard or paper-tape reader. Each such message must be followed by an end-of-message indicator, which is a carriage return (CARR RET)

or a left arrow (←). After this indicator has been entered, the system normally responds with a carriage return and line feed which positions the print head for the next input message. For the purposes of this manual, it is considered advantageous to have a single printable character to represent the end-of-message indicator, and the left arrow will be used for this purpose. The system ignores line feed characters entered by the user. They may thus be used to enhance the readability of inputs to the system.

2.2.1. Backspacing

The apostrophe (') may be used as an equivalent backspace (the print head does not move backward) to delete the most recently typed character. Repeated use of the backspace deletes a corresponding number of characters, but only to the start of the current line. For example

```
MAKE  XYZ'2
```

would appear to CANDE as

```
MAKE XY2
```

2.2.2. Line Deletion

Prior to entering the left arrow, an entire line can be easily deleted by typing an exclamation point (!) After a delete is entered, the system types DEL and sends a carriage return and line feed to position the print head at the start of the next line. The delete can be used repeatedly; however only the current line can be deleted with the exclamation point. For example:

```
X = SQRT(X**2+Z!DEL
100X = SQRT(X**2+Z**2)←
```

2.3. CANDE Command Processing

As previously stated, most CANDE commands may be followed by a list of parameters. The parameters in the list must be separated from the verb and from each other by either commas or blanks. The choice is completely optional and need not be consistent within a given command.

When CANDE has finished processing a command, it types a message either confirming successful completion or indicating an error. After certain verbs, a number sign is typed to indicate successful completion. Errors terminate the processing of a command and the entire line must be retyped correctly. Between the time a command is entered and response is received from CANDE, the user should not normally enter anything from his terminal.

2.3.1. Compound Commands

If complete commands are separated by semicolons, more than one command may be entered on a single line. However, if an error is contained in a command, any further commands on that line are ignored.

2.4. CANDE Error Messages

CANDE error messages consist of the characters ERR: followed by a very terse error description, such as ERR:NOFILE. The user may obtain a more detailed explanation of any error by entering a question mark and left arrow. These detailed explanations are printed automatically if the HELPFUL option is set (see the SET and RESET verbs). Error messages are discussed along with the description of each verb in Section 4. Appendix D contains a complete alphabetical list of all CANDE responses.

2.5. Typographic Conventions

A Teletype terminal (or similar device) only prints upper case characters. Where upper case characters appear in the example, that exact sequence of characters is implied. Variable information is indicated by lower case characters, usually by hyphenated English words, such as "time-of-day".

2.6. Attachment Procedures

The attachment procedure is initiated by the user depressing the ORIG (originate) button, obtaining a dial tone, and dialing the B5700 telephone number. The telephone equipment automatically searches for a non-busy line. If all lines are busy, the user will hear a busy signal; he should depress the CLR (clear) button and try again later. If a non-busy line is available, the user will hear a ringing sound. When the computer accepts the call, the ringing changes to a short high-pitched tone (a beep) and a few seconds later the system will type

B5700 TIME SHARING - LINE logical-line-number

where logical-line-number is the system's recognition of the particular line that was reached.

On the next line, CANDE should begin the log-in sequence by typing

ENTER USER CODE, PLEASE-

The user should comply by entering his uniquely assigned user-code (followed, of course, by a left arrow).

The system will then respond with

```
AND YOUR PASSWORD
```

and will then black out seven spaces on the next line. The user should then enter his uniquely assigned password on the blocked out spaces (followed by a left arrow).

If the user-code and password do not agree with the current account file, the system types

```
BADCODE
ENTER USER CODE,PLEASE-
```

and the log-in procedure begins again. If the user does not log in successfully within three minutes, the terminal is automatically disconnected.

If the user-code and password are in agreement with the current account file, the system types the following message:

```
mm/dd/yy time-of-day
GOOD salutation, user-name: YOU HAVE LINE logical-line-number
```

The first line gives the date and time of day. The second line says "GOOD MORNING", or "GOOD AFTERNOON", or "GOOD EVENING", as appropriate, and is followed by the user-name that is associated with the user-code and password just supplied, typed exactly as it appears in the account file. The logical-line-number is repeated.

At this point the attachment and log-in procedures are complete and the user's session begins. He may now enter various CANDE commands and proceed to accomplish his desired tasks. More information about this part of the session will follow.

2.7. Detachment Procedures

When the user has completed his tasks, he must terminate the session in one of three ways. The first method is used when no other user is waiting to use the terminal and consists of entering the CANDE command

```
BYE
```

The system normally responds by typing

```
ON FOR time
C&E USE time
EXECUTE time
IO TIME time
OFF AT time-of-day
GOODBYE user-code
mm/dd/yy
```

and then spaces up about twelve lines and automatically disconnects the terminal. The first line of this accounting information is the elapsed time of the session; the second line shows the overhead processor time incurred by CANDE for this session; the third and fourth lines show the processor and IO times, respectively, used for processing CANDE commands, compiling, and executing programs; the fifth and seventh lines give the time of day and date when the session ends; the sixth line repeats the user-code that was supplied at log-in time.

The second method of terminating a session is to depress the EOT key on the keyboard (CTRL D). This causes the user to be logged off and his terminal to be immediately disconnected without printing any accounting information.

The third method of terminating a session is used when some other user is waiting to use the terminal. In this case, the first user should enter

```
HELLO
```

which causes the system to print the same accounting information as for the BYE command and to space the paper up about twelve lines. At this point, instead of disconnecting the terminal, the log-in sequence is initiated for the second user. Thus, the HELLO command accomplishes the same function as BYE, except that the telephone connection is maintained for the convenience of the next user.

2.8. Characteristics of CANDE Commands

After a user has successfully logged-in and his session has begun he may proceed to accomplish his desired tasks by entering appropriate CANDE commands. Every CANDE command, along with its parameters and possible error messages, is explained in detail in Section 4. The commands are arranged in

alphabetical order for easy reference. Appendix F also contains an alphabetical list of commands along with a concise explanation of their functions for quick reference. The user is expected to consult these two portions of the manual for details concerning each command. However, the major functions of some of the most frequently used commands will be introduced in the following paragraphs. In addition, some definitions of important terms and concepts will be presented. The CANDE commands will always appear in upper case characters; e.g., BYE, HELLO, CALL, DO, RUN, MAKE, etc.

2.9. Executing Programs from Public Library

A user may execute any program in the public library with the CALL command. A complete list of these programs is given with the description of the CALL command in Section 4. Most of these programs were designed to solve specific kinds of problems. For example, GPSS solves simulation problems, MATRIX performs matrix manipulations, ECAP analyzes electrical circuits, etc. However, two of these public library programs are different in that they permit almost any type of problem solution to be stated in their special languages and executed interpretively. They are named APL and WIPL. APL is a very powerful and concise language that provides many numerical and logical operators. WIPL is a very simple and easy to learn language that is quite useful for small problems and the interactive design of problem solutions. APL, WIPL, and most of the other public library programs are described in separate publications available at the Bookstore.

2.10. Executing User-Defined Programs

A user may execute a program in his own private library with the DO command. DO and EXECUTE are synonymous and may be used interchangeably. He may also EXECUTE a program from some other user's library provided the other user has given his permission. Access to other users files is controlled by a comprehensive file security system based on each user's uniquely assigned user-code. When originally created with the SAVE command, each file is called a sole user file, since only the creator can access it. All users can be given blanket permission to read from a file (or execute it) with the UNLOCK command; the creator still retains sole authority to modify its contents or REMOVE it. The PUBLIC command (not to be confused with the public library) permits any user to read or write or execute a file; only the creator can REMOVE it. Its creator may return an unlocked or public file to sole user

status with the LOCK command. A variation of the LOCK command permits only certain users or programs to access a file. The particular user-codes or program names and the desired degree of access are contained in a separate guard-file which is created with the GUARD command and attached to the file with the LOCK command.

2.11. Definition of File, File-name, and File-type

As implied above, a file is any collection of information which is regarded as a unit. It is the primary means by which a user establishes continuity between sessions. Each file in the system has a unique identification consisting of a file-name and a user-code. When a file is created, the user supplies the file-name and CANDE automatically adds his user-code to form the complete file identifier. File-names may be from one to six characters long. The first character must be a letter and any remaining characters must be either letters or digits in any combination. If the user supplies more than six characters, the rightmost characters are truncated. If a user, whose user-code is RCC1234, should create a file named PROG4, the complete file identifier would be PROG4/RCC1234.

In addition to a file-name, a file-type is also associated with every file in the system. A file-type must be one of the following: ALGOL, BASIC, COBOL, CODASYL, DYNAMO, FORTRAN, GTL, TSPOL, XALGOL, INFO, LOCK, SEQ, DATA. The first nine types specify that the file contains statements written in the corresponding programming language. The last four types specify that, in general, the file contains something other than program statements. A file whose file-type is INFO usually contains the output from a line initiated by a user with the SCHEDULE command; such jobs are to be run non-interactively at a later time. A file-type of LOCK is associated with a guard-file which contains particular user-codes and program names for file security purposes. File-types of INFO and LOCK are assigned automatically at creation time and are of little concern to the user.

File-types of SEQ and DATA must be specifically designated by the user and such files may contain any kind of information (including program statements). Files of type SEQ (as well as all other types except DATA) contain sequence-numbers within each record, whereas files of type DATA are not sequenced. A sequence number is a positive integer consisting of eight digits. All files consist of a number of 80 character records (one Teletype

line). When lines are being entered that are destined to become a record in some sequenced file (all except DATA), the sequence-number must appear as the first item on each Teletype line. The sequence-number specifies the desired position of the line in the file. Even if lines are typed out of order, CANDE arranges them in ascending numerical order. Although the sequence-numbers are actually stored in character positions 73 through 80 in each record and are used by CANDE for ordering and editing purposes, they are not considered to be part of the information in a sequenced file, except for programs written in the BASIC language. Thus, only 72 characters of information may be stored in each record of a sequenced file.

The records of files of type DATA do not contain sequence-numbers and permit all 80 characters to be referenced for any purpose. Such files might be created by a user program for subsequent editing by CANDE. They must contain 10 words per record and 300 words per block. Individual records in DATA files are referenced by their relative position in the file. The first record is record 1, the second is record 2, etc. When records are deleted from a type DATA file, the relative positions of the remaining records may change.

The file-type may be specified with the TYPE command when the file is being created. After creation, the file-type may be changed with the CHANGE command, although caution should be exercised. A variation of the CHANGE command also permits the file-name to be changed.

2.12. Definition of Work-file

The CANDE language allows a user to create or modify the contents of only one file at a time--that file is referred to as the work-file. A new work-file can be created with the LOAD command, but is more often created with the MAKE command. A user may LOAD a copy of the entire contents of some existing file into a newly created work-file or he may MAKE a new, but initially empty, work-file. Once a work-file is created, its contents may be edited with a variety of commands, which will be discussed later.

The contents of the work-file may be edited, compiled, or run just like any other file, with one exception. The contents of the work-file do not become permanent until the user enters a SAVE command. Thus, a work-file may be created, edited, used and then discarded if it is no longer needed. Similarly, a copy of an existing file can be modified and used without affecting the original file. In the latter case, the original file could be

permanently changed, if desired, by entering a SAVE command. By choosing a work-file file-name that is different from the original file-name, and then SAVEing the work-file, both versions can be retained. In other words, the work-file serves as a temporary scratchpad--its contents either can be explicitly retained with the SAVE command or can be automatically discarded by creating a new work-file or terminating the session.

2.13. Manipulating the Work-file

2.13.1. MAKE

If a user desires to write a new program in ALGOL (or any of the other programming languages) he would probably begin by entering a MAKE command similar to the following

```
MAKE ABC1 ALGOL←
```

where ABC1 becomes the file-name and ALGOL becomes the file-type of the work-file. Since this file-type must be a sequenced file, each line he enters must contain a sequence-number as the first item. After the left arrow terminating each line entered by the user, CANDE stores the line in his work-file and issues a carriage return and line feed indicating it is ready to accept the next line. The sequence-numbers of successive program statements are usually incremented by 100 to permit insertions at a later time. When an insertion is necessary, simply choose a sequence-number between those of the two desired lines, and CANDE will automatically place the new line in the proper position.

2.13.2. SEQ

When many sequentially-numbered lines are to be entered, it is more convenient to use automatic sequencing by entering the SEQ command. This causes CANDE to automatically type the next sequence-number (with a constant increment)--the user then need only enter his program statement as the remainder of the line. Optional parameters of the SEQ command permit the starting sequence-number and the increment to be specified. Automatic sequencing is terminated by entering a null line; i.e., just a left arrow.

2.13.3. FIX,DELETE

It quite frequently becomes necessary to replace or delete a line that has already been entered. To replace an existing line, terminate automatic sequencing (if being used) and enter the desired sequence-number and the new line. CANDE will replace the old line with the new one. The FIX command can be used to replace a portion of a line. The DELETE command can be used

to delete lines from the work-file or another file. Small groups of lines should be deleted individually.

2.13.4. PRINT, RESEQ

After entering several lines out of order, or making several changes or deletions, it may be desirable to obtain an ordered listing with the PRINT command. Variations permit a single line, a group of lines, groups of lines, or the entire work-file to be printed on the terminal in various manners (see LIST command syntax). The RESEQ command can be used to assign new constant increment-sequence-numbers, or to move records in a file.

2.13.5 WARNING

Much time can be either consumed or conserved depending on the relative order in which various CANDE commands are entered. As explained above, lines of the work-file may be entered, changed, or deleted in any order. However, CANDE does not necessarily process each editing command as it is entered. Instead, it stores many such commands and accomplishes the desired actions in one efficient, combined operation only when necessary. Commands which necessitate the complete reordering of the work-file, such as PRINT, should be used sparingly. For example, do not change a line, PRINT it, change another, PRINT it, etc. Instead make as many changes as possible before printing all of them. This requires that the work-file be placed in order only once rather than many times. The UPDATE command causes the work-file to be placed in order without printing it, but should be used only when necessary. The PRINT CHANGES option may be used to print the changes to the work-file since the last updating.

2.13.6. LOAD, ADD, MERGE, RMERGE, COPY

In the development of a new program it is sometimes desirable to use all or part of some existing program, or possibly bits and pieces from many different existing files. As already explained, the LOAD command permits a new work-file to be created from a copy of an entire file. Similarly, an entire file can be appended to the end of an existing work-file with the ADD or APPEND command. Any portion of a file may be inserted anywhere in the work-file with the COPY command, which also has many other useful variations. The MERGE and RMERGE commands permit part or all of another file to be merged by sequence number with the work-file. The FIND command is quite useful in determining the location of the first or all occurrences of an arbitrary character string in a file. This helps determine the parameters needed for subsequent COPY or MERGE commands.

2.13.7. REPLACE, FIND

The REPLACE command is used to locate some character string and replace it with another string. The FIND command may be used similarly to locate some character string in a file without replacement. The search may span an entire file or a specified portion of it. This provides a convenient manner, for example, in which to change the name of some program variable in all the statements in which it appears. However, caution must be exercised in specifying the sought-string, else damage may be done to the file that is difficult to repair.

2.13.8. COMPILE, DO, EXECUTE, RUN

After a source language work-file has been created and edited, the user is ready to have it compiled and executed. These two steps can be accomplished individually with the COMPILE and EXECUTE (or DO) commands, but are more easily combined in the simple RUN command. The RUN command requests CANDE to do whatever is necessary to execute the program. In the case of a newly created work-file, it means first, to use the appropriate compiler (as specified by file-type) to create an object code version of the program and, second, to immediately execute the object code (if it was successfully compiled).

If the compiler detects syntax errors, they will be printed on the terminal along with the corresponding sequence-number. If any syntax errors are found, no object code is produced and the implied execution contained in the RUN command is inhibited. The contents of the work-file are still intact, so after a compilation phase has ended, the user may correct any syntax errors by entering editing commands, just as he did when creating the work-file initially. He may alternately edit and RUN the work-file as often as necessary. When no syntax errors are detected, an object code file is produced and immediately executed. The user may now proceed to test the logic of his program. He may desire to RUN it several times in succession with different sets of test data. When a RUN command is entered and there have been no intervening changes to the work-file source language, the implied compilation is automatically inhibited. When changes are made to the source language, an existing object version is automatically deleted, thus indicating the need to recompile.

As described above, CANDE recognizes both versions of the work-file by its single file-name and uses the appropriate one in response to the RUN command. In contrast, the COMPILE and EXECUTE (or DO) commands specifically refer to the source language and object versions, respectively. If the user desires to retain the efforts of some session and enters a SAVE command, both source language and object code versions (if both exist) are saved under the single file-name of the work-file, such as ABC1. The file-type is also retained (ALGOL in the example). At his next session, the user need only enter

```
LOAD ABC1
```

and proceed as if no time had elapsed between sessions. This capability to distinguish between source and object versions and to remember the file-type are most convenient--the user need only remember a single file-name for each of his programs and CANDE chooses the proper version and compiler.

2.13.9. SAVE, RENAME, MONITOR

In the example above, the work-file (both source and object versions) were saved for use during a future session. During the course of developing a program it might become desirable to save intermediate copies during a single session. This is also accomplished with the SAVE command. However, since each file in the system must have a unique file identifier, the work-file's file-name must be changed with the RENAME command prior to entering each SAVE command. If this is not done, each intermediate copy will be permanently overwritten by the next one, leaving only the final copy. The MONITOR command can be used to retain a permanent record of all changes to the work-file. This type of audit trail can be quite useful when recalling the events of a busy session.

2.14. Other Commands

2.14.1. FILES, REMOVE, CHANGE

Although saving intermediate copies of a program can be quite useful, large amounts of disk storage space can be consumed. Each user should frequently and routinely perform housekeeping chores, called library maintenance, especially at the end of each session. The FILES command causes the file-name of each file created by the requesting user to be printed on his terminal. Since the source language and object code versions of the same program are stored in separate areas on disk, the same file-name may

be printed twice if both versions exist. The file-names of object code versions are preceded with an asterisk. Each user should examine this list of files and REMOVE those that he no longer needs. Variations of the REMOVE command permit source and object versions to be removed either independently or jointly. The CHANGE command permits the file-name of any permanent file to be altered (both source and object versions are changed jointly).

2.14.2. WHATS, LIST FILES, LOCK, UNLOCK, PUBLIC, GUARD

If the file-name as printed by the FILE command is not sufficient to remind the user of its contents, he may enter the WHATS command and obtain more details about a particular file. The LIST FILES command produces similar detailed information about many files. While performing library maintenance, the user may desire to alter the security status of his files with the LOCK, UNLOCK, PUBLIC, or GUARD commands, as previously explained.

2.14.3. PUNCH, TAPE, ?END

All users are urged to make use of the paper tape feature of most terminals. A copy of any source language file (either the work-file or a permanent file) can be punched into paper tape with the PUNCH command, and provides a more permanent and economical means of storing valuable or seldom used programs. When needed, a paper tape can be rapidly read into the work-file with the TAPE command. The ?END command is entered to terminate TAPE mode. Considerable time can be saved if a paper tape is prepared off-line prior to the beginning of a session. The tape format must be the same as that produced by the PUNCH command. While off-line from the computer, the lines of the work-file that would have been entered from the keyboard can be punched into tape. Typing mistakes can be corrected by overpunching with the rub-out character, by recording for later editing, or by regenerating the tape. The edited tape can then be read into the work-file very rapidly during an on-line session.

2.14.4. TO

Sometimes it becomes necessary, or desirable, for a user to communicate with the operator or some other active user. The TO command is used to transmit a one-line message of the sender's own composition. TO SPO sends it to the operator and TO user-code sends it to the terminal logged in with that user-code. Communication with the operator might be used to properly

identify or request magnetic tapes. Communication with some other user might be desirable to discuss programming details or to request permission to use his files, etc. When a message is received, the sender is identified so that he may send an answer.

2.14.5. SET, RESET, TYPE OPTIONS, BREAK, WRU

When a user is running a program, CANDE attempts to leave him alone until it terminates. This avoids printing some extraneous message in the middle of a neat tabular printout. However, if the user desires, he may override this feature in certain instances. For example, he may precede certain commands with a question mark (?) and force CANDE to accept them. He may also SET the ALLOWMSG option indicating that he is willing to receive any incoming messages. It might happen that a user specifically desires to terminate a program in execution. If the program is in a print loop, he may accomplish this by depressing the "BREAK" key on the keyboard. If the program is not printing, he should depress the "WRU" key.

The user may SET and RESET several other options at any time during a session. The HELPFUL option causes a full explanation of each error message to be automatically printed, just as if the user had entered a question mark to ask for more. Note that the question mark alone asks for an explanation of some error message, but a question mark preceding certain commands, such as ?TO, is used to force CANDE to accept the command at a time when it would normally ignore it. The CONCISE option inhibits the printing of many CANDE messages. The MONITOR option can be used to control the recording of work-file changes, and operates in conjunction with the MONITOR command.

2.14.6. EQUATE, LIST PROGRAM FILES

There are many occasions when a user desires to change certain attributes of program files. One very awkward method is to modify the source language and recompile the program. However, this may not always be possible and, in any event, is time consuming and undesirable. Usually, all that is desired is to change some attributes of a few files for only one run. For example, a new program can be more easily checked out by supplying small quantities of test data directly from the terminal. Large quantities of production data can be more easily read from another file which has been previously created and edited. The EQUATE command can be used immediately preceding a RUN, EXECUTE (or DO), or CALL command to temporarily change the

attributes of one or more files just for the duration of the following run. The internal program file-name may be temporarily equated to some other external file-name. In addition, the internal peripheral unit type may be temporarily changed to any other unit type, such as the remote terminal, the line printer, magnetic tape, disk, etc. The LIST PROGRAM FILES command may be used to determine the internal file attributes of any object version program to which file security will otherwise permit access. This is especially useful with respect to programs for which documentation is not conveniently at hand.

2.14.7. ?, SCHEDULE, STATUS, STOP

As stated previously, once a program is in execution CANDE will not accept further commands. The few exceptions are those that may be preceded by a question mark, such as TO (or SS), TAPE, DATA, END, or STATUS. This means that normally only one program at a time can be initiated from a terminal. Frequently this feature may be undesirable. For example, after initiating a lengthy production run it might be desirable to begin creating and editing another data file for a subsequent run. The SCHEDULE command permits this to be accomplished by instructing CANDE to process commands from a file (instead of from the terminal) and to place all its output in some other file. Thus, while the scheduled job is running the user may be performing other tasks. With the STATUS command, he may inquire about the progress of the scheduled job, or he may terminate it (where possible) with the STOP command.

3. EXAMPLE REMOTE TERMINAL SESSIONS

3.1 Introduction

The following example sessions are intended to illustrate several attributes of the B5700 Time Sharing System and CANDE. In several of them, the same problem is solved several different programming languages available on the B5700 to illustrate the techniques used in each language for handling remote terminal files. Others illustrate specific points. The remaining pages of this chapter are used to display a collection of sample sessions.

3.2 Sample Sessions

```
MAKE EXALG ALGOL←
FILE:EXALG - TYPE:ALGOL  -- CREATED
SEQ←
100BEGIN←
200FILE REM REMOTE(1,9);←
300FORMAT FMTOUT("  N = ",U,"AVERAGE = ",U);←
400INTEGER I,N;←
500REAL SUM,AVERAGE;←
600ARRAY A[0:50];←
700READ(REM,//,N,FOR I:=1 STEP 1 UNTIL N DO A[II]);←
800FOR I:=1 STEP 1 UNTIL N DO SUM:=SUM+A[I];←
900AVERAGE:=SUM/N;←
1000WRITE(REM,FMTOUT,N,AVERAGE);←
1100END.←
1200←
#
RUN←
 WAIT.


 COMPILING.
00000700:COL 46:II #100
UNDECLARED IDENTIFIER.


 ERR COMPILE 3.7 SEC.

*700/II/I←
RUN←
 WAIT.


 COMPILING.


 END COMPILE 3.2 SEC.

 RUNNING


?0←

-DIV BY ZERO,NEAR LINE 00000900


 ERR EXALG .3 SEC.

RUN←
 RUNNING


?9←
?1 2 3 4 5 6 7 8 9←
  N = 9 AVERAGE = 5.0


 END EXALG .5 SEC.
```



```
REMOVE←
#
```

```
CALL APL←
 RUNNING


       APL/B5500 UW COMPUTER SCIENCE # 12/22/71
        LOGGED IN MONDAY 03-13-72  07:04 PM
       $AVERAGE←
[1]    "N = ";(RHO X);"  AVERAGE = ";(+/X)%RHO X:=[]←
[2]    $←
       AVERAGE←
[]:
       1 2 3 4 5 6 7 8 9←
N =  9  AVERAGE =  5

       )OFF DISCARD←


 END APL 7.6 SEC.

REMOVE APLIBRY←
#
```

```
MAKE EXBAS BASIC←
FILE:EXBAS - TYPE:BASIC  -- CREATED
SEQ←
100INPUT N←
200FOR I=1 TO N←
300INPUT A(I)←
400NEXT I←
500FOR I=1 TO N←
600S=S+A(I)←
700NEXT I←
800A=S/N←
900PRINT "N = ";N;" AVERAGE = ";A←
1000END←
1100←
#
RUN←
 WAIT.


 COMPILING.


 END COMPILE 3.2 SEC.

 RUNNING


?9←
?1,2,3,4,5,6,7,8,9,←
N =        9     AVERAGE =        5


 END EXBAS .5 SEC.

REMOVE←
#
```

```
MAKE EXCOB COBOL←
FILE:EXCOB - TYPE:COBOL  -- CREATED
SEQ←
100IDENTIFICATION DIVISION.←
200PROGRAM-ID. COBOL EXAMPLE.←
300ENVIRONMENT DIVISION.←
400CONFIGURATION SECTION.←
500SOURCE-COMPUTER. B-5700.←
6000BJECT-COMPUTER. B-5700.←
700INPUT-OUTPUT SECTION.←
800FILE-CONTROL.←
900 SELECT REMIN ASSIGN TO REMOTE.←
1000 SELECT REMOUT ASSIGN TO REMOTE.←
1100DATA DIVISION.←
1200FILE SECTION.←
1300FD REMIN←
1400 DATA RECORD REMIN01.←
150001 REMIN01.←
1600 02 RI PICTURE 99 OCCURS 40.←
1700FD REMOUT←
1800 DATA RECORD REMOUT01.←
190001 REMOUT01.←
2000 02 NVAL PICTURE ZZZZZ99.←
2100 02 FILLER PICTURE X(12).←
2200 02 RESLT PICTURE 99.99.←
2300WORKING-STORAGE.←
240077 N PICTURE 9(8) COMPUTATIONAL-1/'.←
250077 I PICTURE 9(8) COMPUTATIONAL-1.←
260077 SSUM PICTURE 9(8) COMPUTATIONAL-1.←
2700PROCEDURE DIVISION.←
2800START.←
2900 OPEN INPUT REMIN OUTPUT REMOUT.←
3000 READ REMIN AT END STOP RUN.←
3100 EXAMINE REMIN01 REPLACING ALL SPACES BY ZEROES.←
3200 MOVE RI(N'1) TO N.←
3300 MOVE "N =       AVERAGE =       " TO REMOUT01.←
3400 MOVE ZERO TO I SSUM.←
3500L1.←
3600 ADD 1 TO I.←
3700 IF I IS 0'NOT GREATER THAN N ←
3800 ADD 1 TO !DEL
3800 ADD RI(I + 1) TO SSUM←
3900 GO TO L1.←
4000 MOVE N TO NVAL.←
4100 DIVIDE N INTO SSUM GIVING RESLT.←
4200 WRITE REMOUT01.←
4300 STOP RUN.←
4400END-OF-JOB.←
4500←
#
RUN←
 WAIT.
```

```
 COMPILING.
**ERROR @002400: SYNTAX ERROR


 END COMPILE 7.5 SEC.

 RUNNING


? 9 1 2 3 4 5 6 7 8 9←
     09  AVERAGE = 05.00


 END EXCOB .7 SEC.

1950 02 FILLER PICTURE ZZZZZ.←
2'*2000/ZZZZZ/←
*2300/./ SECTION.←
RUN←
WAIT.


 COMPILING.
?←
YOUR PROGRAM IS BEING COMPILED.


 END COMPILE 7.4 SEC.

 RUNNING


? 9 1 2 3 4 5 6 7 8 9←
N =  09  AVERAGE = 05.00


 END EXCOB .6 SEC.

REMOVE←
#
```

```
MAKE EXFOR FORTRAN←
FILE:EXFOR - TYPE:FORTRAN  -- CREATED
SEQ←
100FILE  5=REMIN,UNIT=REMOTE,RECORD=9←
200FILE  6=REMOT,UNIT=REMOTE,RECORD=9←
300REAL A(50)←
400READ(5,/)N,(A(I),I=1,N)←
500DO 10 I=1,N←
60010 SUM=SUM+A(I)←
700AV=SUM/N←
800WRITE(6,500)N,AV←
900500 FORMAT("  N = ",I2,"  AVERAGE = ",F5.2);'←
1000STOP←
1100END←
1200←
#
RUN←
 WAIT.


 COMPILING.


 END COMPILE 3.8 SEC.

 RUNNING


?9, 1,2,3,4,5,6,7,8,9,←
  N =  9  AVERAGE =  5.00


 END EXFOR .5 SEC.

REMOVE←
#
```

```
MAKE EXGTL GTL←
FILE:EXGTL - TYPE:GTL  -- CREATED
SEQ←
100BEGIN FILE REMOTE;←
200INTEGER I,N;←
300REAL SUM,AVERAGE;←
400READ[N];←
500FOR I:=1 STEP 1 UNTIL N DO ←
600SUM:=SUM+A[!DEL
600SUM:=SUM+READN(TWX);←
700AVERAGE:=SUM/N;←
800PRINT #N = # N # AVERAGE = # AVERAGE;←
900END.←
1000←
#
RUN←
 WAIT.


 COMPILING.


 END COMPILE 6.6 SEC.

 RUNNING



N =
?9←
?1 2 3 4 5 6 7 8 9←
N = 9 AVERAGE = 5

 END EXGTL 1.0 SEC.

REMOVE←
#
```

```
CALL WIPL←
 RUNNING

PLEASE WAIT.
TYPE HELP IF YOU HAVE QUESTIONS  (WIPL VERSION 1.5)
?1.0 DIMENSION A[50]←
?1.01 ACCEPT N←
?1.02 DO PART 2, FOR I=1,N←
?1.03 SUM=0←
?1.04 DO PART 3, FOR I=1,N←
?1.05 AVERAGE=SUM/N←
?1.06 PRINT FORM 1.9,N,AVERAGE←
?1.07 STOP←
?1.9 FORM  N = &&  AVERAGE = &&.&&←
?2.0 ACCEPT A[I]←
?3.0 SUM=SUM+A[I]←
?RUN←
N=?
   ?9←
A[1]=?
      ?1←
A[2]=?
      ?2←
A[3]=?
      ?3←
A[4]=?
      ?4←
A[5]=?
      ?5←
A[6]=?
      ?6←
A[7]=?
      ?7←
A[8]=?
      ?8←
A[9]=?
      ?9←
 N =  9  AVERAGE =  5.00
STOP AT STATEMENT 1.07
?QUIT←

 END WIPL 7.1 SEC.
```

```
CALL DERIV←
 RUNNING


DERIVATIVE-TAKING PROGRAM.
ENTER HELP FOR ASSISTANCE
?X*2+Y*2←
TOS: ((X*2)+(Y*2))
?DERIV X←
TOS: ((X*2)\(2/X))
?SET X=4←
X=4
?EVAL←
8
?FOR X:=0 STEP 1 UNTIL 10 DO←
0     0
1     2
2     4
3     6
4     8
5     10
6     12
7     14
8     16
9     18
10    20
RESULT MAY BE INCORRECT BECAUSE OF THE FOLLOWING ERRORS:
   DIVISION BY ZERO
?PLOT X:=0 STEP 1 UNTIL 10 DO←
•
•
•
•
•
•
•• RANGE IS 0 TO 20
0          0          *
1          2          0     *
2          4          0     1     *
3          6          0     1     2     *
4          8          0     1     2     3     *
5          10         0     1     2     3     4     *
6          12         0     1     2     3     4     5     *
7          14         0     1     2     3     4     5     6     *
8          16         0     1     2     3     4     5     6     7     *
9          18         0     1     2     3     4     5     6     7     8     *
10         20         02468024680246802468024680246802468024680246802468*
•
•
•
•
•
•
RESULT MAY BE INCORRECT BECAUSE OF THE FOLLOWING ERRORS:
   DIVISION BY ZERO
```

```
?SIN(X)\COS(Y)←
TOS: (SIN (X)\COS (Y))
?PUSH←
?DERIV X←
TOS: (COS (Y)\COS (X))
?DEFIV'''RIV Y←
TOS: (COS (X)\(0-SIN (Y)))
?EVAL←
Y=
?1.57←
.83907
?POP←
4    (COS (Y)\COS (X))
?DERIV Y←
TOS: (COS (X)\(0-SIN (Y)))
?DERIV X←
TOS: ((0-SIN (Y))\(0-SIN (X)))
?EVAL←
-.54402
?CONTOUR X Y←
ENTER BASE,INCREMENT,MAXIMUM VALUES FOR X, SEPARATED BY COMMAS
?-3.14,.1,3.14←
TOO MANY ITERATIONS INDICATED:
FOR X:=(-3.14,.1,3.14)
?SIN(X)*2+COS(Y)*2←
TOS: ((SIN (X)*2)+(COS (Y)*2))
?CONTOUT'R X Y←
ENTER BASE,INCREMENT,MAXIMUM VALUES FOR X, SEPARATED BY COMMAS
?-3.14,.2,3.14←
ENTER BASE,INCREMENT,MAXIMUM VALUES FOR Y
?0.,.2,6.28←
ERROR AT .
?LIST←
TOS: 0
?SIN(X)*2+COS(Y)*2←
TOS: ((SIN (X)*2)+(COS (Y)*2))
?CONTOUR X Y←
ENTER BASE,INCREMENT,MAXIMUM VALUES FOR X, SEPARATED BY COMMAS
?-3.14,.2,3.14←
ENTER BASE,INCREMENT,MAXIMUM VALUES FOR Y
?-3.14,.2,3.14←
ARGUMENT OF LN LE@ 0
?POP←
6    3.14
?POP←
5    (COS (X)\(0-SIN (Y)))
?POP POP POP POP POP←
4    (COS (Y)\COS (X))
3    (SIN (X)\COS (Y))
2    (SIN (X)\COS (Y))
1    ((X*2)\(2/X))
0    ((X*2)+(Y*2))
?SIN(X)*2+COS(Y)*2←
TOS: ((SIN (X)*2)+(COS (Y)*2))
?SIN(X)+COS(Y)←
TOS: (SIN (X)+COS (Y))
?CONTOUR X Y←
ENTER BASE,INCREMENT,MAXIMUM VALUES FOR X, SEPARATED BY COMMAS
?-3.14,.2,3.14←
ENTER BASE,INCREMENT,MAXIMUM VALUES FOR Y
?-3.14,.2,3.14←
```

```
.
.
.
.
.
.
.. RANGE IS -1.9995 TO 1.9942
.
          0    1    2    3    4    5    6    7    8    9    0
          02468024680246802468024680246802468024680246802468024680
           0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ[(&$*)/%=]#@:+
.
-3.14     B974310   01357ACFHJLMNOOONLJHFD
-2.94     C975310  002358ADFHJLNOOOONLKIFD
-2.74     CA8542100012468BDGIKMNOPPPOMKIGE
-2.54     EB97532112245 7ACEHJLNPQQQQPNMKHF
-2.34     FDA865433345 79BEGJLNPQRSSRQPNLJG
-2.14     HFCA876555679BDGILNPRSTUUTSRPNLI
-1.94     JHFCB9877789BDFIKNPRTUVWWWVTRPNL
-1.74     MJHFDBAA9ABCDFIKNPRUVXYYYYXWUSPN
-1.54     OMKHFEDCCCDEGIKNPSUWYZ[(((ZYWUSP
-1.34     ROMKIGFFEFGHIKNPSUWZ[&$$$$&(ZXUS
-1.14     TROMKJIHHHIJLNPSUWZ($*)//)*$(ZXU
-.94       VTROMLKJJJKLNPRUWZ($)/%==%/)$(ZW
-.74       XVSQONMLLLMNPRTWY[$)%=]##]=%)$(Y
-.54       ZWUSQONMMNNPQSVX[&*/=#@@@@#=%)&[
-.34       [XVTRPONNOOQRTWY($)=]@::::@]=/$(
-.14       [YWTRQPOOOPQSUWZ(*/=#@:+++:@#=/*(
6@-2       (YWTSQPOOOPQSUWZ(*/=#@:+++:#=/*&
.26        [YVTRQOOOOPQSUWY($/=#@:++:@#=/*(
.46        ZXUSQPONNNOPRTVY[&)%]#@::@#]%)$[
.66        YVTRPNMMLMNOPRUWZ($/%]####]=/*(Z
.86        WURPNMLKKKLMOQSUXZ&*/%=]]=%/*&[X
1.06       USPNLKIIIIJKMOQSVX[&*)/%%/)*&[YV
1.26       SPNLJHGGFGGIJLOQTVX[($****$&[YVT
1.46       PNLIGFEDDDEFHJLOQTVXZ[(&&([ZXVTQ
1.66       NKIGECBBABCDEGJLOQSVWYZZZZYXVTQO
1.86       KIGDBA98889ACEGJLOQSUVWXXWWUSQOL
2.06       IGDB98666678ACEGJLOQSTUVVUTSQOMJ
2.26       GEB97644445 68ACEHJMOQRSTTSRQOMKH
2.46       EC975432223468ADFIKMOPQRRQPOMKIF
2.66       DA86421 10123479BEGILMOPPPPONLJGE
2.86       C9753100 012468ADFIKLNOOOONMKIFD
3.06       C974310   01357ACFHJLMNOOONLJHFD
.
.
.
.
.
.
?STOP←

 END DERIV 18.5 SEC.
```

4. DETAILED CANDE COMMANDS

4.1. General

The commands available with the Burroughs B5700 Time Sharing System are described in detail in this section. These commands give the computer directions as to specific actions which must be executed in order to perform the required tasks. The commands are presented in alphabetical order and in a modular format. The first subsection here displays the typographic conventions and the second subsection displays the CANDE commands themselves.

4.2. Typographic Conventions

The notation conventions described below are used throughout the remainder of the manual to describe the syntax of each command.

4.2.1. Notation Used in Verb Syntax Formats

First, the syntax of a command is presented. This gives the user a concept of the potential power of the command. For example,

```
(C|COMPILE)[file-name][compiler-file-type|:first-letter-of-compiler-
    file-type]program-parameter-info
```

4.2.1.1. Parentheses

Information that is enclosed within parentheses indicates that a choice must be made between the entries. In the example given above, a selection must be made between C and COMPILE.

4.2.1.2. Vertical Bars

Entries separated by vertical bars indicate that a choice is to be made between them. In the example above, a choice is to be made between C and COMPILE.

4.2.1.3. Brackets

Information that is enclosed within brackets indicates that a choice may be made between the entries, but not necessarily. In the example given above, a selection may be made between compiler-file-type and first-letter-of-compiler-file-type, or the option may be ignored.

4.2.1.4. Lower Case Letters

Words that appear in lower case letters indicate that a value must be supplied. In the example given above, the word file-name indicates that a value of one to six characters must be supplied, if the option is chosen.

4.2.1.5. Upper Case Letters

Words that appear in upper case letters indicate that the word is a literal and must be substituted verbatim, when used. In the example given above, the word COMPILE or the letter C would be entered first when constructing this command statement.

4.2.1.6. Ellipsis Periods

Ellipsis periods denote the occurrence of the immediately preceding syntactical item one or more times. For example,

```
VERB[file-name ...]
```

indicates that the command VERB may be followed by a file-name which in turn may or may not be followed by one or more file-names to form a longer list.

4.2.1.7. Underlines

Normally, in the examples in this manual, a line followed with a ← is assumed to be entered in its entirety by the user, and a line not terminated with a ← is assumed to be typed by the system. In the cases in which part of a line is entered by the user and part is typed by the system, that part typed by the system is underlined only if otherwise ambiguous. In most cases, it is not ambiguous or confusing, and underlines will not normally be used.

4.2.1.8. Concatenation

Terms written adjacent to each other denote the incidence of each represented term in the order they appear in the command. Spaces are required to separate alphanumerical and/or numerical terms. Commas are allowed to separate parameters.

4.2.2. Examples of Use of Notation

The following represents an expansion of the example syntax presented earlier:

```
COMPILE
COMPILE f
COMPILE c
COMPILE f c
COMPILE :c1
COMPILE f:c1
COMPILE p
COMPILE f p
COMPILE c p
COMPILE f c p
COMPILE :c1 p
COMPILE f:c1 p
```

where f is the file-name,

c is a compiler-file-type,

c1 is the first letter of a compiler-file-type,

p is program-parameter-info.

The term COMPILE may be abbreviated to C, thus doubling the above list of possible variations of this command.

4.2.3. Definitions of Important Terms

The following definitions illustrate the use of the notation used here.

4.2.3.1. Sequence-list

Sequence-list is defined to be

[(sequence-number-1|END)[(TO|-)(sequence-number-2|END)]] ...

where the sequence-numbers are in ascending order from left to right.

4.2.3.2. Resequence-info

Resequence-info is defined to be

[sequence-list[base-sequence-number][+resequence-increment]]

4.2.3.3. Program-parameter-info

Program-parameter-info is defined to be

[WITH[(STACK|PROCESS|IO|COMMON) = integer] ...]

4.2.3.4. Examples of Use of Definitions

Following the definitions given above, some of the possible combinations may be enumerated as follows:

sequence-list:

```
(empty)
s1
s1 TO s2
s1 TO END
s1 - s2
s1 - END
s1,s2
s1,s2 TO s3,s4
s1,s2,s3 - END
:
:
```

where s1, s2, s3, s4 represent sequence-numbers and $0 \leq s1 < s2 \leq s3 \leq s4 < 100000000$,

resequence-info:

```
(empty)
s1
s1 TO s2
s1 - s2, s3
s1 + i1
s1, s2, s3 - END, s4 + i1
+ i1
:
```

where s1, s2, s4 are as before and 0<i1<50000000, where i1 represents an increment,

program-parameter-info:

```
(empty)
WITH  STACK = s1
WITH  PROCESS = p1
WITH  IO = i1
WITH  COMMON = c1
WITH  PROCESS = p1, IO = j1
WITH  COMMON = c1, STACK = s1, PROCESS = p1
:
```

where s1 represents stack size (512≤s1≤4096),

p1 represents processor time limit in minutes (0≤p1≤10000),

j1 represents I/O channel time limit in minutes (0≤j1≤10000),

c1 represents common value (0≤c1≤99999999).

Some specific examples are as follows:

sequence-list:

```
(empty)
100
200 - 900
1 - END
450 - 1500, 2000 TO END
100, 300, 500, 1000 - END
100, 200 - 1000, 2000, 3000 - 5000, 7000 - 9000
```

resequence-info:

```
(empty)
100
100 + 100
+ 200
100 - 1000, 2000 + 10
1000 - 2000, 14000, 15000 + 1000
```

program-parameter-info:

```
(empty)
WITH  STACK = 1000
WITH  PROCESS = 10, IO = 5
WITH  STACK = 1000, COMMON = 2, IO = 1
WITH  IO = 1, STACK = 200, PROCESS = 2
```

4.3. CANDE Commands

Following are the CANDE commands, arranged in alphabetical order, for reference purposes.

4.3.1. ADD|APPEND

Records from a given file may be copied onto the end of the existing work-file with the ADD or APPEND command. The format is as follows:

(ADD|APPEND) file-name [/user-code] sequence-list [RESEQ resequence-info]

If no sequence-numbers are specified, the entire file is appended. Otherwise, only the specified portions are added. If the file is not of type DATA, the appended lines are given sequence-numbers equal to their old sequence-numbers plus the highest sequence-number originally in the work-file, unless the RESEQ option is used.

If the RESEQ option is specified, the base and increment for resequencing are assumed by default to be 100, and moving of records (RESEQ 100,300) is not permitted.

If the sequencing process for a non-file-type DATA file would result in a sequence-number of more than eight digits, the following error message

ERR:TOOBIG

is typed, indicating an invalid sequence-number, and the remaining lines are not APPENDed.

File Adjustment

The CANDE APPEND command provides for proper file adjustment depending upon the file-types of the work-file and input file. The different types of file adjustment, described in the following paragraphs, are as follows:

Work-file is a sequential file, and the input file is a type DATA file.

Work-file is a sequential file, and the input file is a sequential file.

Work-file is a type DATA file.

Work-file : Sequential File - Input file : Type DATA File

If the RESEQ option is not specified, the APPEND command attempts to convert the characters in the input file in record positions 73 through 80 into a sequence-number. If this would otherwise result in an improperly-sequenced file, error messages are given and the sequence-numbers are adjusted, if possible; otherwise, the operation is terminated. If only a portion of the

input file is to be copied, the records to be copied are referenced by their position in the file, not by sequence-number. The sequence-numbers of the records APPENDed to the work-file are obtained by adding the values of the sequence-numbers produced by the APPEND command to the sequence-number of the last record in the work-file prior to execution of the APPEND command.

Work-file : Sequential File - Input File : Sequential File

If the RESEQ option is not specified, the APPEND command expects to find sequence-numbers in the input file in record positions 73 through 80. If the input file is not properly sequenced, error messages are given, and the sequence-numbers are adjusted, if possible; otherwise, the operation is terminated. If only a portion of the input file is to be APPENDed, the records to be APPENDed are referenced by their sequence-numbers.

Work-file : Type DATA File

When the work-file has file-type DATA, the APPEND command does not examine the input file sequence field, but transfers the records as 80-character units onto the end of the work-file. If the input file is, in fact, a sequential file, an 8-digit sequence-number is normally present in character positions 73 through 80, though this information is ignored by the APPEND command.

```
MAKE FILE1←
FILE:FILE1 - TYPE:SEQ  -- CREATED
SEQ←
100RECORD ONE←
200RECORD 2←
300RECORD THE'REE←
400RECORD 4←
500RECORD 5←
600RECORD SIX←
700←
#
SAVE←
 WAIT.


FILE:FILE1 - TYPE:SEQ  -- SAVED.

MAKE FILE2←
FILE:FILE2 - TYPE:SEQ  -- CREATED
SEQ←
100REC 1←
200REC 2←
300REC 3←
400REC 4←
500REC 5←
600REC 6←
700←
#
APPEND FILE1←
 WAIT.


 WAIT.
6 RECORDS APPENDED (LAST RECORD APPENDED=1200)


 END APPEND 1.4 SEC.

P←
100 REC 1
200 REC 2
300 REC 3
400 REC 4
500 REC 5
600 REC 6
700 RECORD ONE
800 RECORD 2
900 RECORD THREE
1000 RECORD 4
1100 RECORD 5
1200 RECORD SIX


#
```

```
ADD FILE1 400 TO END RESEQ 100 TO END + 10←
 WAIT.
3 RECORDS APPENDED (LAST RECORD APPENDED=1320)


 END APPEND 1.3 SEC.

P←
100 REC 1
200 REC 2
300 REC 3
400 REC 4
500 REC 5
600 REC 6
700 RECORD ONE
800 RECORD 2
900 RECORD THREE
1000 RECORD 4
1100 RECORD 5
1200 RECORD SIX
1300 RECORD 4
1310 RECORD 5
1320 RECORD SIX


#
```

4.3.2. BYE

The BYE command terminates a user's session. The format is as follows:

BYE

After the user types BYE, the system normally responds with statistics concerning the user's current session, and then performs a terminal disconnect. The user may also log-out by entering a HELLO command or by striking the EOT character (CTRL D); in this case, the system response cannot be given. If the terminal is idle for five minutes, the system will assume that a BYE command has been entered.

The system response, if any, contains the following information:

elapsed time of session,
overhead processor time incurred by CANDE during session,
processor time used by programs during session,
I/O channel time used by programs during session,
time of day at end of session,
date at end of session.

```
BYE←
 ON FOR  44 MIN, 02.6 SEC.
 C&E USE 24.4 SEC.
 EXECUTE 25.9 SEC.
 IO TIME 58.7 SEC.
 OFF AT   7:50 PM.
 GOODBYE RCC63YH
03/06/72
```

4.3.3 CALL

The remote user may execute a program in the public library through the use of the CALL command. The format of the CALL command is as follows:

CALL program-name[/suffix] program parameter-info

Currently the following programs are in the public library:

APL	B5700 version of APL language
CIRCUS	electrical circuit simulator
CONTROL	enters MCP control records
CXREF	conversational manual-writing system
DERIV	symbolic differentiation and evaluation
ECAP	electrical circuit analysis
ELIZA	psychiatric examination program
FCTNS	special function evaluation language
GPSS	B5700 version of GPSS/360
MATRIX	matrix manipulation
MIX	B5700 version of MIX Assembly Language
POLY	polynomial manipulation
OL	lists labels of mounted tapes
REFORM	ALGOL and GTL source file reformatter
SEARCH	interactive LIST FILES program
SIMULA	B5700 version of SIMULA/67
TAPDUMP	interactive tape file list to remote terminal
WIPL	WIPL language

In case the user decides to prematurely discontinue a program, and the program is not typing at the time, he may depress WRU(CTRL D). If the program is typing at the time, it may be necessary to depress BREAK to stop the typing; if BREAK itself does not discontinue the program, he may then depress WRU.

The CALL verb may be preceded by EQUATE commands or immediately followed by program-parameter-info. If no program-parameter-info occurs, the following parameters are assumed:

PROCESS = (maximum allowed for user)
IO = (maximum allowed for user)
STACK = 512
COMMON = 0

The term suffix is required only when the program in the public library has a second name not equal to "TSHARER".

```
CALL MATRIX←

ENTER I-O FILE NAMES

?TE TE←
ENTER PROGRAM NAME

?INVERT←
INVALID ANSWER INVERT
ENTER PROGRAM NAME

?EX←

THE PROGRAMS ARE
1. LINEQN
2. DETLINEQN
3. LINEQNIMPRV
4. DETLINEQNIMPRV
5. INVERSE
6. INVWITHIMPRV
7. EIGENVALUES
8. EIGENVECTOR
9. DETOFA
PLEASE USE FULL NAME
ENTER PROGRAM NAME

?INVERSE←
ENTER N

?3←

N =  3
```

CALL

```
ENTER ROW  1

?1 2 3←
ENTER ROW  2

?3 2 1←
ENTER ROW  3

?2 3 1←
CHANGES OR DISPLAY DESIRED?

?NO←


INVERSE MATRIX

N =  3

     -0.08333333        0.58333333       -0.33333333

     -0.08333333       -0.41666667        0.66666667

      0.41666667        0.08333333       -0.33333333

DETERMINANT =       12.0000000001


RUN AGAIN?

?NO←

 END MATRIX 2.5 SEC.
```

4.3.4. CC

The remote user may change the parameter which CANDE uses to determine the length of a station's carriage. The format of the CC command is the following:

```
CC (SHORT|LONG)
```

Since the system assumes that each teletype or equivalent station has a standard carriage length of 72 characters, the CC command allows a user to notify the system that the carriage is not SHORT (standard 72 character length), but is LONG (greater than 72 characters in length).

When the system attempts to send a message to the terminal which is longer than 72 characters, and the carriage is SHORT, the message will be split into segments each shorter than 72 characters. No such splitting is performed if the carriage is LONG. Also, for LONG carriages, the proper timing is invoked such that the mechanism should never print "on the fly," even for an extremely long line.

```
CC SHORT←
#
CC LONG←
#
```

4.3.5. CHANGE FACTOR

The CHANGE FACTOR command allows the user to alter the save-factor which is associated with a user's disk file. The format of the CHANGE FACTOR is as follows:

CHANGE [SOURCE|OBJECT] file-name FACTOR TO integer

The SOURCE or OBJECT option may be used to specify that only the source or the object version's save-factor is to be changed. If neither SOURCE nor OBJECT options appear after the CHANGE portion of the command, the save-factors of both the source and object files are altered, if they are present.

```
WHATS G22←
FILE G22, TYPE GTL, 18 RECORDS, CREATED 03/02/72 (1944) SF=7
#
WHATS OBJECT G22←
FILE OG22, TYPE GTL, 21 RECORDS, CREATED 03/02/72 (1944) SF=8
#
CHANGE G22 FACTOR TO 50←
#
WHATS G22←
FILE G22, TYPE GTL, 18 RECORDS, CREATED 03/02/72 (1944) SF=50
#
WHATS OBJECT G22←
FILE OG22, TYPE GTL, 21 RECORDS, CREATED 03/02/72 (1944) SF=50
#
CHANGE SOURCE G22 FACTOR TO 10←
CHANGE OBJECT G22 FACTOR TO 20←
#
#
WHATS G22←
FILE G22, TYPE GTL, 18 RECORDS, CREATED 03/02/72 (1944) SF=10
#
WHATS OBJECT G22←
FILE OG22, TYPE GTL, 21 RECORDS, CREATED 03/02/72 (1944) SF=20
#
```

4.3.6. CHANGE

The CHANGE command allows a user to change a file's file-name. The format of the CHANGE command is the following:

CHANGE file-name TO file-name

When this command is used, both source and object versions of the designated file (if they exist) will have the old file-name changed to the new file-name. In order to change the file-name of the work-file, the RENAME command must be used.

```
CHANGE GTW02 TO G22←
#
CHANGE GTW02 TO G12453←
ERR: CANNOT
?←
FILE NOT IN YOUR LIBRARY.
```

4.3.7. CHANGE TYPE

The CHANGE TYPE command allows a user to change a file's file-type. The format of the CHANGE TYPE command is as follows:

CHANGE[file-name]TYPE[TO]file-type

This command causes the file-type of the source version of the designated file or work-file to be changed. The object version's file-type is always associated with the compiler used for its creation, and normally cannot be changed. When the file-name term is omitted, CHANGE TYPE may be shortened to TYPE.

The file-type of a file may be changed from DATA to sequential and vice versa. A work-file's file-type may be changed from sequential to DATA only.

```
WHATS SEARCH←
FILE SEARCH, TYPE GTL, 38 RECORDS, CREATED 03/02/72 (1210) SF=7
#
WHATS OBJECT SEARCH←
FILE OSEARCH, TYPE GTL, 67 RECORDS, CREATED 03/02/72 (1210) SF=8
#
CHANGE SEARCH TYPE TO ALGOL←
#
WHATS SEARCH←
FILE SEARCH, TYPE ALGOL, 38 RECORDS, CREATED 03/02/72 (1210) SF=7
#
WHATS OBJECT SEARCH←
FILE OSEARCH, TYPE GTL, 67 RECORDS, CREATED 03/02/72 (1210) SF=8
#
LOAD SEARCH←
FILE:SEARCH - TYPE:ALGOL  -- LOADING

38 RECORDS LOADED.


 END LOAD 1.0 SEC.

CHANGE TYPE TO GTL←
#
WHATS←
FILE SEARCH (WORKFILE), TYPE GTL, 38 RECORDS
#
WHATS OBJECT←
ERR: OBJECT
?←
FILE NOT IN YOUR LIBRARY.

TYPE ALGOL←
#
WHATS←
FILE SEARCH (WORKFILE), TYPE ALGOL, 38 RECORDS
#
CHANGE TYPE GTL←
#
REMOVE←
#
CHANGE SEARCH TYPE GTL←
#
WHATS SEARCH←
FILE SEARCH, TYPE GTL, 38 RECORDS, CREATED 03/02/72 (1210) SF=7
#
```

4.3.8. COMPILE|C

To compile a source program in order to create an object file, the user may enter a COMPILE command. The format is as follows:

(C|COMPILE)[file-name][compiler-file-type|:first-letter-of-compiler-file-type]program-parameter-info

If a file-name is not given, the source version of the work-file is compiled; otherwise, the source version of the specified file is compiled and the resultant object file is saved. If the work-file is compiled, the object file is kept with the work-file so that both source and object versions can be saved with a SAVE command. Program parameter information applies to the object program being compiled, not to the compiler.

It should be noted that a SAVE command is ignored if data has not been entered into the work-file since being created or saved. Thus, it is not necessary to enter the sequence of commands

```
LOAD X;COMPILE;SAVE
```

in order to create and save the object version of X. The command which should be used is

```
COMPILE X
```

which causes the object version of the file to be saved.

The compiler-file-type may be ALGOL, BASIC, COBOL, CODASYL, DYNAMO, ESPOL, FORTRAN, GTL, TSPOL, XALGOL, or an abbreviation consisting of a colon followed by the first letter of the compiler-file-type, when non-ambiguous. Specifying a compiler is required when the file is associated with a non-compiler-file-type. In order to override a compiler-file-type which is associated with the source version of a work-file, a compiler-file-type may be specified in the command.

In case the user decides to prematurely discontinue a compilation, and the compiler is not typing at the time, he may depress WRU(CTRL E). If the compiler is typing at the time, the user may depress BREAK.

COBOL, CODASYL, and FORTRAN source language input is assumed by default to be entered and stored in remote free-field format.

For COBOL and CODASYL, the following describes the remote free-field format:

- column 1 is margin A,
- column 2 is margin B,
- continuation is denoted by a hyphen in column 1,
- compiler control card images are denoted by a dollar sign in column 1,
- NOTE card images are denoted by an asterisk in column 1,
- only 66 columns of input text per line are allowed,
- sequence-number is in columns 73-80.

For FORTRAN, the following describes the remote free-field format:

- continuation is denoted by a hyphen in column 1,
- comment card images are denoted by the characters "C-" in columns 1 and 2,
- labels may be a maximum of five columns long; a non-blank non-numeric character, or the seventh column after the start of the label, ends the label and starts the card text; a label should be separated from the sequence-number by one or more blanks,
- FILE card images must start in column 1; the word FILE must be followed by two blanks,
- compiler control card images are denoted by a dollar sign in column 1,
- for other card images, text begins with the first non-blank character,
- only 66 columns of input text per line are allowed.

ALGOL and GTL allow mnemonic syntactic substitutions to be made for the relational and assignment operators, as shown by the following table:

Operator	Mnemonic
←	:=
>	GTR
<	LSS
≥	GEQ
≤	LEQ
≠	NEQ
=	EQL

Since most of the characters on the left may not be entered from nor printed on a remote terminal as data, the mnemonics should be used in the correct syntactic positions. In addition, GTL allows QMARK as a substitution for the one-character string containing the question mark character.

COMPILE

The B5700 DYNAMO compiler accepts all of DYNAMO II except for the following differences:

- no MACRO definitions are allowed,
- SAMPLE function has only two arguments, as in DYNAMO I,
- STEP and RAMP functions can be initialized to nonzero values,
- the RUN card must precede the run to which it pertains,
- implicit multiplication is allowed,
- no equation types or numbers are required,
- a table definition must have an * following the table name,
- RUN, PRINT, PLOT, etc. must begin in column 1; equations may begin in column 2-72,
- continuation is denoted by the " symbol as the last character of the continued line and the next line starting in column 2 of the next image.

Plotting and printing may be produced on the remote terminal, automatically scaled to the width, by placing the following image first in the DYNAMO source program, starting in column 1:

```
REMOTE LIST PRINT
```

Otherwise printing and plotting will be produced on the high-speed onsite printer.

If a work-file is not open when required, the error message printed is the following:

```
ERR:NO FILE
```

If a source version of the file-name is not present, the error message printed is as follows:

```
ERR:file-name
```

If the work-file has a non-compiler-file-type, the error message printed is the following:

```
ERR:TYPE
```

If the input file is found and compilation has begun, CANDE types the following message:

```
COMPILING
```

and, when the compilation is finished, the following message:

```
END COMPILE n.m SEC.
```

is typed indicating the amount of processor time used during the compilation, if the compilation was successful. If the compilation was not successful, the following message is typed:

```
ERR COMPILE n.m SEC.
```

```
MAKE TEST GTL←
FILE:TEST - TYPE:GTL  -- CREATED
SEQ←
100BEGIN←
200FILE REMOTE;←
300INTEGER I;←
400PRINT#ENTER I#;←
500I:=READN(TWX);←
600I:=I*2;←
700PRINT #I SQUARED =#,I;←
800J:=READN(TWX);←
900PRINT J;←
1000END.←
1100←
#
COMPILE←
 WAIT.

 COMPILING.
00000800:COL 02:J #100
UNDECLARED IDENTIFIER.
00000900:COL 08:J #100

 ERR COMPILE 3.9 SEC.

*300/;/,J;←
COMPILE←
 WAIT.

 COMPILING.

 END COMPILE 6.2 SEC.

RUN←
 RUNNING

ENTER I
?144←
I SQUARED = 20736
?1234←
1234

 END TEST .9 SEC.

REMOVE←
#
```

4.3.9. COPY

The COPY command is used to copy a file or a set of files into the work-file, another file, or onto a peripheral unit. The format of the COPY command is the following:

```
COPY([file-name[/user-code]]TO(PRINTER|PUNCH|TAPE|file-name)|
     file-name[/user-code]sequence-list[RESEQ resequence-info])
```

If the COPY operation is successful, the number of records copied and the sequence-number of the last record copied are returned to the terminal.

When the COPY...TO... option is used, the entire designated file is copied to a peripheral unit or to another, newly-created file. When the COPY TO TAPE form is used, all the user's files are copied to a dump tape labeled with his unique user-code for more permanent retention. The user should allow a sufficient time for his files to have been copied to tape, then request the reel number from the operator, if the operator has not already sent the reel number to the terminal. When the COPY...TO PUNCH form is used, the proper information is sent to the operator for identifying the deck to be punched. If the cards are to be charged to a different account number than the reference number, this information must be sent to the operator.

Within the rules of file security (see Appendix C), the COPY command may be used to access files under another user's user-code by including a/and the proper user-code after the file-name. If the user issuing the COPY command has not been authorized by the other user to access that user's files, the COPY is not performed and CANDE types the following message:

```
ERR:user-code
```

If the file is not in the library, CANDE types the following message:

```
ERR:file-name
```

If there is no work-file, and one is required, the error message is as follows:

```
ERR:WRKFILE
```

Otherwise, except for COPY TO TAPE, after the copy has been performed, CANDE types the following:

```
mm RECORDS COPIED (LAST RECORD COPIED = dddd)
END COPY n.m SEC.
```

File Adjustment

The CANDE COPY command provides for the proper file adjustment depending upon the file-types of the work-file and input file. The different types of file adjustment, described in the following paragraphs, are as follows:

Work-file is a sequential file, and the input file is a type DATA file.

Work-file is a sequential file, and the input file is a sequential file.

Work-file is a type DATA file.

Work-File : Sequential File - Input File : Type DATA File

If the RESEQ option is not specified, the COPY command attempts to convert the characters in the input file in record positions 73 through 80 into a sequence-number. If this would otherwise result in an improperly sequenced file, error messages are given and the sequence-numbers are adjusted, if possible; otherwise, the operation is terminated. If only a portion of the input file is to be copied, the records to be copied are referenced by their position in the file, not by sequence-number.

Work-File : Sequential File - Input File : Sequential File

If the RESEQ option is not specified, the COPY command expects to find sequence-numbers in the input file in record positions 73 through 80. If the input file is not properly sequenced, error messages are given, and the sequence-numbers are adjusted, if possible; otherwise the operation is terminated. If only a portion of the input file is to be copied, the records to be copied are referenced by their sequence-numbers.

Work-File : Type DATA File

When the work-file has file-type DATA, the COPY command does not examine the input file sequence field, but transfers the records as 80-character units to the work-file. If the input file is, in fact, a sequential file, an 8-digit sequence-number is normally present in character positions 73 through 80, though this information is ignored by the COPY command.

```
MAKE EXAMPL SEQ←
FILE:EXAMPL - TYPE:SEQ  -- CREATED
SEQ←
100REC 1←
200REC 2←
300REC 3←
400REC 4←
500REC 5←
600←
#
SAVE←
 WAIT.


FILE:EXAMPL - TYPE:SEQ  -- SAVED.

MAKE FILER←
FILE:FILER - TYPE:SEQ  -- CREATED
COPY EXAMPL←
 WAIT.
5 RECORDS COPIED (LAST RECORD COPIED=500)


 END COPY 1.2 SEC.

P←
100 REC 1
200 REC 2
300 REC 3
400 REC 4
500 REC 5


#
COPY EXAMPL 100, 400-END RESEQ 10+2←
 WAIT.
3 RECORDS COPIED (LAST RECORD COPIED=14)


 END COPY 1.1 SEC.

P←
10 REC 1
12 REC 4
14 REC 5


#
"NOTE THAT COPY DESTROYED PREVIOUS CONTENTS OF WORK-FILE←
COPY EXAMPL RESEQ 1000←
 WAIT.
5 RECORDS COPIED (LAST RECORD COPIED=1400)


 END COPY 1.2 SEC.
```

```
P←
1000 REC 1
1100 REC 2
1200 REC 3
1300 REC 4
1400 REC 5


#
COPY TO PRINTER←
 WAIT.
5 RECORDS COPIED (LAST RECORD COPIED=1400)


 END COPY .8 SEC.

COPY EXAMP TO PRINTER←
 WAIT.
ERR: EXAMP
?←
FILE NOT IN YOUR LIBRARY.

COPY EXAMPL TO PRINTER←
 WAIT.
5 RECORDS COPIED (LAST RECORD COPIED=500)


 END COPY .9 SEC.

COPY EXAMPL TO NEWFIL←
 WAIT.
5 RECORDS COPIED (LAST RECORD COPIED=500)


 END COPY .9 SEC.

COPY EXAMPL TO PUNCH←
 WAIT.
PUNCHING...
5 RECORDS COPIED (LAST RECORD COPIED=500)


 END COPY .9 SEC.

REMOVE;REMOVE EXAMPL←
#
```

4.3.10. CREATE

The CREATE command creates a new file and establishes it as the work file. Format for the CREATE command is the following:

```
CREATE file-name[file-type|:first-letter-of-file-type]
```

The file-type may be ALGOL, BASIC, COBOL, CODASYL, DATA, DYNAMO, ESPOL, FORTRAN, GTL, INFO, SEQ, TSPOL, or XALGOL which can be abbreviated as colon, followed by the first letter of the file-type, when nonambiguous. If no file-type is specified, then sequenced (SEQ) is assumed.

If the file has been successfully created with the CREATE command, CANDE responds with the following message:

```
FILE:file-name - TYPE:file-type -- CREATED
```

If a file with the specified file-name already exists, CANDE sends the following message:

```
FILE:file-name - TYPE:file-type -- DUPLICATE NAME
```

If no user disk is available, an error message will be given and the CREATE command will be ignored.

```
CREATE BFILE BASIX'C←
FILE:BFILE - TYPE:BASIC  -- CREATED
REMOVE←
#
CREATE←
ERR: NO NAME
?←
THAT COMMAND REQUIRES A FILE NAME.

CREATE AFILE←
FILE:AFILE - TYPE:SEQ  -- CREATED
REMOVE←
#
CREATE FORTRN :R'F←
FILE:FORTRN - TYPE:FORTRAN  -- CREATED
REMOVE←
#
CRW'EATE DATAFI DATA←
FILE:DATAFI - TYPE:DATA  -- CREATED
?E'DATA←
 OK
THIS RECORD WAS ENTERED IN DATA MODE.←
?END←
#
P←
THIS RECORD WAS ENTERED IN DATA MODE.


#
REMOVE←
#

CREATE FILE33←
FILE:FILE33 - TYPE:SEQ  -- CREATED
SAVE←
FILE:FILE33 - TYPE:SEQ  -- SAVED.

CREATE FILE33←
FILE:FILE33 - TYPE:SEQ  -- DUPLICATE NAME
WHATS←
FILE FILE33 (WORKFILE), TYPE SEQ, 0 RECORDS
#
REMOVE←
#
REMOVE FILE33←
#
```

4.3.11. DELETE

The DELETE command is used to delete all or part of the contents of a work-file or of another file. The format is the following:

DELETE[file-name][ALL|sequence-list][RESEQ resequence-info]

If there are no parameters following the DELETE command, the ALL option is assumed. The parameter ALL causes the contents of the work-file to be removed, but does not affect the file-name associated with the work-file, nor the SEQ base or increment.

If the RESEQ option is specified, the base and increment for resequencing are assumed by default to be 100, and MOVING records (RESEQ 100,300) is not permitted. If the file has file-type DATA, the records to be DELETEd are referenced by their position in the file, not by sequence-numbers. If the file has file-type SEQ, the records to be DELETEd are referenced by their sequence-numbers.

The file-type of a file may be changed from DATA to SEQ and vice versa. A work-file's file-type may be changed from SEQ to DATA only. The COPY command, with RESEQ option used if necessary, may also be used to effect this conversion.

The DELETE command should never be used to delete just a few lines; deleting them individually by typing their sequence-numbers, followed by group marks is much more efficient in terms of computer time.

If sequence-number parameters are used, an entry of the form s causes the line with that sequence-number to be deleted. An entry of the form s_1-s_2 causes the deletion of all lines from the first through the second sequence-number, inclusively. The use of the word END is equivalent to using the highest sequence-number in the file. Thus, if the last entry is END, the last line in the file is deleted and, if the last entry has the form s TO END, all lines with a sequence-number greater than or equal to s are deleted. The sequence-numbers must be arranged in ascending numerical order. A maximum of nine sequence-numbers are allowed in a given list. A request to delete non-existent records according to sequence-number reference is ignored.

```
MAKE DELFIL←
FILE:DELFIL - TYPE:SEQ   -- CREATED
100REC 1←
200REC 2←
300REC 3←
400REC 4←
←
DELETE 200←
 WAIT.


 WAIT.
1 RECORDS DELETED.


 END DELETE 1.1 SEC.

P←
100 REC 1
300 REC 3
400 REC 4


#
500RE C''C 5←
600REC 6←
DELETE 500-END RESEQ 1000←
 WAIT.


 WAIT.
2 RECORDS DELETED.


 END DELETE 1.1 SEC.

P←
1000 REC 1
1100 REC 3
1200 REC 4


#
DELETE ALL←
#
P←


#
```

```
REMOVE←
#



MAKE FILE1 DATA←
FILE:FILE1 - TYPE:DATA  -- CREATED
?DATA←
 OK
REC 1←
REC 2←
REC 3←
REC 4←
REC 5←
REC 6←
?ENDO'←
#
SAVE←
 WAIT.


FILE:FILE1 - TYPE:DATA  -- SAVED.

DELETE 2-4←
 WAIT.
3 RECORDS DELETED.


 END DELETE .8 SEC.

P←
REC 1
REC 5
REC 6


#
```

```
P FILE1←
REC 1
REC 2
REC 3
REC 4
REC 5
REC 6


#
DELETE FILE1 3-5←
ERR: WRKFILE
?←
I CANNOT ALTER THAT FILE - IT IS YOUR WORK-FILE.
REMOVE←
#
DELETE FILE1 3-5←
 WAIT.
3 RECORDS DELETED.


 END DELETE .7 SEC.

P FILE1←
REC 1
REC 2
REC 6


#
REMOVE FILE1←
#
```

4.3.12. DISPLAY|D

The DISPLAY command is a variation of the PRINT or LIST command and is used to print records on the terminal. Its format is as follows:

```
DISPLAY[($|CHANGES)|file-name[/user-code]]sequence-list
    [*|SQUASHED][#|NUMBERED]
```

The sequence-numbers printed will be to the entire eight digits. This variation is useful in certain applications to eliminate a conflict with the first character in a record being numeric or to line up printed output.

For more details, see the LIST command.

DISPLAY

```
MAKE FILE9←
FILE:FILE9 - TYPE:SEQ  -- CREATED
1REC 1←
10REC 2←
100REC 3←
1000REC 4←
10000REC 5←
100000REC 6←
1000000REC 7←
10000000REC 8←
P←
1 REC 1
10 REC 2
100 REC 3
1000 REC 4
10000 REC 5
100000 REC 6
1000000 REC 7
10000000 REC 8


#
DISPLAY←
00000001REC 1
00000010REC 2
00000100REC 3
00001000REC 4
00010000REC 5
00100000REC 6
01000000REC 7
10000000REC 8


#
LIST←


FILE:FILE9 - TYPE:SEQ  --03/07/72  7:21 PM.

1 REC 1
10 REC 2
100 REC 3
1000 REC 4
10000 REC 5
100000 REC 6
1000000 REC 7
10000000 REC 8


 END QUIKLST 1.0 SEC.

REMOVE←
#
```

4.3.13. DO|EXECUTE|E

The work-file or an object file on disk can be executed by using the EXECUTE (or the DO) command. Its formats are as follows:

(DO|EXECUTE|E)[file-name[/user-code]]program-parameter-info

The program is run from the object file associated with the file specified in the command. If authorized to do so, a user may execute the object versions of files in another user's library by following the file-name with a slash and then the other user-code.

The EXECUTE (or DO) command may be preceded with EQUATE commands and followed by program-parameter-info. In case no program-parameter-info occurs, the following parameters are assumed:

PROCESS = 2
IO = 2
STACK = 512
COMMON = 0

In case the user decides to prematurely discontinue a program and the program is not typing at the time, he may depress WRU(CTRL E). If the program is typing at the time, it may be necessary to depress BREAK to stop the typing; if BREAK itself does not discontinue the program, he may then depress WRU.

If there is no object version of the designated file, the error message is as follows:

ERR:NOFILE

If the object file is found, CANDE types the following message:

RUNNING

and then any terminal output the program produces.

If the program finishes successfully the message typed is as follows:

END file-name n.m SEC

If it finishes with an error finish, the message typed is as follows:

ERR file-name n.m SEC.

In either case, the processor time used to execute the job is given as n.m SEC.

```
DO SEARCH←
 RUNNING


?RESET CONCISE OPTION FOR THIS PROGRAM←
?SEARCH←
RCC63YH LOCK
FILE TYPE GTL      DATA
LOGICAL RECORD LENGTH 10
PHYSICAL RECORD LENGTH 300
BLOCKING FACTOR 30
CREATION DATE 72067
CREATION TIME 1936
DATE OF LAST ACCESS 72068
SAVE FACTOR 7
NUMBER OF LOGICAL RECORDS 79
SEGMENTS/ROW 30
NUMBER OF ROWS IN USE 1
SIZE OF FILE(SEGMENTS) 30
?OSEARCH←

RCC63YH UNLOCK
FILE TYPE GTL      CODE
LOGICAL RECORD LENGTH 30
PHYSICAL RECORD LENGTH 30
BLOCKING FACTOR 1
CREATION DATE 72067
CREATION TIME 1936
DATE OF LAST ACCESS 72068
SAVE FACTOR 8
NUMBER OF LOGICAL RECORDS 86
SEGMENTS/ROW 87
NUMBER OF ROWS IN USE 1
SIZE OF FILE(SEGMENTS) 87
?END←

 END SEARCH 1.8 SEC.
```

4.3.14. EQUATE

The user may change certain of the attributes of the files in his programs (or other users' programs) without recompiling them through the use of the EQUATE command. The format of it is as follows:

```
EQUATE internal-name = [prefix/]suffix[unit...]
```

The term internal-name is the name of the file as referenced in the program, prefix is the first name of the desired file-name, suffix is the second name, and unit is the peripheral unit desired to be accessed, and may be selected from the following table:

unit	meaning
BACKUP DISK	printer backup disk
BACKUP TAPE	printer backup tape
CARD	card reader
DISK	serial disk file access method
DISK RANDOM	random disk file access method
DISK SERIAL	serial disk file access method
DISK UPDATE	update disk file access method
FORM	special forms on output
PAPER	remote terminal
PAPER TAPE	remote terminal
PRINT	printer backup tape
PUNCH	punch backup tape
RANDOM	random disk file access method
REMOTE	remote terminal
SERIAL	serial disk file access method
SPO	supervisory printer
TAPE	magnetic tape
UPDATE	update disk file access method
(empty)	serial disk file access method

If the unit references any type of disk file, and the term [prefix/] is not specified, the system will automatically assume a prefix of the user-code of the user entering the EQUATE command. If the unit references any type of disk file, and the term [prefix/] is specified, the system will check for the existence of a disk file with name prefix/suffix which the user is capable of accessing; if such a file does not exist, the EQUATE command will be rejected with an appropriate error message.

An EQUATE command may be legally followed only by an EQUATE, RUN, EXECUTE, DO, or CALL command. If any other type of command is entered, all previous EQUATE commands in the current chain will be forgotten and an appropriate error message will be given. Each EQUATE command must appear on a separate line of input. Any number of EQUATE commands may be entered prior to the execution of a program, and all will apply only to that one program being executed.

The LIST PROGRAM FILES command may be used to determine the proper internal-names of program files for use in the EQUATE command. No error message will be given if an EQUATE command references a non-existent internal file-name. Not all legal internal file-names may appear in EQUATE commands; the only legal ones are those which begin with a letter and are composed of letters and digits. Thus COBOL and CODASYL programs may require correction on certain internal file-names which may begin with a digit or contain one or more hyphens, and yet must be used in EQUATE commands. The REPLACE command may be used to great advantage in this case.

```
MAKE EQT GTL←
FILE:EQT - TYPE:GTL   -- CREATED
100BEGIN←
200FILE IN EQTFIL DISK SERIAL (2,10,300);←
300READ(EQTFIL);←
400END.←
RUN←
 %AIT.

 COMPILING.

 END COMPILE 2.9 SEC.

 RUNNING


-NO FILE ON DISK EQTFIL RCC63YH,NEAR LINE 00000400

 ERR EQT .4 SEC.

EQUATE EQTFIL=R REMOTE←
RUN←
 RUNNING

?NOTICE THAT THIS TIME EQTFIL REFERENCED TERMINAL, NOT DISK←

 END EQT .3 SEC.

EQUATE EQTFIL=ANY/THING TAPE←
SEQ←
FRR:  SEQ
?←
A RUN OR EXECUTE OR CALL OR EQUATE MUST FOLLOW AN EQUATE COMMAND

REMOVE←
#
```

4.3.15. FILE|FILES

The FILES command may be used to obtain the file-names and versions of the files in the user's library. Its format is the following:

(FILE|FILES)

The file-names of all the files with names of the form file-name/user-code are listed. The end of the list is indicated by a number symbol. An asterisk is used to indicate object versions of files.

```
FILE←
  *SEARCH *G22      *SERCH  SEQ2     SERCH   G22      SEARCH  *GET
  TAP     SCHL      *TAP    GET
#
```

4.3.16. FIND

The FIND command allows the user to search a file or a subset of a file for the records containing a given string. The FIND command has the following format:

```
(FIND  [FILE file-name[/user-code]]
       [FIRST][LITERAL]
       (delimiter sought-string delimiter|mnemonic)
       sequence-list
       [PRINT(SEQUENCE|TEXT|SITE|FILE file-name),) ...
```

If the FILE option is specified, the designated file is searched, if it is present and the user has access to it. If the FILE option is not specified, the work-file is searched, if it is present.

When the first record with the required characteristics being sought is found, the FIRST option, when used, terminates the search of the file; otherwise the entire designated portion of the file is searched.

The LITERAL option is used to specify that the element which is contained between the delimiters is to be sought as an entity.

The mnemonic option allows the user to search a file for records containing special characters most of which may not be entered from the remote terminal. These are ARROW, GEQ, LEQ, GTR, LSS, NEQ, EQL, with interpretations as $\leftarrow$, $\geq$, $\leq$, $>$, $<$, $\neq$, $=$, respectively.

The sequence-list option may be used to limit the area in which records with the required characteristic are to be sought. By default the entire file is searched.

The PRINT option is used to specify the output form. The PRINT SEQUENCE, which is the default for this option, causes only the sequence-number of the records containing desired strings to be printed on the terminal. When more than one sought-string exists in a record and the PRINT SEQUENCE option is used, an asterisk will precede the sequence-numbers. The PRINT TEXT option causes the entire record to be listed on the terminal. The PRINT SITE causes the records to be written on a printer at the site. The PRINT FILE file-name option causes the records to be written to a file named file-name.

The REPLACE command is similar to the FIND command except that a search-and-replace operation is allowed in the earlier command. Both commands may be iterated by following each iteration except the last with a space, a comma, and a left arrow. Only one PRINT option is allowed in the string of commands.

```
MAKE FINDER←
FILE:FINDER - TYPE:SEQ   -- CREATED
180END←
190END OF JOB←
200ENDOFJOB←
FIND LITERAL/END/←
 WAIT.


 WAIT.
 180 190 200
NUMBER OF STRINGS FOUND =          3


 END FIND .7 SEC.

#
FIND FIRST /END/←
 WAIT.
 180
NUMBER OF STRINGS FOUND =          1


 END FIND .7 SEC.

#
FIND/END/←
 WAIT.
 180 190
NUMBER OF STRINGS FOUND =          2


 END FIND .7 SEC.

#
REPLACE/JOB/WITH ARROW←
 WAIT.

NUMBER OF STRINGS REPLACED =           1


 END REPLACE .9 SEC.

#
```

```
FIND ARROW PRINT TEXT←
 WAIT.
190 END OF ?

NUMBER OF STRINGS FOUND =          1


 END FIND .7 SEC.

#
FIND LITERAL /END/ 190-END←
 WAIT.
 190 200
NUMBER OF STRINGS FOUND =          2


 END FIND .8 SEC.

#

REMOVE←
#
```

4.3.17. FIX

The FIX command is used to delete or replace a portion of a line of input. It has the following format:

(*|FIX) sequence-number delimiter sought-string delimiter
[replacement-string]

A FIX command causes CANDE to replace the characters that are specified by the old string with the characters in the new string. CANDE performs this action by searching the line with the given sequence-number from left to right until it finds the first string of characters in the line which is identical to the specified sought-string. It then discards those characters and, if a replacement-string is included in the command, it inserts the characters of the replacement-string in their place. Therefore, any given string of characters in a line may be deleted or replaced by another string.

The delimiter is used to mark the beginning and end of the sought-string. It may be any valid non-blank character that does not appear in the old string. The first non-blank character after the sequence-number is taken as the delimiter. All characters, including blanks, between the first two appearances of the delimiter, are taken into the sought-string.

All characters up to but not including the left arrow following its second appearance are taken into the replacement-string. Neither of the strings may exceed 63 characters in length. If the FIX command results in a record of more than 72 characters (80 for type DATA files), the record is truncated to 72 (80) characters. If it results in a record of less than 72 (80) characters, the record is space-filled from the right.

When type DATA records are FIXed and a sequence-number appears in character positions 73 through 80, the user must include sufficient spaces in representing the existing string so that the replacement string is not longer or shorter than the existing string. If the replacement string is longer or shorter, the sequence characters are shifted to the right, or left, rendering them useless as a sequence-number. When the sequence field is empty, or unimportant, the user need not be concerned with string sizes.

CANDE does not apply the change when it is entered. Instead, it stores the contents as a record with other data entries in the work-file. Then, when a command which affects the work-file is issued, such as LIST, RUN, SAVE, etc., all of the changes are made, and any errors in the FIX commands are

noted. Thus, error messages for non-match and truncation are typed following the first command which uses the work-file. Except for causing error messages to be typed, FIX commands in error are ignored and processing continues. FIX commands containing syntactical errors are ignored and a suitable error message is typed when an offending FIX command is entered.

Because of this, it is good practice to use the UPDATE command after the desired set of corrections has been made to the work-file. The FIX errors, if any, will be printed at this point. If, instead of UPDATE, the user had entered COMPILE, and errors were found, the compilation would already have been started before the user knew of any FIX errors he had made.

Since CANDE initially treats a FIX command as if it is a record in the work-file, more data or another command may be entered after the FIX command. However, the FIX command may not be combined with other commands through the use of the semicolon.

If the specified string cannot be found, the following message is typed:

CANNOT LOCATE YOUR FIX STRING FOR RECORD sequence-number

If the specified string is found, but would cause nonblank information to be truncated, the following message is typed:

NOT ENOUGH ROOM FOR YOUR FIX IN RECORD sequence-number

```
MAKE FIXER←
FILE:FIXER - TYPE:SEQ  -- CREATED
100 ABCDEFGHIJKLMNOPQRSTUVWXYZ←
*/AR/←
ERR: /
*100/A/$←
*100/AR/←
*100/Z/••••••••••••••••••••••••••••••••••••••••••••••••••←

U←
CANNOT LOCATE YOUR FIX STRING FOR RECORD 100
NOT ENOUGH ROOM FOR YOUR FIX IN RECORD 100


#
P←
100  $BCDEFGHIJKLMNOPQRSTUVWXYZ


#
DELETE ALL←
#
100THIS IS A SAMPLE←
200TO SHOW HOW FIX←
300WORKS←
FIX 100• S•N EX←
FIX 200#HOW#THE WAY IN WHICH←
P←
100 THIS IS AN EXAMPLE
200 TO STHE WAY IN WHICH HOW FIX
300 WORKS


#
*200 $THE WAY IN WHICH$←
*300 •S•ING←
P←
100 THIS IS AN EXAMPLE
200 TO S HOW FIX
300 WORKING


#
*200+S +S←
P←
100 THIS IS AN EXAMPLE
200 TO SHOW FIX
300 WORKING


#
REMOVE←
#
```

4.3.18. GUARD

The GUARD command permits the user to build or modify a GUARD file in order to allow other users or user's programs to read or to read and write a file. Its format is as follows:

GUARD

The program starts by typing the following message:

NEW OR OLD GUARD FILE?

and the user responds with the word NEW if the user wishes to create a new GUARD file, or OLD if the user wishes to update an existing GUARD file. The program then types the following message:

LOCK FILE NAME?

and the user types the name of the old file or the name of the new file. If a current GUARD file is being updated, the program types the following message:

ADD, DELETE, LIST, SAVE, OR QUIT?

The user then responds with an applicable choice causing the program to again type the following message:

ADD, DELETE, LIST, SAVE, OR QUIT?

With the exception of QUIT, the user can type any of these words in any order until all operations on the file are accomplished. Actions are taken for each option as follows:

- ADD is used to add user-codes and/or program names to the file. After ADD is typed, CANDE types the following message:
  READ ONLY NAMES?
- The user may then enter a list of user-codes and/or program names which are added to the file. These user-codes/program names are allowed to read but not to change those files with which this GUARD file is associated. Program names must be entered in the following format:

  file-name/user-code (of the owner of the file)

The items in the list must be separated by commas or blanks. If the user does not wish to add any read-only names to the GUARD file, he should enter a left arrow.

If the user does not wish to add any read-only names to the GUARD file, he should enter a left arrow.

Next the program sends the following message:

READ/WRITE NAMES?

In this case, any user-codes and program names that are entered are able to access and change those files with which this GUARD file is associated. If the user does not wish to add read/write names to the GUARD file, he should enter a left arrow.

- DELETE causes the program to type the following message:

  NAMES TO BE DELETED?

  The user then enters those user-codes and program names ahat are desired for removal from the GUARD files.

- LIST produces a list of all the user-codes and programs in the file. Read-only user-codes and programs are preceded by (R) and read/write user-codes and programs are preceded by (W).

- SAVE must be entered to save the GUARD file. The file can be saved more than once, in which case only the version last saved remains on disk.

- QUIT causes the program to terminate. Any additions or deletions made since the last SAVE are not entered into the GUARD file. The use of the WRU key to discontinue GUARD may result in an undefined GUARD file, and should not normally be used.

When a new file is being created, the program first asks for read-only names and then for read/write names, just as it does for an ADD command.

Then the program types the following message:

ADD, DELETE, LIST, SAVE, OR QUIT?

and the user may use any of the options which are described above.

The LOCK...WITH command must be used to actually attach the GUARD file to the file to be guarded.

```
GUARD←

NEW OR OLD LOCK FILE??NEQ←
WHAT??NEW←
LOCK FILE NAME??BEST←
READ ONLY NAMES??ONE, EYED, JACKS←
READ/WRITE NAMES??LEFT, HANDED, KINGS←
ADD, DELETE, LIST, SAVE, OR QUIT?
?LIST←
(R) ONE
(R) EYED
(R) JACKS
(W) LEFT
(W) HANDED
(W) KINGS
ADD, DELETE, LIST, SAVE, OR QUIT?
?D'ADD←
READ ONLY NAMES??WILD DEUCES←
READ/WRITE NAMES??←
ADD, DELETE, LIST, SAVE, OR QUIT?
?LIST←
(R) ONE
(R) EYED
(R) JACKS
(W) LEFT
(W) HANDED
(W) KINGS
(R) WILD
(R) DEUCES
ADD, DELETE, LIST, SAVE, OR QUIT?
?SAVE←
LOCK FILE SAVED.
ADD, DELETE, LIST, SAVE, OR QUIT?
?QUIT←
THANK YOU.


 END GUARD 1.8 SEC.

LOCK G22 WITH BEST←
#
LIST FILES G22←
03/07/72 RCC63YH  7:43 PM.
NAME    TYPE     RECS SEGS   CREATED   ACCESSED  W/R  W/B  S-F LOCKD BY

G22     GTL        18   10  03/02/72  03/07/72   10  300   10  BEST
Ø22     *GTL       21   21  03/02/72  03/07/72   30   30   20  BEST
#
```

4.3.19. HELLO

The HELLO command is used to initiate a log-in sequence. The format is:

HELLO [user-code][password]

The HELLO command causes the session of the current user to be terminated and a new session to be initiated without physically performing a terminal disconnect. The password should be entered in this command only if the printed output may be disposed of securely.

```
HELLO←
  ON FOR   12.1 SEC.
  CPU USE  .3 SEC.
  EXECUTE  .0 SEC.
  IO TIME  .0 SEC.
  OFF AT    2:08 PM.
  GOODBYE RCC63YH
03/13/72
```

```
ENTER USER CODE, PLEASE←RCC63YH←
AND YOUR PASSWORD
[redacted]
03/13/72   2:09 PM.
GOOD AFTERNOON, PASS E M: YOU HAVE LINE 20

#
HELLO RCC63YH←
[redacted]
03/13/72   2:09 PM.
GOOD AFTERNOON, PASS E M: YOU HAVE LINE 20

#
```

4.3.20. LIST|L

The LIST command is used to list the contents of a file. Its format is as follows:

```
(L|LIST)[($|CHANGES)|filename[/user-code]]sequence-list
    [*|SQUASHED][#|NUMBERED]
```

The PRINT command is similar to the LIST command, except that the heading identifying the file is omitted in the output resulting from the PRINT command. The DISPLAY command is similar to the PRINT command, except that sequence-numbers are printed to the full eight digits.

Without a file-name, the LIST command lists the work-file, if present. With a file-name, the specified file is listed. If a list of sequence-numbers is not included, the entire file is listed. Otherwise, the lines with the specified sequence-numbers are listed. An entry of the form s causes that line, if present, to be listed. An entry of the form s_1-s_2 produces a list of all lines (if any) with sequence-numbers in the range from the first sequence number through the second sequence-number. The word END is equivalent to the highest sequence-number in the file. A maximum of nine sequence-numbers are allowed in the list. The sequence-numbers in the list must be in ascending numerical sequence. Requests to list non-existent records according to sequence-number reference are ignored, and no error message is given. In order to LIST file without sequence-numbers, CHANGE its TYPE to INFO. If necessary, after it is listed, CHANGE its TYPE back to what it was originally.

CHANGES Option

The CHANGES option, or the dollar symbol, may be used to obtain a list of the alterations made to the work-file since the last update.

SQUASHED Option

The SQUASHED option, or the asterisk symbol, may be used to list record contents with extraneous blanks removed. This option is particularly useful for type DATA files in which a digit string appears in the sequence field or for fast listing of source language files.

NUMBERED Option

The NUMBERED option, or the number symbol, has meaning only with type DATA files, and it is ignored when listing sequential files. This option causes the printing of a record location on the line immediately preceding the data record.

```
LIST←
ERR: NOFILE
?←
NO WORK-FILE - USE MAKE OR LOAD.

MAKE LISTER DATA←
FILE:LISTER - TYPE:DATA  -- CREATED
?DATA←
 OK
REC 1←
REC 2←
REC 3←
REC 4←
←
←
?END←
#
LIST←


FILE:LISTER - TYPE:DATA  --03/07/72  7:46 PM.

REC 1
REC 2
REC 3
REC 4



 END LIST 1.3 SEC.

LIST NUMBERED←


FILE:LISTER - TYPE:DATA  --03/07/72  7:47 PM.

1
REC 1
2
REC 2
3
REC 3
4
REC 4
5

6



 END QUIKLST .8 SEC.

*1/1/     1←
```

```
LIST 1←

FILE:LISTER - TYPE:DATA  --03/07/72  7:47 PM.

REC      1


 END LIST 1.2 SEC.

LIST SQUASHED←

FILE:LISTER - TYPE:DATA  --03/07/72  7:48 PM.

REC 1
REC 2
REC 3
REC 4



 END QUIKLST .6 SEC.

SAVE←
FILE:LISTER - TYPE:DATA  -- SAVED.

CHANGE LISTER TYPE TO SEQ←
#
RESEQ LISTER←
ERR: WRKFILE
?←
I CANNOT ALTER THAT FILE - IT IS YOUR WORK FILE.

REMOVE←
#
RESEQ LISTER←
 WAIT.


 END RESEQ .9 SEC.

LIST←
ERR: NOFILE
LIST LISTER←


FILE:LISTER -03/07/72  7:49 PM.

100 REC      1
200 REC 2
300 REC 3
400 REC 4
500
600


 END QUIKLST .7 SEC.
```

LIST

```
LIST LISTER 100 400-END SQUASHED←


FILE:LISTER -03/07/72  7:50 PM.

100 REC 1
400 REC 4
500
600


 END QUIKLST .7 SEC.

CHANGE LISTER TYPE TO DATA←
#
LIST LISTER←


FILE:LISTER -03/07/72  7:51 PM.

REC      1                                                          \\
          \\  00000100
REC 2                                                               \\
          \\  00000200
REC 3                                                               \\
          \\  00000300
REC 4                                                               \\
          \\  00000400
                                                                    \\
          \\  00000500
                                                                    \\
          \\  00000600


 END QUIKLST .5 SEC.

LIST NUMBERED!DEL

LIST LISTER NUMBERED SQUASHED←


FILE:LISTER -03/07/72  7:53 PM.

1
REC 1 00000100
2
REC 2 00000200
3
REC 3 00000300
4
REC 4 00000400
5
 00000500
6
 00000600


 END QUIKLST .7 SEC.
```

4.3.21. LIST FILES

The LIST FILES command may be used to obtain much relevant information about disk files to which the user has access. The format of the command is as follows:

```
LIST FILES [TO (PRINTER|TELETYPE|file-name)]
    [[file-type]
     [SOURCE][OBJECT]
     [LOCKED][UNLOCKED][PUBLIC][SOLEUSER]
     [LITERAL string]
     [file-name][/user-code]] ...
```

The information which may be obtained through the use of this command is as follows:

NAME	name of each file (alphabetical order).
TYPE	associated file-type of each file. If the file version being described is object it is indicated by an *.
RECS	number of records contained in the file.
SEGS	number of disk segments used.
CREATED	date of creation.
ACCESSED	date last accessed. If accessed today, an asterisk flags the date.
W/R	words per record.
W/B	words per block.
S-F	save factor of file.
LOCKED BY	if the file is locked, with a security file, then this reflects the security file used in the GUARD command. If the file has been UNLOCKed or PUBLICed, such is indicated; otherwise this entry is blank.
FILES	total number of files listed.
SEGMENTS	total number of disk segments occupied by the files listed.
RECORDS	total number of records in the files listed.

The information normally sent to the user's terminal may be sent instead to the high-speed onsite printer or into a newly-created disk fiie through the use of the TO PRINTER or TO file-name option.

The default action of the LIST FILES command is to list all of the disk files belonging to the user entering the command. This action may be modified or restricted through the use of the other terms in the syntax. They may be entered in any order and, other than the [/user-code] term, form a restriction of the set of files listed. The [/user-code] term allows a user to interrogate another user's library; the only files listed in this case will be those to which the requesting user has access.

```
LIST FILES←
03/08/72 RCC63YH          08:39 PM
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

GET      GTL         40   20   02/24/72    03/03/72   10   300    7
G22      GTL         18   10   03/02/72 *  03/08/72   10   300   10  UNLOCKD
LSERC    GTL         38   20   03/02/72 *  03/08/72   10   300    7  UNLOCKD
SCHL     DATA       108   40   03/02/72    03/03/72   10   300   10
SEARCH   GTL         80   30   03/07/72 *  03/08/72   10   300    7
TAP      GTL        176   60   03/03/72    03/03/72   10   300    7
GET      *GTL       102  102   02/24/72    03/03/72   30    30    8
G22      *GTL        21   21   03/02/72 *  03/08/72   30    30   20  UNLOCKD
LSERC    *GTL        67   67   03/02/72 *  03/08/72   30    30    8  UNLOCKD
SEARCH   *GTL        87   87   03/07/72 *  03/08/72   30    30    8  UNLOCKD
TAP      *GTL        97   97   03/03/72    03/03/72   30    30    8
      11  FILES          554  SEGMENTS      834  RECORDS

 END LFILES .9 SEC.

LIST FILES UNLOCK←
ERR: UNLOCK
LIST FILES UNLOCKED←
03/08/72 RCC63YH          08:42 PM
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

G22      GTL         18   10   03/02/72 *  03/08/72   10   300   10  UNLOCKD
LSERC    GTL         38   20   03/02/72 *  03/08/72   10   300    7  UNLOCKD
G22      *GTL        21   21   03/02/72 *  03/08/72   30    30   20  UNLOCKD
LSERC    *GTL        67   67   03/02/72 *  03/08/72   30    30    8  UNLOCKD
SEARCH   *GTL        87   87   03/07/72 *  03/08/72   30    30    8  UNLOCKD
       5  FILES          205  SEGMENTS      231  RECORDS

 END LFILES .9 SEC.

LIST FILES SEARCH←
03/08/72 RCC63YH  8:43 PM.
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

SEARCH   GTL         80   30   03/07/72    03/08/72   10   300    7
SEARCH   *GTL        87   87   03/07/72    03/08/72   30    30    8  UNLOCKD
#
```

LIST FILES

```
LIST FILES /RISGWLW-
03/08/72 RISGWLW          08:43 PM
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

ACCESS   SEQ         18   10  02/23/72 * 03/08/72    10   300    7  UNLOCKD
ALLOC    SEQ         40   20  03/08/72 * 03/08/72    10   300    7  UNLOCKD
DELETE   SEQ         25   10  02/23/72 * 03/08/72    10   300    7  UNLOCKD
FIELD    SEQ         31   20  02/18/72 * 03/08/72    10   300    7  UNLOCKD
INSERT   SEQ         26   10  02/23/72 * 03/08/72    10   300    7  UNLOCKD
LALLOC   SEQ         42   20  03/08/72 * 03/08/72    10   300    7  UNLOCKD
LINKF    SEQ         52   20  02/23/72 * 03/08/72    10   300    7  UNLOCKD
LLIST    SEQ         69   30  03/08/72 * 03/08/72    10   300    7  UNLOCKD
NODE     SEQ         95   40  03/08/72 * 03/08/72    10   300    7  UNLOCKD
OPERNS   SEQ         16   10  03/08/72 * 03/08/72    10   300    7  UNLOCKD
QUEUE    SEQ         40   20  02/23/72 * 03/08/72    10   300    7  UNLOCKD
SALLOC   SEQ         49   20  03/08/72 * 03/08/72    10   300    7  UNLOCKD
SCRPT3   SEQ        143   50  02/18/72   02/23/72    10   300    7  UNLOCKD
SCRPT4   SEQ        120   40  02/23/72 * 03/08/72    10   300    7  UNLOCKD
STACK    SEQ         42   20  02/23/72 * 03/08/72    10   300    7  UNLOCKD
STKQUE   SEQ         14   10  03/08/72 * 03/08/72    10   300    7  UNLOCKD
XPLAIN   DATA        27   10  06/29/71   02/25/72    10   300    7  UNLOCKD
      17  FILES          360  SEGMENTS     849  RECORDS

 END LFILES 1.3 SEC.
```

4.3.22. LIST PROGRAM FILES

The user may request a listing of the attributes of the files of a given object program with the LIST PROGRAM FILES command. Its format is the following:

LIST PROGRAM FILES [file-name[/user-code]]

If no file-name is included, the object version of the work-file is assumed by this command. The information printed may be used to properly construct EQUATE commands to change the program file attributes when the program is run.

```
MAKE TESTF GTL←
FILE:TESTF - TYPE:GTL  -- CREATED
SEQ←
100BEGIN←
200FILE TERM REMOV'TE (1,17);←
300FILE DISKIN DISK "FILER" (1,10,300);←
400FILE DISKOUT DISK SERIAL [20:300] "DISSERL'" (2,10,300,SAVE 2);←
500FILE PRINTER PRINTER''(2,15);←
600FILE TAPE 2(2,305);←
700FILE CPUNCH 0(1,10);←
800FILE PTPUNCH 8(2,401);←
900END.←
1000←
#
C←
 WAIT.


 COMPILING.


 END COMPILE 3.2 SEC.

LIST PROGRAM FILES←

       TYPE          FIRST   LAST        INTERNAL NAME
                      NAME   NAME

     REMOTE        0000000/TERM          TERM
 DISK (RANDOM)     FILER  /RCC63YH       DISKIN
 DISK (SERIAL)     DISSER /RCC63YH       DISKOUT
     PRINT         0000000/PRINTER       PRINTER
 TAPE OR READER    0000000/TAPE          TAPE
   CARD PUNCH      0000000/CPUNCH        CPUNCH
PAPER TAPE UNLAB   0000000/PTPUNCH       PTPUNCH


 END FILES .7 SEC.
```

4.3.23. LOAD

The LOAD command loads an existing disk file into a newly-created work-file. The format is the following:

LOAD file-name

If the load is successful, CANDE types information concerning the name, type, and size of the file. However, if the file is not in the user's library, CANDE types the following message:

ERR:file-name

```
LOAD SEARCH←
FILE:SEARCH - TYPE:GTL  -- LOADING

38 RECORDS LOADED.


 END LOAD 1.2 SEC.

SAVE←
FILE:SEARCH - TYPE:GTL  -- SAVED.

LOAD←
ERR: NAME
?←
"VERB" CONTAINS TOO MANY PARAMETERS.

LOAD NOTHER←
ERR: NOTHER
?←
FILE NOT IN YOUR LIBRARY.

MAKE AFILE←
FILE:AFILE - TYPE:SE@  -- CREATED
100 A RECORD←
MAK
LOAD G22←
ERR: NO SAVE
?←
WORKFILE HAS UNSAVED RECORDS IN IT - PLEASE SAVE OR REMOVE IT.

REMOVE←
#
```

4.3.24. LOCK

The LOCK command restores a file to its original security status or attaches a GUARD file to another file. Its format is as follows:

LOCK[[SOURCE|OBJECT]file-name ...] ...[WITH guard-file-name]

All files which are SAVEd or COMPILEd are initially LOCKed under the user's user-code. The initial LOCKing is an automatic function giving the file the file security status of sole user. Thus, only the creator may access the file. The major function of the LOCK command is to return a file's security to its original security status (sole user under the user's user-code).

The purpose of the GUARD command is to create a GUARD file with additional user-codes. The LOCK...WITH command assigns the GUARD file to a source or object file in order to regulate the file's security. This device limits the number of users allowed to access the file and controls their class of access individually.

```
WHATS SEARCH←
FILE SEARCH, TYPE GTL, 38 RECORDS, CREATED 03/02/72 (1128) SF=7
#


LIST FILES SEARCH←
03/07/72 RCC63YH  8:10 PM.
NAME    TYPE      RECS SEGS   CREATED   ACCESSED  W/R  W/B  S-F LOCKD BY

SEARCH  GTL         38   20  03/02/72  03/07/72   10  300    7
SEARCH  *GTL        67   67  03/02/72  03/07/72   30   30    8  UNLOCKD
#
LOCK SEARCH←
#
LIST FILES SEARCH←
03/07/72 RCC63YH  8:11 PM.
NAME    TYPE      RECS SEGS   CREATED   ACCESSED  W/R  W/B  S-F LOCKD BY

SEARCH  GTL         38   20  03/02/72  03/07/72   10  300    7
SEARCH  *GTL        67   67  03/02/72  03/07/72   30   30    8
#
```

```
LOCK SOURCE SEARCH WITH LOCKER←
 ERR: LOCKER
#
?←
GUARD FILE MUST BE OF TYPE "LOCK".

REMOVE LOCKER←
#
GUARD←

NEW OR OLD LOCK FILE??NEW←
LOCK FILE NAME??BEST←
READ ONLY NAMES??←
READ/WRITE NAMES??←
ADD, DELETE, LIST, SAVE, OR QUIT?
?SAVE←
LOCK FILE SAVED.
ADD, DELETE, LIST, SAVE, OR QUIT?
?QUIT←
THANK YOU.


 END GUARD 1.5 SEC.




LOCK SOURCE SEARCH WITH BEST←
#
LIST FILES OBJECT !DEL


LIST FILES SEARCH←

LOCK SOURCE SEARCH WITH BEST←
#
LIST FILES SOURCE SEAT'RCH←
03/07/72 RCC63YH  8:14 PM.
NAME    TYPE     RECS SEGS   CREATED   ACCESSED  W/R  W/B  S-F LOCKD BY

SEARCH  GTL        38   20  03/02/72  03/07/72   10  300    7  BEST
SEARCH  *GTL       67   67  03/02/72  03/07/72   30   30    8
#

LOCK SEARCH←
#
LIST FILES SEARCH←
03/07/72 RCC63YH  8:16 PM.
NAME    TYPE     RECS SEGS   CREATED   ACCESSED  W/R  W/B  S-F LOCKD BY

SEARCH  GTL        38   20  03/02/72  03/07/72   10  300    7
SEARCH  *GTL       67   67  03/02/72  03/07/72   30   30    8
#
```

4.3.25. MAKE

The MAKE command creates a new disk file and establishes it as the work-file. The format of the MAKE command is as follows:

MAKE file-name[file-type|first-letter-of-file-type]

The file-type may be ALGOL, BASIC, COBOL, CODASYL, DATA, DYNAMO, ESPOL, FORTRAN, GTL, INFO, LOCK, SEQ, TSPOL, or XALGOL which can be abbreviated as colon, followed by the first letter of the type, for non-ambiguous cases. If no type is specified, then sequenced (SEQ) is assumed.

If the file has been successfully created with the MAKE command, CANDE responds with the following message:

FILE: file-name - TYPE: file-type -- CREATED

If a file with the specified file-name already exists, CANDE sends the following message:

FILE: file-name - TYPE: file-type -- DUPLICATE NAME

If no user disk is available, an error message will be given and the MAKE command will be ignored.

```
MAKE FILM BASIC←
FILE:FILM - TYPE:BASIC  -- CREATED
SAVE←
FILE:FILM - TYPE:BASIC  -- SAVED.

MAKE FILM←
FILE:FILM - TYPE:SEQ  -- DUPLICATE NAME

MAKE←
ERR: NO NAME
?←
THAT COMMAND REQUIRES A FILE-NAME.

REMOVE FILM←
ERR: WRKFILE
?←
I CANNOT REMOVE THAT FILE--IT IS YOUR WORK-FILE.

REMOVE←
#
REMOVE FILN'M←
#

MAKE FORTR:F←
FILE:FORTR - TYPE:FORTRAN  -- CREATED
REMOVE←
#
```

4.3.26. MERGE

An existing file can be merged into the work-file by the use of the MERGE command. It has the following format:

MERGE file-name[/user-code]sequence-list[RESEQ resequence-info]

The specified file, or the indicated portions of it, are MERGEd into the work-file according to sequence-numbers. In case of duplicate sequence-numbers, the record in the work-file is used in the case of the MERGE command and the record in the input file is used in the case of the RMERGE command.

The numbers in the sequence-list must be in ascending numerical order. END is equivalent to the last sequence-number in the list. A maximum of nine entries are allowed in the list. If a work-file has not been opened, the message typed is the following:

ERR:WRKFILE

For information concerning the RESEQ option, see the RESEQ command. This option can be used to move records about in a file while merging the file with another, and can be very powerful or very destructive, depending upon how well it is used. An especially useful variant of the MERGE command allows the user to copy a portion of the work-file into itself at a different location with resequencing if necessary. For example, in order to reproduce the records with sequence-numbers a-b in the work-file into the location starting with sequence-number s with sequence increment i, the following commands may be used:

SAVE;MERGE work-file-name a-b RESEQ s+i

For further examples of the use of the MERGE command, see the RMERGE command. Information there concerns the differences between the MERGE and RMERGE commands in terms of resolving sequence-number conflicts between the input file and work-file.

The base and increment for resequencing are assumed by default to be 100, and moving of records within the file (RESEQ 100,300) is not permitted.

At the completion of the merging operation, the number of records merged and the sequence number of the last record are printed.

```
MAKE F1←
FILE:F1 - TYPE:SEQ   -- CREATED
100 1←
200 2←
300 3←
SAVE←
 WAIT.


FILE:F1 - TYPE:SEQ   -- SAVED.

MAKE F2←
FILE:F2 - TYPE:SEQ   -- CREATED
150 1.5←
250 2.5←
350 3.5←
MERGE F1←
 WAIT.


 WAIT.
3 RECORDS MERGED (LAST RECORD MERGED=300)


 END MERGE 1.2 SEC.

P←
100   1
150   1.5
200   2
250   2.5
300   3
350   3.5


#

REMOVE←
#
REMOVE F1←
#
```

4.3.27 MONITOR

The use can specify a more permanent facility for recording work-file changes than provided through PRINT CHANGES through the use of the MONITOR command. Its format is the following:

```
MONITOR file-name
```

When the MONITOR file-name command is entered, CANDE searches the user's files for the presence of the specified file. If the file is already present, an error message will be given. If the file is not present, CANDE creates a file for the user, and places the file-name in the REMOTE/USERS record for subsequent reference. This file-name will remain in the record until it is replaced with another name by entering another MONITOR command.

The execution of the MONITOR command also sets a toggle in the LIST/CANDE program. When this toggle is ON, all changes (additions, FIXes, deletions) to the work-file will be recorded on the MONITOR file each time that the work-file is updated. If MONITORing is no longer required, the user may enter

```
RESET MONITOR
```

RESETting the MONITOR toggle does not remove the current MONITOR file-name from the REMOTE/USERS record, nor does it alter the MONITOR file itself. MONITORing the work-file changes may be resumed by entering

```
SET MONITOR
```

When CANDE receives this command, the REMOTE/USERS record is accessed to determine the name of the current MONITOR file. If no file-name is listed there, the following message will be given:

```
ERR:MON.FIL
```

If a file-name is, in fact, listed there, CANDE then proceeds to determine whether the specified file is actually on disk. If the file is not found in the user's library, an error message is given. If the file is found, the file-name is printed on the user's terminal, and the MONITOR toggle is again SET.

The name of the MONITOR file may be changed at any time by entering another MONITOR file-name command, bearing in mind that the file specified may not be on disk prior to entering the MONITOR request. If the MONITOR file is removed after the MONITOR toggle is set, the LIST/CANDE program will ignore the MONITOR request.

```
MONITOR MONFIL←
#
MONITOR MONFIL←
ERR: MONFIL
?←
YOU ALREADY HAVE A FILE BY THAT NAME.

MAKE TEST ←
FILE:TEST - TYPE:SEQ  -- CREATED
100 REC 1←
200 REC 2←
300 REC 3←
U←


#
200 REC 2 AGAIN←
300 REC 3 ONCE MORE←
100←
U←


#
*300/  /...←
U←


#
P←
200  REC 2 AGAIN
300  REC 3 ONCE MORE..


#
RESET MONITOR←
#
400 NO CHANGE HERE←
U←


#
SAVE←
FILE:TEST - TYPE:SEQ  -- SAVED.

P←
200  REC 2 AGAIN
300  REC 3 ONCE MORE..
400  NO CHANGE HERE
#
P MONFIL SQUASHED←
MONITOR 1S020 08:26 P.M. ********
 REC 1 00000100
 REC 2 00000200
 REC 3 00000300
MONITOR 1S020 08:27 P.M. ********
 00000100
 REC 2 AGAIN 00000200
 REC 3 ONCE MORE 00000300
MONITOR 1S020 08:27 P.M. ********
 REC 3 ONCE MORE.. 00000300
```

4.3.28. PRINT|P

The PRINT command is used to print a portion of or all of a file. It is different from the LIST command in that it suppresses the heading and causes CANDE to type only a number sign when finished. The formats are as follows:

```
(P|PRINT)[($|CHANGES)|file-name[/user-code]] sequence-list
    sequence-list]*|SQUASHED][#|NUMBERED]
```

The PRINT command causes the work-file, the file specified, or a sequence range within the file to be listed. When a file-name is not specified, the work-file, if present, is used for the input. The lack of a sequence option causes the entire file to be listed.

The PRINT command is equivalent to the LIST command except that the header and trailer printed by the LIST command are deleted. For further information, see the LIST command.

```
MAKE PRINTR←
FILE:PRINTR - TYPE:SEQ  -- CREATED
100 REC 1←
200 REC 2←
LIST←

FILE:PRINTR - TYPE:SEQ  --03/07/72  8:32 PM.

100  REC 1
200  REC 2

 END LIST 1.8 SEC.

PI'RINT←
100  REC 1
200  REC 2

#
REMOVE←
#
```

4.3.29. PUBLIC

The PUBLIC command allows any user to access a file for read/write (if source), or execute/only (if object). The creator may continue to access it in any manner. The format is the following:

PUBLIC[[SOURCE|OBJECT][file-name] ...] ...

For further information, see APPENDIX C.

```
LIST FILES G22←
03/07/72 RCC63YH  8:33 PM.
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

G22      GTL         18   10  03/02/72   03/07/72    10  300   10
G22      *GTL        21   21  03/02/72   03/07/72    30   30   20
#
PUBLIC SOURCE G22←
#
LIST FILES G22←
03/07/72 RCC63YH  8:33 PM.
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

G22      GTL         18   10  03/02/72   03/07/72    10  300   10  PUBLIC
G22      *GTL        21   21  03/02/72   03/07/72    30   30   20
#
```

4.3.30. PUNCH

The PUNCH command may be used to punch the contents of a work-file or a file on disk onto paper tape on the user's teletype or similarly-equipped device. The format is as follows:

PUNCH [file-name[/user-code]][RESEQ resequence-info]

The base and increment for resequencing are assumed by default to be 100, and moving of records (RESEQ 100,300) is not permitted.

After entering a PUNCH command, the user must turn on the paper tape punch. The system then sends 39 rubouts, the file-name, 40 rubouts, the contents of the file, and 40 more rubouts. Each line of data is ended with a carriage return, a line feed, and a rubout. The tape can be read back to the system using the TAPE command by initially positioning it in the first set of 40 rubouts.

Graphically, the format of the tape is equivalent to that shown below:

39 - RO	file-name CR LF
40 - RO	First-data-line CR LF
RO	second-data-line CR LF
.	.
.	.
.	.
RO	last-data-line CR LF
40 - RO	
where	CR means carriage return,
	LF means line feed, and
	RO means rubout.

```
MAKE PFILE←
FILE:PFILE - TYPE:SEQ   -- CREATED
100 REC 1←
200 REC 2←
300 REC 3←
400 REC 40'←
PUNCH←
  WAIT.


  WAIT.
PFILE
100 REC 1
200 REC 2
300 REC 3
400 REC 4

   END PUNCH .6 SEC.
```

4.3.31. REMOVE

The REMOVE command is used to remove files created by the requesting user. Its format is the following:

```
REMOVE[[SOURCE|OBJECT][file-name ...]] ...
```

If the optional words SOURCE and OBJECT are not used, both versions of the files that are named in the list are removed. The SOURCE and OBJECT options are included to indicate that the files following them in the list should have only the source or object versions removed, respectively. If only the object version of a file exists, and the source does not, then the OBJECT option must be chosen. These options apply to all files following them in the list until another option is invoked or until the end of the list. For instance,

```
REMOVE FILE1, FILE2, SOURCE FILE3, FILE4, OBJECT FILE5
```

would result in the removal of both versions of FILE1 and FILE2, the source versions of FILE3 and FILE4, and the object version of FILE5. Note that all files for which both the source and object versions are to be removed must appear in the beginning of the list. If the source version is removed, a file can only be run or executed; it may not be easily modified or recompiled in its current form.

A maximum of nine entries--i.e., file-names and uses of the SOURCE and OBJECT options--are allowed in the list. After the REMOVE command has been completed, a number sign is typed which indicates that CANDE is ready for the next command.

Reference to a non-existent file is noted by the following message:

```
ERR:file-name
```

```
MAKE FILE1←
FILE:FILE1 - TYPE:SEQ  -- CREATED
SAVE←
FILE:FILE1 - TYPE:SEQ  -- SAVED.

MAKE FILE2←
FILE:FILE2 - TYPE:SEQ  -- CREATED
SAVE←
FILE:FILE2 - TYPE:SEQ  -- SAVED.

MAKE FILE2'3 BASIC←
FILE:FILE3 - TYPE:BASIC  -- CREATED
100 END←
SAVE←
 WAIT.


FILE:FILE3 - TYPE:BASIC  -- SAVED.

COMPILE←
 COMPILING.


 END COMPILE 2.3 SEC.

SAVE←
FILE:FILE3 - TYPE:BASIC  -- SAVED.

REMOVE OBJECT FILE2←
#
REMOVE OBJECT FILE3←
ERR: WRKFILE
SAVE←
FILE:FILE3 - TYPE:BASIC  -- SAVED.


REMOVE OBJECT FILE3←
#
"REMOVE OBJECT, NOT SOURCE, VERSION OF FILE3←
REMOVE FILE1←
ERR: WRKFILE
?←
I CANNOT REMOVE THAT FILE--IT IS YOUR WORK-FILE.

REMOVE←
#
REMOVE FILE1←
#
"REMOVE ALL TRACES OF FILE1←
REMOVE OBJECT FILE2←
#
REMOVE FILE2←
#
REMOVE SOURCE FILE3←
#
```

4.3.32. RENAME

The RENAME command changes the file-name associated with the work-file. The format is as follows:

RENAME file-name

The RENAME command may be used to give the work-file a new name so that a subsequent SAVE command does not destroy an existing file. A number sign is sent to indicate that the renaming operation has been performed.

```
MAKE F1←
FILE:F1 - TYPE:SEQ  -- CREATED
RENAME F12345←
#
WHATS←
FILE F12345 (WORKFILE), TYPE SEQ, 0 RECORDS
#
SAVE←
FILE:F12345 - TYPE:SEQ  -- SAVED.

WHATS F1←
ERR: F1
?←
FILE NOT IN YOUR LIBRARY.

WHATS F12345←
FILE F12345, TYPE SEQ, 0 RECORDS, CREATED 03/07/72 (2046) SF=7
#



REMOVE←
#
```

4.3.33. REPLACE|REP

The REPLACE command allows the user to search a file or a subset of a file for the records which contain a given string and to replace occurrences of that string with another string. The format of the REPLACE command is as follows:

```
((REPLACE|REP) [FILE file-name[/user-code]]
        [FIRST][LITERAL]

        (delimiter sought-string delimiter|mnemonic)
        [WITH(delimiter replacement-string delimiter|mnemonic]
        sequence-list
        [PRINT(SEQUENCE|TEXT|SITE|FILE file-name)],)...
```

If the FILE option is specified, the designated file is searched for replacement, if it is present and the user has read/write access to it. If the FILE option is not specified, the work-file is searched for replacement, if it is present.

The FIRST option, when used, terminates the search and replace operation after the first replacement has been performed on the first record which contains the characteristics which are sought; otherwise, the entire designated portion of the file is searched for replacement.

The LITERAL option is used to specify that the element (string, constant, literal, constant-string, or literal-string) which is contained between the delimiters is to be sought and replaced as an entity.

The mnemonic option allows the user to search a file for records which contain certain special characters (ARROW, GEQ, LEQ, GTR, LSS, NEQ, EQL) and replace any of these characters with an element or replace an element in a record in a file with those characters. The interpretation of those mnemonics is $\leftarrow$, $\geq$, $\leq$, $>$, $<$, $\neq$, $=$, respectively. Caution should be exercised when replacing an element using the mnemonic option as only the element is replaced. That is, the previous data to the left and right of the replaced element remains intact.

The sequence-list option may be used to limit the area in which records containing the sought-string are to be search. By default the entire file is searched.

The PRINT option is used to specify output form. The PRINT SEQUENCE option causes only the sequence-number of the records containing the sought-string to be printed on the terminal. The PRINT TEXT option causes the records containing the sought-string to be listed on the terminal after replacement. The PRINT SITE option causes the records containing the sought-strings to be written on a high-speed printer at the computer site after replacement. The PRINT FILE file-name option causes the records containing the sought-strings to be written to a file named file-name after replacement. When working with files of file-type DATA containing sequence-numbers in characters 73-80, the sought-string and replacement-string should be of the same length to avoid problems.

Multiple replacements may be made by continuing the REPLACE command to as many lines as is necessary, duplicating the REPLACE verb at the beginning of each segment, and by following each iteration except the last with a space, a comma, and a left arrow. Only one PRINT option is allowed in the string of commands.

```
MAKE REPFIL←
FILE:REPFIL - TYPE:SEQ  -- CREATED
100 REC 1←
200 REC 3←
300 REC 5←
400 REC 7←
REPLACE/REC/WITH/RECORD←
ERR: RECORD
?←
MISSING DELIMITER OR STRING TOO LONG.



REP /REC/ WITH /RECORD/←
 WAIT.


 WAIT.

NUMBER OF STRINGS REPLACED =         4


 END REPLACE .8 SEC.

#
P←
100  RECORD 1
200  RECORD 3
300  RECORD 5
400  RECORD 7


#
REP/3/WITH/2/,
REP/5/WITH/3/,
REP/7/WITH/4/←
 WAIT.

NUMBER OF STRINGS REPLACED =         3


 END REPLACE 1.1 SEC.

#
```

```
P←
100   RECORD 1
200   RECORD 2
300   RECORD 3
400   RECORD 4


#
REPLACE /REC/ WITH /XXX/ 100 PRINT TEXT←
 WAIT.

NUMBER OF STRINGS REPLACED =         0


 END REPLACE 1.0 SEC.

#
REPLACE LITERAL /REC/ WITH /XXX/ 100 PRINT TEXT←
 WAIT.
100   XXXORD 1

NUMBER OF STRINGS REPLACED =         1


 END REPLACE 1.0 SEC.

#
REPLACE FIRST LITERAL /X/ WITH /Y/ PRINT TEXT←
 WAIT.
100   YXXORD 1

NUMBER OF STRINGS REPLACED =         1


 END REPLACE .9 SEC.

#









REMOVE←
#
```

4.3.34. RESEQ

The RESEQ command is used to change sequence-numbers or move records in a designated file or in the work-file. It has the following format:

RESEQ[file-name]resequence-info

The RESEQ command may not be used on files with file-type DATA.

CHANGE OPTION

If only a base-sequence-number is used, the entire file is resequenced using the base-sequence-number as the first sequence-number and increasing each successive number by the resequence-increment. If the base-sequence-number and/or the resequence-increment-number are not given, they are assumed by default to be 100. If a pair of sequence-numbers is used, the lines between the two sequence-numbers are resequenced using the given increment. If the resequencing processing results in a sequence-number of more than eight digits, resequencing is abandoned and the following message:

ERR:TOOBIG

is typed. This leaves the last lines of the file with incorrect sequence-numbers. A correct RESEQ command should normally be given before proceeding.

MOVE OPTION

The numeric parameters to the RESEQ command may be unsigned integers, + followed by an unsigned integer, or hyphened integers. The word TO, immediately preceding an integer, is intepreted as a hyphen, and the word END is interpreted as meaning the last record in a file. The rules for determining how CANDE interprets the numeric parameters to the RESEQ command are as follows:

An unsigned integer preceded by a + is always interpreted as a resequence-increment. If no resequence-increment is specified by the user, a resequence-increment of 100 is assumed.

An integer preceded by a - is always interpreted as the upper bound of a sequence range. The first unsigned integer which immediately precedes the hyphen is interpreted as the lower bound for the sequence range.

When only one unsigned integer appears after the RESEQ command, it is interpreted as the following:

The lower bound of a sequence range if a hyphen followed by an integer follows it,

A base for resequencing if a hyphen and an integer do not follow it.

When two unsigned integers appear after the RESEQ command verb, the first is interpreted as the lower bound of a sequence range, and the second is interpreted as the base for resequencing. If a hyphen and an integer do not follow the first unsigned integer, the sequence range is assumed to be the single record specified by the lower bound value (RESEQ 100,300 would resequence record 100 starting at a resequence base of 300, or, in effect, would move the record at position 100 to position 300).

When both a sequence range and a resequence base are specified, records can be lost from the file if the resequenced records have sequence-numbers which are identical with the sequence-numbers of other records in the file. (RESEQ 300,900 would produce a record with a sequence-number of 900. If there had already been a record with a sequence-number of 900 in the file, the old 900 record would be replaced by the new 900 record.) The user should therefore exercise caution in moving records within a file.

```
MAKE RF←
FILE:RF - TYPE:SEQ  -- CREATED
190 REC 1←
200 REC 2←
20002 REC 3←
403050 REC 4←
SAVE←
 WAIT.


FILE:RF - TYPE:SEQ  -- SAVED.

RESEQ←
 WAIT.


 END RESEQ 1.0 SEC.

P←
100  REC 1
200  REC 2
300  REC 3
400  REC 4


#
REMOVE←
#
RESEQ RF RESEQ 100 300 +10←
ERR: RESEQ
?←
ONE OF YOUR PARAMETERS IS ILLEGAL.

RESEQ RF 100 300 +10←
 WAIT.


 END MERGE 1.0 SEC.

P RF←
190  REC 1
200  REC 2
20002  REC 3
403050  REC 4


#
RESEQ RF←
 WAIT.


 END RESEQ .8 SEC.
```

```
RESEQ RF 100 300←
  WAIT.

  END MERGE 1.2 SEC.

P RF←
200   REC 2
300   REC 1
400   REC 4


#
REMOVE RF←
#
```

4.3.35. RESET

The RESET and SET commands may be used to change certain attributes of the remote terminal's interaction with the system. For a complete discussion, see the SET command. The format of the RESET command is as follows:

RESET option-list

```
RESET CONCIE'SE,BUSY←
#
RESET MONITOR←
#
```

4.3.36. RMERGE

The RMERGE command allows the user to reverse the actions of the MERGE command. When a library file is MERGEd into a work-file, and a record with the same sequence-number appears in both files, the work-file record is the record saved. In certain instances, it is desirable to accomplish the reverse of the process; i.e., to eliminate the work-file record and retain the library-file record. The format for the RMERGE command is the following:

```
RMERGE file-name[/user-code]sequence-list[RESEQ resequence-info]
```

For further information, see the MERGE command.

The RMERGE command may not be used on files with file-type DATA.

```
MAKE FILE1←
FILE:FILE1 - TYPE:SEQ  -- CREATED
100F 1 REC 1←
300F 1 REC 2←
500F 1 REC 3←
SAVE←
 WAIT.

FILE:FILE1 - TYPE:SEQ  -- SAVED.

MAKE FILE2←
FILE:FILE2 - TYPE:SEQ  -- CREATED
200F 2 REC 1←
300F 2 REC 2←
400F 2 REC 3←
SAVE←
 WAIT.


FILE:FILE2 - TYPE:SEQ  -- SAVED.

MAKE FILE3←
FILE:FILE3 - TYPE:SEQ  -- CREATED
COPY FILE1←
 WAIT.
3 RECORDS COPIED (LAST RECORD COPIED=500)


 END COPY 1.0 SEC.

MERGE FILE2←
 WAIT.
2 RECORDS MERGED (LAST RECORD MERGED=400)


 END MERGE 1.1 SEC.

P←
100  F 1 REC 1
200  F 2 REC 1
300  F 1 REC 2
400  F 2 REC 3
500  F 1 REC 3


#
```

```
REMOVE←
#
MAKE FILE4←
FILE:FILE4 - TYPE:SEQ  --- CREATED
COPY FILE1←
 WAIT.
3 RECORDS COPIED (LAST RECORD COPIED=500)


 END COPY 1.1 SEC.

RMERGE FILE2←
 WAIT.
3 RECORDS MERGED (LAST RECORD MERGED=400)


 END MERGE 1.1 SEC.

P←
100 F 1 REC 1
200 F 2 REC 1
300 F 2 REC 2
400 F 2 REC 3
500 F 1 REC 3


#
```

4.3.37. RUN|R

The RUN command causes CANDE to take whatever actions are necessary to run the specified program. It has the following format:

```
(R|RUN)[file-name[/user-code]][compiler-file-time|:first-letter-of-
    compiler-file-type]program-parameter-info
```

CANDE runs the specified file by executing the object version if it is available, or, if there is no object version, by compiling and executing the source version, if it is available. If a file-name is not included in the RUN command, the work-file is executed.

The RUN command may be preceded by EQUATE commands or immediately followed by program-parameter-info. If no program-parameter-info appears, the following parameters are assumed:

```
PROCESS = 2
IO      = 2
STACK   = 512
COMMON  = 0
```

If RUN file-name requires the file-name to be compiled, the resulting object file, if any, is executed but not saved. However, if the work-file is compiled with RUN, the object file is kept in the work-file so that, if the work-file has been changed since it was created or last saved, both versions can be saved.

The compiler-file-type can be ALGOL, BASIC, COBOL, CODASYL, DYNAMO, ESPOL, FORTRAN, GTL, TSPOL, XALGOL, or an abbreviation consisting of a colon followed by the first letter of the compiler-file-type, if unambiguous. It is required for files of non-compiler-file-type, but in any case, it overrides the original file-type and can be used for a work-file with any file-type. For further information about the compilation process, see the COMPILE command.

If it has been arranged through the user of the LOCK, UNLOCK, PUBLIC, and GUARD commands, object files belonging to another user can be executed by including that user's user-code in the RUN command. Note that one user can only execute the object version of another user's file. The user cannot directly compile the source version, even if available to him, without copying it first. Therefore, a RUN command specifying a file in another user's library is equivalent to an EXECUTE command specifying that file. The RUN command may be preceded with EQUATE commands and followed by program-parameter-info.

The messages typed by CANDE are similar to those typed for the COMPILE and EXECUTE commands. Thus, if compiling is necessary, the messages printed are the following:

```
COMPILING
(any syntax errors here)
END COMPILE n.m SEC.
```

The messages printed for execution are the following:

```
RUNNING
(any output here)
END file-name n.m SEC.
```

In case the user decides to prematurely discontinue a program, and the program is not typing at the time, he may depress WRU(CTRL E). If the program is typing at the time, it may be necessary to depress BREAK to stop the typing; if BREAK itself does not discontinue the program, he may then depress WRU.

```
MAKE EXAMPL BASIC←
FILE:EXAMPL - TYPE:BASIC  -- CREATED
100LET X=Y=4←
150PRINT "X="X,"Y=",Y←
175LET ABC=DEF←
176LAST LINE WILL CAUSE A SYNTAX ERROR, AS WILL THIS ONE←
200END←
RUN←
 WAIT.

 COMPILING.
     175 UNRECOGNIZABLE STATEMENT OR MISSING EQUAL.
     176 UNRECOGNIZABLE STATEMENT OR MISSING EQUAL.

 ERR COMPILE 2.4 SEC.

175←
176REM NOW GOOD←
RUN←
 WAIT.

 COMPILING.

 END COMPILE 2.4 SEC.

 RUNNING

X= 4            Y=              4

 END EXAMPL .4 SEC.

RUN←
 RUNNING

X= 4            Y=              4

 END EXAMPL .5 SEC.

50READ Y←
100 LET X=Y+5←
RUN←
 WAIT.
```



```
 COMPILING.
```

```
RUN

 END COMPILE 2.7 SEC.

 RUNNING


-OUT OF DATA,NEAR LINE 00000050


 ERR EXAMPL .6 SEC.

*50;READ;INPUT←
RUN←
 WAIT.


 COMPILING.


 END COMPILE 2.8 SEC.

 RUNNING


?7.9←
X= 12.9          Y=                7.9


 END EXAMPL .4 SEC.

RUN←
 RUNNING


?4321.90654←
X= 4326.907      Y=                4321.907


 END EXAMPL .4 SEC.

REMOVE←
#
```

4.3.38. SAVE

The SAVE command causes the current copy of the current work-file to be saved. Its format is as follows:

```
SAVE [save-factor][file-type]
```

The SAVE command does not clear the work-file but it does establish a permanent disk file which reflects the work-file at the time the SAVE command is processed. Any previous copies of the file are removed. If the work-file is saved more than once, only the version last saved remains on disk.

If the work-file has not been changed since being initially created or last saved, the SAVE command is ignored. If the SAVE is performed and there is an object version of the work-file which agrees with the source version, both versions are saved. Otherwise, the object file is not saved and the disk space allocated for it is returned to the system. Whenever a SAVE is done, the old versions of both the source and object files are removed.

If save-factor and/or file-type are specified, the work-file will be saved with the revised specifications. The work-file itself will not be altered, however.

Each file is saved as a locked file unless the user has previously specified a different form of file security through the use of the GUARD, LOCK, PUBLIC, and UNLOCK commands.

When the file is saved, CANDE types the following message:

```
FILE:file-name - TYPE: file-type -- SAVED
```

If there is no work-file, CANDE types the following message:

```
ERR:WRKFILE
```

SAVE

```
MAKE TESTSV←
FILE:TESTSV - TYPE:SEQ   -- CREATED
100 REC 1←
200 REC 2←
300 REC 3←
SAVE←
  WAIT.


FILE:TESTSV - TYPE:SEQ   -- SAVED.

400 REC 4←
500 REC 5←
200←
*100/    /   .....←
P←
100   REC 1  .....
300   REC 3
400   REC 4
500   REC 5


#

P TESTSV←
100   REC 1
200   REC 2
300   REC 3


#
SAVE←
FILE:TESTSV - TYPE:SEQ   -- SAVED.

P TESTSV←
100   REC 1   .....
300   REC 3
400   REC 4
500   REC 5


#
REMOVE←
#
REMOVE TESTSV←
#
```

4.3.39. SCHEDULE|SCH

The SCHEDULE command is used to schedule a series of commands to be processed independently of the user. The format of the SCHEDULE command is the following:

(SCHEDULE|SCH)[file-name[/user-code]] TO file-name [AFTER integer]

The first file-name in the format, called file 1 here, specifies the input file to be scheduled. This file may be altered or removed after the SCHEDULE command has been acknowledged by CANDE. The second file-name in the format, called file 2 here, must not be present on disk before the SCHEDULE command is entered. All output from file 1 that would normally be sent to the remote user's terminal is placed in this file. File 2 is created when the SCHEDULE command is given.

Once a SCHEDULE command has been successfully entered, the user is said to occupy a schedule line, or to be performing a scheduled task.

The AFTER integer clause, if used, specifies the time of day after which the task may be initiated. From 0001 to 0759 is considered after 2400. If the AFTER clause is not specified, the task is started as soon as possible.

The file 1 to be scheduled contains CANDE commands, possibly intermixed with input data sets for programs. The input data sets must appear in the proper place, such as after a RUN or EXECUTE command. At the time the scheduled job is initiated, CANDE sequentially processes each command and the programs in the schedule line receive the remote input from the scheduled input file. A copy of each input record and all remote output are written into file 2. An error condition, such as an invalid command, duplicate file, syntax error on a compilation, or error program termination, other than a reference to a non-existent file in a REMOVE command, causes termination of the scheduled job, unless the NOSTOP option is set. In file 1, trailing blanks are ignored: there is an implicit carriage return immediately after the last non-blank character or as the first character of a blank record. The HELPFUL option is automatically set for a SCHEDULE line.

The following commands are invalid in a schedule file: HELLO, PUNCH, TAPE, and special functions of a question mark.

A user may have more than one schedule line simultaneously in the schedule queue. Of course, he may differentiate among them, because they all have unique output file-names.

```
MAKE SCHLR DATA←
FILE:SCHLR - TYPE:DATA  -- CREATED
?DATA←
 OK
MAKE BBB BASIC←
SEQ←
READ N!DEL
INPUT N←
PRINT N←
READ N←
PRINT N←
END←
←
?END!DEL
RUN←
+1.2345←
?END←
#
SAVE←
 WAIT.


FILE:SCHLR - TYPE:DATA  -- SAVED.









LOAD SCHLR←
FILE:SCHLR - TYPE:DATA  -- LOADING


END LOAD
P←
MAKE BBB BASIC
SEQ
INPUT N
PRINT N
READ N
PRINT N
END

RUN
+1.2345


#
SCHEDULE TO XXX←
 WAIT.

 END SCHEDUL .9 SEC.
```

```
STATUS XXX←
RUNNING(9)
STATUS XXX←
RUNNING(9)
STATUS XXX←
DONE.
P XXX←

MAKE BBB BASIC
FILE:BBB - TYPE:BASIC  -- CREATED
SEQ
100INPUT N
200PRINT N
300READ N
400PRINT N
500END
600
RUN
 WAIT.

 COMPILING.


 END COMPILE 2.4 SEC.

 RUNNING


?+1.2345
 1.2345

-OUT OF DATA,NEAR LINE 00000300


 ERR BBB .6 SEC.


BYE
ERR: NO SAVE
WORKFILE HAS UNSAVED RECORDS IN IT - PLEASE SAVE OR REMOVE IT.


BYE
 C&E USE 1.6 SEC.
 EXECUTE 4.8 SEC.
 IO TIME 10.9 SEC.
 GOODBYE RCC63YH
03/08/72
```

```
#
REMOVE XXX←
#
SCH TO XXX←
 WAIT.

 END SCHEDUL .9 SEC.

STOP XXX←
RUNNING(9)
P XXX←

MAKE BBB BASIC
FILE:BBB - TYPE:BASIC  -- CREATED
SEQ
100INPUT N
200PRINT N
300READ N
400PRINT N
500END
600
RUN
 WAIT.
**TASK TERMINATED BY USER

-USER DS-ED,NEAR LINE 00054000

 ERR LIST .8 SEC.

BYE
ERR: NO SAVE
WORKFILE HAS UNSAVED RECORDS IN IT - PLEASE SAVE OR REMOVE IT.

BYE
 C&E USE 1.6 SEC.
 EXECUTE .8 SEC.
 IO TIME 2.2 SEC.
 GOODBYE RCC63YH
03/08/72
```

4.3.40. SEQ|S

The SEQ command is used to request CANDE to generate the sequence-numbers for the user as input is entered to the system. It has the following format:

(SEQ|S)[base-sequence-number][+resequence-increment]

If the resequence-increment is missing, the one last entered for this work-file is used; if none has been entered, 100 is assumed. If base-sequence-number is missing, the highest sequence-number currently in the work-file plus the current resequence-increment is used. When automatic sequencing is used, the user must wait (if necessary) for the sequence-number to be typed before entering data. Automatic sequencing is terminated by entering a group mark or WRU immediately following the sequence-number; not even backspace characters may be used on this last line. CANDE types a number sign to indicate that it is ready for further input.

```
MAKE X BASIC←
FILE:X - TYPE:BASIC  -- CREATED
SEQ 1000+10←
1000DIM X(20)←
1010LEY Y=3*X(3)←
1020LET Z=Y**2←
1030←
#
*1010/Y/T←
150INPUT P←
SEQ←
1030I'LET I=I+1←
1040END←
1050←
#
P←
150 INPUT P
1000 DIM X(20)
1010 LET Y=3*X(3)
1020 LET Z=Y**2
1030 LET I=I+1
1040 END

#
REMOVE←
#
```

4.3.41. SET

The user may change certain options concerning the remote terminal's interaction with the system through the use of the SET and RESET commands. The format of these commands is as follows:

```
(RESET|SET) (ALLOWMSG|BUSY|CONCISE|HELPFUL|MONITOR|NONSTOP|
        QUICKBYE|QUICKLOG) ...
```

The interpretation of the options is as follows:

ALLOWMSG	"SET ALLOWMSG" allows TO messages to be received even while the terminal is busy.
BUSY	"SET BUSY" inhibits TO messages being received even while the terminal is not busy.
CONCISE	"SET CONCISE" inhibits the printing of many CANDE messages on the terminal.
HELPFUL	"SET HELPFUL" causes more complete error messages to be printed whenever an error occurs, just as if a question mark input message had been entered.
MONITOR	"RESET MONITOR" inhibits the updating of the MONITOR file (see the MONITOR command).
NOSTOP	"SET NOSTOP" inhibits the flushing of a SCHEDULE file if an error occurs.
QUICKBYE	"SET QUICKBYE" inhibits almost all of the messages normally printed at log-out time.
QUICKLOG	"SET QUICKLOG" inhibits almost all of the messages normally printed at log-in time.

By default, all the options are RESET. A user's options carry over to any SCHEDULE files he starts. Since they are associated with his user-code, they remain in effect until changed or a new REMOTE/USERS file is loaded.

The settings of the options may be interrogated through the use of the TYPE OPTIONS command.

```
TYPE OPTIONS-
ALLOWMSG RESET
BUSY   RESET
CONCISE RESET
HELPFUL RESET
MONITOR RESET
MONITOR FILE: *NONE*
NOSTOP RESET
QUICKBYE RESET
QUICKLOG RESET
#
MONITOR MONFIL-
#
SET QUICKLOG-
#
RESET NOSTOP-
#
SET CONCISE-
#
SET HELPFUL-
#
TYPE OPTIONS-
ALLOWMSG RESET
BUSY   RESET
CONCISE SET
HELPFUL SET
MONITOR SET
MONITOR FILE: MONFIL
NOSTOP RESET
QUICKBYE RESET
QUICKLOG SET
#
RESET CONCISE,HELPFUL,MONITOR,QUICKLOG-
#
```

4.3.42. SS

The SS command allows the remote user or console operator to send a message to any user with the ability to receive it. The format of this command is as follows:

[?](TO|SS)(logical-line|user-code|SPO|SITE)[message]

For further information, see the TO command syntax.

```
SS SPO PLEASE SEND A MESSAGE TO MY TERMINAL...TESTING IT.←
#
SS 23 MESSAGE TO MYSELF←
** FROM RCC63YH (23)      MESSAGE TO MYSELF

#
SS 32 IS ANYBODY THERE?←
 NOT ON
SS RCC63YH MESSAGE TO ME←
** FROM RCC63YH (23)      MESSAGE TO ME

#
SS INVUSER NOT THERE←
 NOT ON
```

4.3.43. STATUS

The STATUS command is used to obtain the present condition of a user's jobs or schedule lines. The format of the STATUS command is the following:

```
[?]STATUS[file-name]
```

The ? must be used if and only if the user is currently running a program. If the file-name term is included CANDE will respond in one of the following manners:

SCHEDULE	If the task has not yet been initiated.
RUNNING(n)	If the task is running. The integer n indicates the record number of the last record read in the input file.
DONE	If the task is completed.
ERR:file-name	If the file-name specified is not on disk or is not a schedule output file.

For further information and examples, see the SCHEDULE command.

If the file-name term is not included, and the user is currently running or compiling a program from the terminal, information concerning the status of the program or compiler will be displayed on the terminal.

4.3.44. STOP

A scheduled task may be terminated with the STOP command. Its format is as follows:

```
[?]STOP file-name
```

The ? must be used if and only if the user is currently running a program.

CANDE will respond in the same manner as to the STATUS command, then terminate the task, if possible. The schedule output file remains on disk and may be handled as desired.

For further information and examples, see the SCHEDULE command.

4.3.45 TAPE

The TAPE command is used to specify that a paper tape file is to be read from the remote terminal. Its format is as follows:

(?TAPE| TAPE [SEQ[base-sequence-number][+resequence-increment]])

The system sends the message OK and then sends an X-ON character which initiates the tape reader if it is set to AUTO-START. If the reader is not set for AUTO-START, the user must manually start the reader after the OK message has been sent. After the tape has been read, the user must turn off the reader and enter ?END← to terminate TAPE mode. The system responds with a number sign to indicate that it is no longer in TAPE mode. All system output to a terminal in TAPE mode is suppressed between the time the X-ON and the number sign are sent. The tape itself should be prepared in the format described under the PUNCH command. The tape should be positioned somewhere within the group of rubouts between the file-name and the data.

When the SEQ option is not used, the data is treated the same way as ordinary input. Each line must have a sequence-number, but the lines may be out of order. Corrections, in the form of FIX commands or retyped lines, may be included on the tape. Other system commands may not be included on the tape.

When SEQ is used, the first line is given a sequence-number equal to the base-sequence-number, and the sequence-number for each succeeding line is increased by the resequence-increment. If base-sequence-number or response-increment is missing, the action is similar to that in the SEQ command. The lines must be in order and they cannot contain sequence-numbers or any CANDE commands, including FIX commands.

A work-file must be specified by the use of LOAD or MAKE commands before the TAPE command without a preceding ? may be entered. If a work-file is not open when required, the system types the following message:

ERR:WRKFILE

and ignores the TAPE command.

Paper tape may also be entered to a program by preceding the word TAPE with a question mark. In this case, all programmatic and system output is

saved until after the ?END is entered. All output to the terminal appears after the number sign in the order in which it would have been sent if the terminal were not in TAPE mode. Also, the program receives the ?END← message to signify end of TAPE mode to it and normally would be written to discard the message.

```
TAPE←
ERR: NO FILE
MAKE TAPFIL←
FILE:TAPFIL - TYPE:SEQ  -- CREATED
TAPE←
OK
100 REC 1
200 REC 2
300 REC 3
400 REC 4
?END←

WAIT.


#
P←
100   REC 1
200   REC 2
300   REC 3
400   REC 4


#
REMOVE←
#



CALL WIPL←

PLEASE WAIT.
TYPE HELP IF YOU HAVE QUESTIONS  (WIPL VERSION 1.5)
??TAPE←
OK
1.1 THI'''TYPE"THIS WAS ENTERED FROM TAPE"
1.2 THIS WILL CAUSE A SYNTAX ERROR
1.3 STOP
?END←
UNRECOGNIZABLE STATEMENT
?LIST←
1.1 TYPE"THIS WAS ENTERED FROM TAPE"
1.3 STOP

?RUN←
THIS WAS ENTERED FROM TAPE
STOP AT STATEMENT 1.3
?QUIT←


 END WIPL 2.8 SEC.
```

4.3.46. TIME

The TIME command allows a remote user to interrogate the general statistics concerning his use of the system during the current session. The format of the TIME command is as follows:

TIME

```
TIME←
 USER IS RCC63YH LINE 23
 TIME IS  8:00 PM.
 C&E USE 29.7 SEC.
 EXECUTE 1 MIN, 01.8 SEC.
 IO TIME 2 MIN, 25.3 SEC.
03/08/72

#
```

4.3.47. TO

The TO command is used to send a message to other attached users or to the computer operator at the central site. The format is as follows:

[?](TO|SS)(logical-line|user-code|SPO|SITE)[message]

The question mark must be used if the user is running a program to distinguish the message from program input data.

Depending on whether SPO, a user-code, or logical-line is used in the command, the message is typed at the operator's console, at the terminals of any users logged on with the specified user-code, or at the terminal connected to the specified line, respectively. In all cases, the message is sent in the following format:

FROM sender's-user-code(sender's-logical-line-number) message

If the message is sent successfully, CANDE types either a number sign if the sending user is not currently running a program or types a dollar sign if the user is currently running a program.

If there are no users connected with the given user-code, or if the specified line is not logged in, CANDE responds with the following message:

NOT ON

If the receiving user is running a program and has not set the ALLOWMSG option, or has set the BUSY option, CANDE sends the following message to the sending user:

BUSY

```
TO 34 HI JOE←
 NOT ON
TO INVUSER NOT THERE←
 NOT ON
TO 23 HI ME←
** FROM RCC63YH (23)      HI ME

#
TO RCC63YH HELLO THERE AGAIN←
** FROM RCC63YH (23)      HELLO THERE AGAIN

#
RUN PROGJ←
 RUNNING

?TO SPO PLEASE DISMOUNT OUTPUT TAPE NOT'W←
 4←

 $
?TO 23 HELLO ME←
 $
** FROM RCC63YH (23)      HELLO ME


 END PROGJ 1.2 SEC.
```

4.3.48. TYPE

The TYPE command is used to change the file-type which is associated with the work-file. The format of the TYPE command is the following:

```
TYPE(file-type|first-letter-of-file-type)
```

The file-type options allowed in this command are ALGOL, BASIC, COBOL, CODASYL, DATA, DYNAMO, ESPOL, FORTRAN, GTL, INFO, LOCK, SEQ, TSPOL, and XALGOL; however, a work-file's file-type may be changed from sequential to DATA only. All of the file-types may be abbreviated with the first letter of the file-type preceded by a colon, if nonambiguous.

If a work-file has not been declared by the user, the following error message is typed:

```
ERR:WRKFILE
```

CANDE responds with a number sign after the file-type has been changed.

```
MAKE F23 SEQ←
FILE:F23 - TYPE:SEQ  -- CREATED
WHATS←
FILE F23 (WORKFILE), TYPE SEQ, 0 RECORDS
#
TYPE BASIC←
#
100 TYPE SIN(1)←
200 END←
RUN←
 WAIT.


 COMPILING.
     100 UNRECOGNIZABLE STATEMENT OR MISSING EQUAL.


 ERR COMPILE 2.2 SEC.

*100/TYPE/PRINT←
RUN←
 WAIT.


 COMPILING.


 END COMPILE 2.2 SEC.

 RUNNING


 0.841471


 END F23 .4 SEC.

WHATS←
FILE F23 (WORKFILE), TYPE BASIC, 2 RECORDS
#
SAVE←
FILE:F23 - TYPE:BASIC  -- SAVED.
```

4.3.49. TYPE OPTIONS

The current setting of the remote terminal options manipulated with the SET, RESET, and MONITOR commands may be interrogated through the use of the command TYPE OPTIONS. Its format is as follows:

```
TYPE OPTIONS
```

For examples of the use and manipulation and interrogation of remote terminal options, see the SET command.

4.3.50. UNLOCK

The UNLOCK command allows any user to access a file- for read only (if source version), or execute only (if object version). The creator may continue to access it in any manner. Its format is the following:

UNLOCK[[SOURCE|OBJECT][file-name] ...] ...

For further information, see APPENDIX C.

```
LIST FILES G22←
03/08/72 RCC63YH  8:10 PM.
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

G22      GTL         18   10  03/02/72   03/07/72    10  300   10  PUBLIC
G22      *GTL        21   21  03/02/72   03/07/72    30   30   20
#
UNLOCK G22←
#
LIST FILES G22←
03/08/72 RCC63YH  8:11 PM.
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

G22      GTL         18   10  03/02/72   03/08/72    10  300   10  UNLOCKD
G22      *GTL        21   21  03/02/72   03/08/72    30   30   20  UNLOCKD
#
LOCK SOURCE G22←
#
LIST FILES G22←
03/08/72 RCC63YH  8:12 PM.
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

G22      GTL         18   10  03/02/72   03/08/72    10  300   10
G22      *GTL        21   21  03/02/72   03/08/72    30   30   20  UNLOCKD
#

UNLOCK SOURCE G22←
#
LIST FILES G22←
03/08/72 RCC63YH  8:13 PM.
NAME     TYPE      RECS SEGS   CREATED    ACCESSED   W/R  W/B  S-F LOCKD BY

G22      GTL         18   10  03/02/72   03/08/72    10  300   10  UNLOCKD
G22      *GTL        21   21  03/02/72   03/08/72    30   30   20  UNLOCKD
#
```

4.3.51. UPDATE

The UPDATE command allows the user to update his workfile with the latest additions and changes. The format of the UPDATE command is as follows:

(UPDATE|U)

Generally, the UPDATE command is implicitly invoked by CANDE more often than the user invokes it explicitly. At any point in which the work-file must be brought physically into order and is not currently in order, an implicit UPDATE command is invoked. Usually, unless the CONCISE option is set, when this occurs, the following message is typed:

WAIT

However, this is not the only case in which this message is typed to the terminal.

The UPDATE command may be used explicitly by the user quite advantageously in many cases to ensure that the work-file is placed into order and updated with the latest additions and changes. FIX errors are printed at the time the work-file is updated, and thus would be printed, if any exist, when an UPDATE command is issued. If some other operation is used, instead, and FIXes are in error, computer time may be wasted due to an incorrect copy of the work-file being used.

It is advantageous for the user to be aware of the manner in which CANDE actually maintains work-files in order to prevent unnecessary recopying of work-files.

CANDE maintains internal files for each user's work-file. These files are named the following:

1P line-number	(record pointers to disk tank)
1S line-number	(sorted, updated, record pointers)
1T line-number	(record pointers to new source file)
01S line-number	(object version of work-file)

They are maintained in the following manner:

- Inputs from a remote terminal are placed into the 1P file; when enough records are acquired or updating is required, the 1P file is transferred into the 1S file.
- when an updating operation is required, the 1S file is transferred into the 1T file.
- when the work-file is compiled, the object code is placed into the 01S file.

The process of updating requires that all additions and/or changes to a user's work-file be merged with the version of the file which previously existed, since no holes are permitted in a user's file. A work-file consists of a set of contiguous records, arranged in a serial fashion. The merging process may, or may not, require that the old version of the file be re-copied in order to produce a valid new version.

The old version of the file must be re-copied whenever either of the following conditions are satisfied:

A new record must be inserted into (placed between two existing records) or added to (placed before the first or after the last existing record) the file.
An old record is deleted from the work-file, and is not replaced with another record with the same sequence-number.

The old version need not be re-copied whenever only the following conditions are satisfied, and not the preceding ones:

Records are FIXed.
Records are replaced with other records with the same sequence-numbers.

The exception to this rule is as follows:

A re-copy is always required after the first change is made to a loaded or saved file.

Therefore, frequent saving of files, followed by additional changes to the work-file, require more time than frequent updating of the work-file. The difference in processor time becomes more dramatic as the file size increases.

Consider the following example as a case in point:

File NUFILE is a relatively large library file, containing approximately 2000 records. To alter this file, the user would normally LOAD the file, make the necessary corrections, and then SAVE the file.

```
WHATS NUFILE←
FILE NUFILE, TYPE SEQ, 2014 RECORDS, CREATED 03/16/72 (1108)SF = 99

LOAD NUFILE←
FILE:NUFILE - TYPE:SEQ -- LOADING
2014 RECORDS LOADED.
END LOAD 6.2 SEC.
```

The loading process prepares a table relating the sequence-numbers of the file records to their position in the file. The file has not yet been copied, so that only one version of the file exists on the disk. For instance,

```
PRINT 23100 ←
23100   MSG2(WORK,STRTIME,FINTIME); TWXOUT(LL,WORK[0],45,  2);

#
```

This is one of the records which requires changing. Since a COPY command is required after the first change to a loaded file, it makes no difference which records are altered here. For instance,

```
FIX 23100 /MSG2/MSG3←
```

A request to LIST CHANGES causes CANDE to call out an UPDATE routine. A re-copy is required here in order to produce a second version of the library file while maintaining the integrity of the original file. For instance,

```
LIST CHANGES←

FILE:NUFILE - TYPE:SEQ -- 03/19/72  6:43 PM.
23100      MSG3(WORK,STRTIME,FINTIME);  TWXOUT(LL,WORK[0],45,  2);
END LIST 11.3 SEC.
```

The large amount of processor time, as shown above, reflects the fact that the file is relatively large and has been copied in its entirety.

Observe the reduction in processor time as compared with the previous example.

After the file is saved, only one file is available for use (the work-file replaces the previously-saved file, and another change to the file requires another re-copy. For instance,

```
SAVE←
FILE:NUFILE - TYPE:SEQ -- SAVED.
FIX 23100   /45/50←
LIST CHANGES←
FILE:NUFILE - TYPE:SEQ -- 03/18/72  6:44 PM.
231000  MSG3(WORK,STRTIME,FINTIME); TWXOUT(LL,WORK[1],50, 2);
 END LIST 11.4 SEC.
```

If the file is not SAVEd, a re-copy is not mandatory for this change, and a considerable saving in processor time is realized. Therefore, unnecessary SAVE should be avoided whenever possible to take advantage of the faster update methods available for the work-file. The CANDE command UPDATE is used to update the work-file without printing any information on the user's terminal, and thus to facilitate this action.

```
MAKE UP SEQ←
FILE:UP - TYPE:SEQ   -- CREATED
100REC 1←
200 REC 2←
300 REC 3←
UPDATA'E←


#
WHATS UP←
ERR: UP
?←
FILE NOT IN YOUR LIBRARY.

WHATS←
FILE UP (WORKFILE), TYPE SEQ, 3 RECORDS
#
U←
#



400 REC 4←
U←
```



```
#
```

4.3.52. WHATS

The WHATS command returns the file-name, tile-type, size, creation-date, creation-time, and save-factor for a file. Also, for a file saved on disk, it shows the number of records it contains. The WHATS command has the following format:

```
WHATS [OBJECT|SOURCE][file-name]
```

If the file-name is not included, and a work-file is open, the answer returned is as follows:

```
FILE: file-name (WORKFILE),TYPE: file-type,number RECORDS
```

Otherwise, if the file-name is included and denotes a file currently on disk, the answer returned is as follows:

```
FILE:file-name,TYPE file-type, number RECORDS,CREATED nn/nn/nn (time)
     SF = integer
```

If the word OBJECT immediately follows the WHATS command, CANDE searches for the OBJECT version of the file. Omission of the word OBJECT implies a search for the source version. If no file-name follows the WHATS command, the information concerning the work-file is listed.

```
MAKE FILE2 ALGOL←
FILE:FILE2 - TYPE:ALGOL  -- CREATED
WHATS←
FILE FILE2 (WORKFILE), TYPE ALGOL, 0 RECORDS
#
SAVE←
FILE:FILE2 - TYPE:ALGOL  -- SAVED.

WHATS FILE2←
FILE FILE2, TYPE ALGOL, 0 RECORDS, CREATED 03/08/72 (2018) SF=7
#
1BEGIN←
2END.←
WHTS←
ERR: WHTS
WHATS←
 WAIT.


FILE FILE2 (WORKFILE), TYPE ALGOL, 2 RECORDS
#
SAVE←
FILE:FILE2 - TYPE:ALGOL  -- SAVED.

WHATS FILE2←
FILE FILE2, TYPE ALGOL, 2 RECORDS, CREATED 03/08/72 (2019) SF=7
#
COMPILE←
 COMPILING.


 END COMPILE 1.9 SEC.

SAVE←
FILE:FILE2 - TYPE:ALGOL  -- SAVED.

WHATS OBJECT FILE2←
FILE OFILE2, TYPE ALGOL, 9 RECORDS, CREATED 03/08/72 (2019) SF=8
#
WHATS FILE2←
FILE FILE2, TYPE ALGOL, 2 RECORDS, CREATED 03/08/72 (2019) SF=7
#
REMOVE FILE2←
ERR: WRKFILE
REMOVE←
#
REMOVE FILE2←
#
WHATS FILE2←
ERR: FILE2
?←
FILE NOT IN YOUR LIBRARY.
```

4.3.53. ?

The ? command is used to request special actions by CANDE. Its format is as follows:

```
?[DATA|END|SS|(STATUS|STOP)file-name|TAPE|TO]
```

The special functions which may be requested from CANDE are as follows:

(empty)	Requests further information concerning last message from CANDE.
DATA	Places the line in DATA mode. In DATA mode, input is placed into the end of the work-file, which must be of file-type DATA.
END	Terminates DATA or TAPE mode.
SS	Sends a message when user's line is attached to a program.
STATUS [file-name]	Requests information about a job or scheduled task when user's line is attached to a program.
STOP file-name	Terminates a previously-scheduled task when user's line is attached to a program.
TAPE	Initiates programmatic TAPE mode.
TO	Sends a message when user's line is attached to a program.

5. FILE HANDLING ON THE B5700 TIME SHARING SYSTEM

5.1. General

All users of the B5700 Time Sharing System must, of necessity, be concerned with various files. The degree of concern varies between users and depends on the nature and complexity of their objectives. The task of creating, compiling, and debugging a program usually involves a work-file only. The fact that a work-file is a temporary disk file is of little concern and is almost totally transparent to the user. If a user should SAVE his work-file for later use, a "permanent" disk file is created. This fact may also be of little concern unless disk space should become scarce or his file should disappear, for various reasons. When an executing program causes small amounts of data to be read from or written on the user's terminal, the fact that a programmatic remote file is involved may also be of small concern. However, when a program is required to transfer data directly from some file, other than the terminal, many aspects of file-handling become of vital concern.

5.2. Storage Media Available

A very important aspect of every file is the particular medium on which it is stored or by which it is accessed. The available media are disk, magnetic tape, punched cards, high speed line printer listings, remote terminals (including the hard copy listing and punched paper tape), and Calcomp plots. Line printer listings, remote terminal listings, and Calcomp plots are output media only; i.e., once a file exists on these media there is no automatic method of getting it back into the system. Disk, magnetic tape (hereafter referred to as tape), punched cards, and punched paper tape may be used for both input and output.

5.3. Comparison of Different Storage Media

Almost every program requires at least one input file and one output file. In batch mode these are usually punched cards for input and the line printer for output. The remote user usually desires both of these files to be his remote terminal--his keyboard for input and his printer for output. Alternatively, he may use punched paper tape for either input or output via the TAPE and PUNCH commands as previously described. Paper tape is the

most economical medium on which to store relatively short, infrequently-needed source language or data files. The terminal keyboard and printer are the easiest means for effecting small quantities of programmatic input and output. Such files are called remote terminal files and are described more fully in Section 5.4.3.

When source-language or data files are needed quite frequently, it is more convenient to have them stored on disk. This saves the time required to read them from paper tape each time they are needed. Disk files are by far the most convenient storage medium for the remote user, since they are immediately available. However, disk is the most expensive medium and the one in shortest supply. For these reasons only those files of moderate size that are needed frequently should be stored on disk. Good housekeeping practices and the procedures for periodically removing certain files are described in Section 5.4.1.2. Although every effort is made to preserve the integrity of disk files, each user should maintain his own backup copy in some other medium. Disk catastrophes are rare, but they can, and do, happen.

Large and infrequently-needed files, which should not be stored on disk, may be kept on magnetic tape, the next most convenient medium for the remote user. One reel of tape can hold more than one million computer words of information, or many, many short files. Although the user actually controls the reading and writing process, the machine room operator must mount and dismount the proper reel on one of the ten tape units. The proper tape handling procedures are described in Section 5.4.2. Even though very rigid procedures are followed for labeling, storing, and handling tapes, human mistakes are bound to happen from time to time. Also, the magnetic properties can deteriorate or dust particles can accumulate so as to make a tape unreadable. Tape unit malfunctions can also physically destroy a tape, sometimes quite spectacularly. For these reasons, users should maintain backup copies of important files, either on another tape or on some other medium.

The next most convenient backup medium is probably punched cards. With the COPY command, a user can cause the contents of a disk file to be punched into cards. This, of course, occurs in the machine room, not at the remote terminal, and requires the user to retrieve them from the I/O counter. The

retrieval process is described in Section 5.4.4.1. When it becomes necessary to create a disk file from a punched card deck, the user must follow the procedures described in Section 5.4.1.6, and submit the deck at the I/O counter. This same procedure is quite useful for preparing large data files on punched cards and then loading them on disk (or tape) for remote processing. Punched card decks are a very economical, permanent, and private means of file storage provided the decks are tightly stacked in a cool, dry place (without rubber bands).

Line printer listings and Calcomp plots are inconvenient as backup media, but are almost indispensable for displaying large quantities of data in tabular or graphic form. Source language program listings may be generated in the machine room via the COPY command, or programmatic output may be directed to a line printer file. These details are discussed in Section 5.4.4.2. The procedures for creating a Calcomp plot are fully described in a separate manual available at the Bookstore entitled "B5500 Calcomp Plotter Manual." The disadvantage of having to retrieve printer and plotter output from the I/O counter is somewhat balanced by the saving in remote terminal printtime and the labor involved in manual plotting. Each user must judge accordingly.

The operational procedures for using the different storage media are described in Section 5.4, along with details for declaring and using the corresponding programmatic files. The above comparison of the different media is intended only as an overview of certain advantages and disadvantages. Section 5.4 and possibly the appropriate language manual should be consulted for details.

5.4. Procedures for Using Different Storage Media

5.4.1. Disk Files

5.4.1.1. General

All disk files with an associated remote user-code are available for use only during the authorized hours for remote operations. At the end of each period of remote operation, all such disk files are dumped to magnetic tape and removed from disk by RECC personnel. Prior to the beginning of the next period of remote operations, these files are reloaded to disk. This is done to make more disk space available for batch operations and in an attempt to insure the integrity of the remote users' disk files.

5.4.1.2. Weekly Disk Usage Report

Once each week, a disk usage report is produced and forwarded to each Departmental Computer Coordinator, if applicable. It shows disk usage by user-name and gives summary totals for the School, Department, or Division. Each Coordinator should examine this report and note the quantity, size, and information-packing density of his users' files. He should then make this listing available to each of his users, along with any necessary recommendations about reducing the size or improving the blocking or packing densities of his users' files. Each user is urged to consult these listings regularly and make every effort to remove unneeded files and improve the packing density of those that are needed.

5.4.1.3. ACTIVE and EXPIRED Files

Each file shown in the weekly disk usage report is classified as being either ACTIVE or EXPIRED. ACTIVE files are those that have been accessed within the last week; all others are EXPIRED.

All EXPIRED files are removed from disk and dumped to a tape labeled EXPdddc, where ddd is the Julian date on which the tape was created, and where c is 0, 1, 2,..., for uniqueness. The expired tape label is shown for each EXPIRED file on the weekly usage report. A tape containing EXPIRED files is normally purged four weeks after its creation.

If it should become necessary or desirable for a user to retrieve an EXPIRED file, he should first ask the operator to mount the tape with a message of the form (see Section 5.4.2)

```
TO SPO PLEASE MOUNT EXPIRED TAPE EXPddc←
```

After the tape is mounted as determined by a message from the operator or from CALL OL, the user should then enter

```
CALL CONTROL←
```

and answer NO to the question HELP REQUIRED (YES OR NO)? The user should then enter

```
ADD FROM EXPdddc file-1/user-code,←
    file-2/user-code, ..., file-n/user-code;END.←
```

where EXPdddc represents the expired tape label, and file-1, file-2, ..., file-n represent one or more file-names thought to be on the mounted tape. If more than one expired tape is required, the above procedure must be repeated.

5.4.1.4. Disk File Naming Conventions

As explained in Section 2.11, all disk files created by a remote user are identified by the creator's user-code (for file security purposes), by a file identifier (for referencing a specific file), and by a file-type (for distinguishing data files and specific source language files). The file identifier is of the form

prefix/suffix

As explained in Section 2.11, the user supplies a file-name for the prefix, and CANDE automatically supplies his user-code as the suffix. The prefix will normally be of the form

XXXXXXb	for source language files, or
OXXXXXX	for object files,

where b represents one space, O is the zero character, and XXXXXX represents the user-supplied file-name. This file-name must be an identifier of one through six characters, left justified in a field of blanks. The first character of the identifier must be a letter; the five or fewer following characters must be either letters or digits.

As explained later, disk files may be created by means other than CANDE commands. In such cases, the creator is responsible for choosing a compatible prefix as described above. The creator is also responsible for choosing the suffix to be, in all but the rarest cases, his own user-code. The user-codes assigned to individual remote users are of the form RaaXXXX, where aa is a two-letter department abbreviation (see Section 5.4.4.1), and XXXX represents four randomly-chosen letters or digits. One non-human, but valid, user of the system is CANDE itself, whose "user-code" is TSHARER. A user is not permitted to create a disk file whose suffix is TSHARER. A remote program will be terminated with the following error message:

-BAD FILNAME prefix suffix, NEAR LINE line

if it attempts to create a disk file whose prefix begins with a digit, or whose suffix is TSHARER, or whose suffix is not the creator's user-code, but is of the form RaaXXXX.

There is a single exception to the above rules which exists only to permit batch programs to be run under the Time Sharing System without any source language changes. If a remote program attempts to reference or create a disk file whose identifier is of the form

0000000/yyyyyyX

the MCP will automatically change the external identifier of the file to be of the form

yyyyyyb/user-code

In other words, if the prefix is seven zeroes and the suffix is seven or fewer characters beginning with a letter (a valid prefix/suffix for a batch program disk file), the MCP will automatically use the first six characters of the old suffix as the new prefix, and will automatically attach your user-code as the new suffix. Note that the internal file identifier contained in in the source language is not affected--only the external name is automatically changed. Therefore, each time such a program is run, the MCP will automatically make the necessary change.

If a program executed from a remote terminal attempts to reference a disk file which is not present, it will be discontinued with the following message:

-NO FILE ON DISK prefix suffix, NEAR LINE line

If a program attempts to enter a disk file into the directory with the same prefix/suffix as a file which already exists, two cases are possible. If the existing file and new file were created under the same user-code, the old file will be removed and replaced by the new file. If that is not the case, the program will be terminated with the following message:

-FILE prefix suffix IN USE, NEAR LINE line

5.4.1.5. Disk File Blocking and Other Conventions

There are certain blocking parameters associated with each disk file (in fact, with every file) that describe its logical and physical structure. These parameters are used by both the MCP and the program referencing the file to accomplish a steady, buffered, transfer of information between disk

and core memory. Programs read and write information in units called logical records. The size of a logical record is measured in computer words; e.g., a 10-word logical record. The actual transfer of information between disk and core memory is accomplished in units called physical records. The size of a physical record is also measured in computer words. Each physical record must contain an integral number of logical records; hence, the physical record size must be an integral multiple of the logical record size. Certain electro-mechanical characteristics of the disk mechanism impose an additional requirement that the physical record size be an integral multiple of 30 words. Other blocking parameters specify the maximum number of equal-sized areas that the file may ever occupy and the size of each area, measured as a number of logical records. Since areas are not allocated unless they are actually needed, disk space can be conserved by specifying a small area size; however, the area size must contain an integral number of physical records. The user should consult the appropriate programming language manual for further details.

Most users create disk files with CANDE commands, such as CREATE and SAVE. The blocking parameters of such files are automatically specified by CANDE, and are of little concern to the user if he references these files with CANDE commands only. When some program must create or reference a disk file, knowledge of the blocking parameters becomes more important. Blocking parameters are always specified at the time a disk file is created, and compatible parameters must be specified in any program that references an existing disk file. The parameters that CANDE specifies when it creates a file for a user are fixed and are not under the user's control. Similarly, when CANDE references an existing file it expects it to have the same fixed parameters. Therefore, it is the user's responsibility to guarantee that his program specifies correct disk file blocking parameters whenever a programmatically created file is to be referenced by CANDE or whenever a CANDE-created file is to be referenced programmatically.

An integer save-factor is another parameter associated with every disk file. It was originally intended to specify the number of days the file was to be retained on disk; however, this is not the interpretation at Georgia Tech. A save-factor of zero identifies a temporary disk file that may be

removed without warning at any point in time. Any positive save-factor, such as 1, 5, 365, etc., identifies a permanent disk file, which will be reloaded provided it has been accessed within the last week, regardless of the integer value.

CANDE specifies, and expects, all source language and type DATA disk files to be blocked with 10-word logical records and 300-word physical records. Since each computer word can hold 8 characters, the 10-word logical record is sufficient to contain 80 characters (one Teletype line or one card image). Each 300-word physical record can contain 30 logical records.

If a programmatically created disk file is blocked with 10-word logical records and 300-word physical records, its file-type will be DATA; otherwise, its file-type will be UNKNOWN.

Users writing ALGOL or GTL programs should consult the appropriate Programmers Reference Manual for details concerning disk specifications.

An existing CANDE-compatible disk file should be referenced through a FILE declaration of the form

```
FILE F1 DISK SERIAL "file-name" (2,10,300,SAVE factor)
```

where integer represents one twentieth of the maximum number of logical records in the file and must be a multiple of 3, and factor represents the save-factor of the file.

The construct LOCK(fileid,*) may be used on newly created disk files to cause them to be entered in the MCP disk directory and to return as much unused space as possible. In most cases this is the easiest method for minimizing the space actually occupied, though it is not necessarily optimum.

Users writing BASIC programs should declare new disk files with a statement of the form

```
FILES F1(integer 1), ..., Fn(integer n)
```

where F1, F2, etc. are the file-names and the integers in parentheses designate the maximum number of logical records in each file. The integers must be integral multiples of 60. All other blocking parameters are automatically chosen to be compatible with CANDE.

Users writing COBOL programs should consult the B5500 COBOL Reference Manual for details concerning disk file specifications. The following portions of disk file descriptions may be used for CANDE-compatible disk files:

```
SELECT F1 ASSIGN TO DISK.
.
.
.
APPLY TECHNIQUE-A ON F1.
.
.
.
FD F1
   FILE CONTAINS 20* integer RECORDS
   ACCESS MODE SEQUENTIAL
   BLOCK CONTAINS 30 RECORDS
   RECORD CONTAINS 80 CHARACTERS
   VALUE OF IDENTIFICATION "file-name"
   SAVE-FACTOR factor
.
.
.
```

for new disk files:

```
OPEN OUTPUT F1.
.
.
.
CLOSE F1 WITH CRUNCH
```

for existing disk files:

```
OPEN INPUT F1.
.
.
.
CLOSE F1.
```

where integer represents one-twentieth of the maximum number of logical records in the file and must be a multiple of 3, and factor represents the save-factor of the file. The construct CLOSE fileid WITH CRUNCH may be used to return unused disk space in a newly-created disk file. Note that the internal file identifier, here represented by F1, should start with a letter and contain no hyphens.

Users writing CODASYL programs should consult the B5700 COBOL Reference Manual for details concerning disk file specifications. The following portions of disk file descriptions may be used for CANDE-compatible disk files:

```
SELECT F1 ASSIGN TO 20 * integer DISK
     ACCESS MODE SEQUENTIAL
.
.
.
FD   F1
     BLOCK CONTAINS 30 RECORDS
     RECORD CONTAINS 80 CHARACTERS
     VALUE OF IDENTIFICATION "file-name"
     SAVE-FACTOR factor
.
.
.
```

for new disk files:

```
OPEN OUTPUT F1
.
.
.
CLOSE F1 WITH CRUNCH
```

for existing disk files:

```
OPEN INPUT F1
.
.
.
CLOSE F1.
```

where the parameters and other comments are the same as for COBOL.

Users writing FORTRAN programs referencing disk files must use FILE card images, as explained in Appendix B of the B5700 FORTRAN Compiler Reference Manual. FILE card images must appear before any other source language statements. An existing CANDE-compatible disk file should be referenced using a FILE card image of the form

```
FILE  u = file-name, UNIT = DISK,BLOCKING = 30, RECORD = 10,
```

where u represents the file's unit number. To be CANDE-compatible a new disk file should be declared with a FILE card image of the form

```
FILE  u=file-name,UNIT=DISK,BLOCKING=30,RECORD=10,
    - SAVE=factor,LOCK,AREA=integer
```

where factor designates the file's save-factor and integer designates the maximum number of logical records in the file; integer must be an integral

multiple of 60. In both forms above, RECORD=10 was chosen to designate a 10-word logical record and BLOCKING=30 was chosen to designate a 300-word physical record.

5.4.1.6. Transferring Files Between Batch and Remote Modes

Sometimes it is desirable to have remote terminal access to files that were created in batch mode, or, conversely, to have batch mode access to files created in remote mode. The user can accomplish such transfers in either direction by dumping the desired files to magnetic tape in one access mode, either batch or remote, and subsequently loading them to disk from tape in the opposite mode. The proper file security status is important for transfer in either direction, and CANDE compatibility is important for transfer from batch to remote mode.

At Georgia Tech, each user-code and password pair is valid for only one mode of access, whichever was requested. If a user requires both access modes, two separate requests must be made and two pairs of user-codes and passwords are issued. A user with dual mode access credentials appears to the operating systems to be two different users. Hence, files created with one set of credentials cannot be accessed with the other set, unless the proper file security status has been assigned.

To be compatible with CANDE, both the file naming conventions (see Section 5.4.1.4) and file blocking conventions (see Section 5.4.1.5) must be followed. CANDE compatibility is actually required only when the file is to be manipulated with CANDE commands, but is desirable for all files accessed in remote mode. In addition to the naming and blocking conventions, there is one further restriction on the transfer of batch mode files to remote mode. The object version of a program compiled in batch mode can not be executed from a remote terminal--the source language file must be transferred from batch to remote mode and then recompiled under the Time Sharing System.

The steps for transferring files in either direction require that the procedures for handling magnetic tape files be carefully followed (see Section 5.4.2).

Transferring Files to Remote Mode

Step 1. In batch mode, assure that the desired disk files are properly named and blocked, and that the security status of each file is FREE. Then follow the batch mode tape handling procedures to dump the files to tape.

Step 2. In remote mode, follow the procedures described in Section 5.4.2 to have the operator mount the tape. Then enter

```
CALL CONTROL←
```

and to the question

```
HELP REQUIRED (YES OR NO)?
```

enter

```
NO←
```

then enter

```
ADD FROM tapelabel file1/user-code, ..., filen/user-code; END.←
```

Step 3. In remote mode, and after the files have finished loading, use the CANDE commands LOCK, UNLOCK, or PUBLIC to assign the desired security status to each file. If this is not done, the disk files will remain FREE and will be subject to removal at any time.

Transferring Remote Files to Batch Mode

Step 1. In remote mode, follow the procedures described in Section 5.4.2 to have a scratch tape mounted, then enter

```
CALL CONTROL←
```

and to the question

```
HELP REQUIRED (YES OR NO)?
```

enter

```
NO←
```

then enter

```
FREE file1/user-code, ..., filen/user-code;←
DUMP TO tapelabel file1/user-code, ..., filen/user-code; END.←
```

Step 2. In remote mode, and after the operator has informed you of the tape reel number, return the disk files to the desired security status with the LOCK, UNLOCK, or PUBLIC CANDE commands. If this is not done, the disk files will remain FREE and will be subject to removal at any time.

Step 3. In batch mode, follow the batch mode tape handling procedures to load the files from tape to disk.

Creating Disk Files from Punched Card Decks

The procedures described above for transferring files between batch and remote modes require the same person to be both a batch user and a remote user. There are many people who are remote users only, and who may desire to create remote disk files from punched card decks. Such punched card decks may have been produced by the procedures described in Section 5.4.4.3, or they may have been originally prepared on keypunch machines. A remote user may create a LOCKed CANDE compatible disk file from a punched card deck by following the steps of a special procedure given below.

Step 1. Prepare a punched card deck as follows:

```
?USER = user-code.  (to be stamped "REMOTE")
?EXECUTE CARD/DISK.
?FILE DISK = file-name/user-code SERIAL.
?DATA CARD.
      .
      .        (punched cards to be loaded to disk)
      .
?END.  (special orange colored, prepunched card)
```

Step 2. Submit the above card deck to a Programmer Aide. If he approves the deck setup, he will stamp the top card "REMOTE" in large letters, and return the deck to the user.

Step 3. Fill out an I/O bin receipt and submit both the receipt and stamped deck to the I/O attendant, calling his attention to the "REMOTE" stamp. The attendant will place the deck in a special tray. Decks submitted between the time remote operations have begun and one hour prior to the end of remote operations will usually be processed within about 30 minutes. Decks submitted

outside the above times will be held and processed during the first 30 minutes of the next scheduled period of remote operations. Your card deck may be retrieved from the numbered bin stated on your bin receipt, not your permanent department bin.

5.4.2. Magnetic Tape Files

5.4.2.1. General Procedures

For tapes to be used as input to programs, the operator needs the reel numbers and labels of the tapes, and the user's name. For tapes to be created by programs, he needs the number of tapes required, the user's name, reference number, and labels of tapes to be saved, in order to be able to catalog them in the tape library. The remote user must give the operator a reasonable amount of time to obtain and mount input tapes and to ensure that enough scratch tapes are available before running his program. If the program is run too soon, the terminal may appear to be hung, as the system will not answer inputs from the terminal while the program is waiting for tape units for input or output. Thus, at least two minutes before tape activity is required, the user should send a message to the operator. The suggested form for each input tape is the following:

```
TO SPO I NEED TAPE #12345 LABELED SPCTRM ... BURDELL GP←
#
```

The suggested form for output tapes is as follows:

```
TO SPO I NEED 3 SCRATCH TAPES FOR SORT←
#
TO SPO AND TWO TAPES FOR OUTPUT←
#
TO SPO THEY WILL BE LABELED T1A AND T1B←
#
TO SPO #15C12501 BURDELL GP←
```

The user should promptly receive a message from the operator when his tapes are mounted and ready; only then should he proceed to run his program.

If, after a reasonable time following program termination, the operator has not sent the reel numbers of the tapes the program has created, the user should prompt him.

Prior to creating a magnetic tape, each user should consult his Departmental Computer Coordinator about current procedures governing tape storage charges.

5.4.2.2. How to Determine Tapes Currently Mounted

At any point in time that the user is not running a program, he may receive a listing of information concerning all tapes currently mounted and ready for use by entering

CALL OL←

By this means a user may determine if his tapes are actually mounted and ready for use.

5.4.2.3. How to Purge a Saved Tape

Once saved by a user, a tape may be purged only on the signature of the creator, or his representative, on the proper forms.

The purge request forms may be obtained in the Computer Center and submitted there, or through the U. S. Postal Service or Campus Mail, to the following address:

GEORGIA TECH
RECC
OPERATIONS BRANCH
ATLANTA, GEORGIA 30302

ATTN: TAPE LIBRARIAN, B5500

5.4.2.4. How to Copy Disk Files to Tape

When a user decides that his disk files are in a condition suitable for somewhat secure, permanent retention, he may SAVE his current work-file (if any), request and get permission to create and save a tape (labeled by his user-code) from the operator, then enter the command

COPY TO TAPE←

This will produce a tape labeled with the user's user-code that contains all disk files created by the user. The operator should send the reel number to the user, as described above.

5.4.2.5 How to Reload Tape Files to Disk

When a user desires to reload one or more files that he previously dumped to tape by the above procedure, he should first SAVE his current work-file. He should then request the operator to mount his tape, giving him the reel number, his user-code, his Reference Number, and his user-name. The user should then enter

CALL CONTROL←

and to the question

HELP REQUIRED (YES OR NO)?

he should answer

NO←

To reload selected files not currently on disk, he should then enter

ADD FROM user-code file1/user-code, ..., file/user-code; END.←

Or, to reload all files on that tape not currently loaded, he should enter

ADD FROM user-code =/=; END.←

When a user desires to reload one or more EXPIRED files (see Section 5.4.1.3) he must first consult the weekly Disk File Usage Report to determine the label of the tape containing his files (see Section 5.4.1.2). The user should then request the operator to mount the desired expired tape with a message of the form

TO SPO PLEASE MOUNT EXPdddc←

After the tape is mounted (as determined by a message from the operator or from CALL OL), the user should then enter

CALL CONTROL←

and to the question

HELP REQUIRED (YES OR NO)?

answer

NO←

and then enter

ADD FROM EXPdddc file1/user-code, ..., filen/user-code; END.←

5.4.3. Remote Terminal Files

All attributes of REMOTE files are ignored except for record length. There is no major advantage nor disadvantage in opening more than one REMOTE file per program, except in COBOL and CODASYL, in which a REMOTE file may not be opened I-O. Every READ from the terminal first types a question mark on the terminal, unless the READ STOP variant is used, the user is entering data faster than the program is accepting it, or the terminal is in TAPE mode.

5.4.3.1. How to Declare REMOTE Files

The following describes the manner in which REMOTE files are declared and used in various source languages:

ALGOL and GTL

```
FILE F REMOTE (1,9)
FILE F WITH *,*,*,*,*,19
F.TYPE := 19
READ(F, ...) or READ(F[STOP], ...)
WRITE(F,...) or WRITE(F[STOP] ...) or WRITE(F[NO], ...) or
                WRITE(F[n], ...)
```

COBOL and CODASYL

```
SELECT F-IN ASSIGN TO REMOTE
SELECT F-OUT ASSIGN TO REMOTE

OPEN INPUT F-IN OUTPUT F-OUT
READ F-IN [INTO data-name][BEFORE ADVANCING data-name LINES]

WRITE F-OUT-REC[FROM data-name][BEFORE STOP] or
WRITE F-OUT-REC[FROM data-name][BEFORE ADVANCING data-name LINES]
```

FORTRAN

```
READ with no unit and
PRINT compiled from terminal automatically reference terminal, otherwise
FILE  u=R,UNIT=REMOTE,RECORD=9
```

is required for each unit u which is to reference terminal; carriage control on output is printed as first character of record.

5.4.3.2. How to Use Free-Field Input and Output

The following describes the manner in which free-field I/O may be used in various source languages:

ALGOL and GTL

```
READ(F,//,L)[:ERR]
WRITE(F,//,L)[EOF]
```

where F is a REMOTE file,
L is an explicit or implicit list of variables,
EOF is break label,
ERR is input data error label,
data separator is space or end of line.

COBOL and CODASYL

(no direct method)

FORTRAN

```
READ/,L
WRITE//,L
READ(u,/,ERR=n) L
WRITE(u,//,EOF=n) L
```

where u is the REMOTE file unit number,
L is a list of variables
m is the break label,
n is the input data error label,
data separator is comma or end of line,
input must match list types

5.4.4. PRINTER, CARD PUNCH, and Calcomp Plotter Files

5.4.4.1. Retrieval of Output

A special I/O bin is maintained for each campus School, Department, or Division that has remote terminal users. Each bin is labeled with a two-letter department abbreviation, the same as the second and third letters of your individual user-code. During remote operations, all punched card decks, line printer listings, and Calcomp plots are placed in these permanent bins as determined by the user-code of the creator. Each departmental com-

puter coordinator (when necessary) is issued two permanent, plastic encapsulated bin receipts for his bin. When a permanent bin receipt is presented at the I/O counter, the bearer will be given the entire contents of his department's bin--not just a portion of it. The subsequent distribution of the contents is the responsibility of the hearer and his departmental coordinator.

5.4.4.2. Line Printer Files

Line printer output may be generated either via the COPY TO PRINTER command or directly from a user's program. Both methods automatically label the output properly so that it will be placed in the correct departmental I/O bin. However, it is suggested that the user's name appear in the printer file label, if possible, to facilitate subsequent distribution.

No advance permission or communication with the operator is normally required prior to creating printer output. However, it is good practice to send him a message afterwards saying that you have generated some output, and giving him your name and user-code. This will assist the operator in case he is having mechanical trouble with the printer. Each user should consult his departmental Computer Coordinator about procedures governing excessive output.

In case of special forms required on the printer, the EQUATE command may be used to great advantage. Specifically, the user must inform the operator that special forms are to be used, giving the names of the files, carriage tapes, special paper, and other special instructions. Then he must execute his program with the proper files being produced on printer backup tapes with FORM option specified. In ALGOL and GTL, this may be done internally in the program by specifying an output media digit of 38 on the proper files. In other languages, the EQUATE command must be used.

Examples:

```
EQUATE PRINT1 = PRINT1 BACKUP TAPE FORM←
EQUATE PRINT2 = PRINT2 BACKUP TAPE FORM←
EXECUTE PROG32 WITH IO = 15←
.
.
.
```

5.4.4.3. Card Punch Files

Punched cards may be generated either via the COPY to PUNCH command or directly from a user's program. The COPY TO PUNCH command is the preferred method and should be used in almost all cases. It automatically interacts with the operator and labels the output properly. Program output should be directed to a disk file and then punched with the COPY TO PUNCH command, rather than going directly to the punch.

If the COPY TO PUNCH command is used, no advance permission from the operator is required. However, it is good practice to send him a message afterwards saying that you have generated some output, and giving him your name and user-code. This will assist the operator in case he is having mechanical trouble with the card punch. Each user should consult his Departmental Computer Coordinator about current procedures governing excessive output.

Disk files may be created from punched card decks as described in Section 5.4.1.6.

5.4.4.4. Calcomp Plotter Files

Both the operational procedures and programming techniques for creating Calcomp plots are described in a separate RECC publication available at the Bookstore entitled "B5500 Calcomp Plotter Manual."

No communication with the operator is required either before or after creating a plotter file. All plotter output generated during remote operations is automatically labeled with the user's Reference Number, name, and permanent departmental bin.

Prior to generating a Calcomp plot, each user should consult his Departmental Computer Coordinator about current procedures governing plotter charges.

APPENDIX A

B5700 REMOTE CHARACTER SET

The following tables define the character set acceptable to the B5700 Time Sharing System. The left table is arranged in octal sequence, with 0 low and " high. The right table is arranged in collating sequence, with blank low and ? high. For both tables, the column on the extreme left gives the first octal digit and the row above the table gives the second octal digit.

	OCTAL 01234567
0	01234567
1	89#@?:>≥
2	+ABCDEFG
3	HI.[&(<←
4	×JKLMNOP
5	QR$*-);≤
6	/STUVWX
7	YZ,%≠=]"

COLLATING
.[(<←&$
*);≤-/,%
=]"#@:>≥
+ABCDEFG
HI×JKLMN
OPQR≠STU
VWXYZ012
3456789?

Several of the characters are interpreted by the system on input as control characters only. They are the following:

character	keyboard character	meaning
←	← or CARR RET	line terminator
≥	!	line delete
≤	'	backspace

In addition, the ? may be interpreted in some contexts as a control character when it appears in the first position of input (see SS, STATUS, STOP, TAPE, TO, ? commands).

The following characters will all print as a ? when transmitted by the

system to the remote terminal:

$$\leq, \geq, <, >, \neq$$

In addition, ← will print as a ? when transmitted to a remote terminal if nonblank characters follow it on the line being transmitted and will terminate the line if it is followed only with blanks.

APPENDIX B

CANDE RESERVED WORDS

The reserve words as recognized by CANDE are dependent upon a positional format. Hence, the following list of reserved words have been logically divided into the following classes:

CANDE Commands,
File-types,
Compiler-file-types,
Parameters.

Words which are followed by one or more words in parenthesis are all interchangeable alternatives. The use of CANDE reserved words as file-names is generally illegal; since many reserved words are one character in length, the use of one-character file-names is not recommended.

CANDE Commands

ADD (APPEND)	EQUATE	PUNCH	SET
APPEND (ADD)	EXECUTE (DO)(E)	PRINT(P)	SS (TO)
BYE	FILE (FILES)	R (RUN)	SSFILE
C (COMPILE)	FILES (FILE)	REMOVE	STATUS
CALL	FIND	RENAME	STOP
CC	FIX (*)	REP (REPLACE)	TAPE
CHANGE	GUARD	REPLACE (REP)	TIME
CHARGE	HELLO	RESEQ	TO (SS)
COMPILE (C)	L (LIST)	RESET	TYPE
COPY	LIST (L)	RMERGE	U (UPDATE)
CREATE (MAKE)	LOAD	RUN (R)	UNLOCK
DELETE	LOCK	S (SEQ)	UPDATE (U)
D (DISPLAY)	MAKE (CREATE)	SAVE	WHATS
DISPLAY (D)	MERGE	SCH(SCHEDULE)	
DO (E) (EXECUTE)	MONITOR	SCHEDULE (SCH)	
E (E) (EXECUTE)	P(PRINT)	SEQ (S)	

File-types

A (ALGOL)	D (DATA)	GRAMMAR(G)	W(WIPL)
ALGOL (A)	DATA (D)	GTL	WIPL (W)
APL	DYNAMO	I (INFO)	X (XAGOL)
B (BASIC)	E (ESPOL)	L(LOCK)	XALGOL (X)
BASIC (B)	ESPOL(E)	LOCK (L)	
C (COBOL)	F (FORTRAN)	S (SEQ)	
COBOL (C)	FORTRAN (F)	SEQ (S)	
CODASYL	G(GRAMMAR)	T (TSPOL)	

Compiler-file-types

A (ALGOL)	COBOL (C)	F (FORTRAN)	X (ALGOL)
ALGOL (A)	CODASYL	FORTRAN (F)	XALGOL (X)
B (BASIC)	DYNAMO	GTL	
BASIC (B)	E (ESPOL)	T (TSPOL)	
C (COBOL)	ESPOL (E)	TSPOL (T)	

Parameters

ABORT	FILE	NUMBERED (#)	SOLEUSER
ADAPTER	FIND	OBJECT	SOURCE
ADDRESS	FIRST	OPTIONS	SPO
AFTER	FROM	PASSWORD	SQUASHED (*)
ALL	GEQ	PRINT	STATION
ALLOWMSG	GTR	PRINTER	TAPE
ARROW	HELPFUL	PUBLIC	TELETYPE
AT	HEX	PUNCH	TEXT
BACKSPACE	INTERRUPT	QUICKBYE	TO (-)
BUSY	LIBRARY	QUICKLOG	TRANSMIT
CANDE	LITERAL	RECEIVE	TSHARER
CARDS	LEQ	REP	TYPE
CHANGES ($)	LOCKED	REPLACE	UNLOCKED
CONCISE	LONG	RESEQ	WITH
DELETE	LSS	SEQ	$ (CHANGES)
DISK	MONITOR	SEQUENCE	# (NUMBERED)
END	NAME	SHORT	* (SQUASHED)
EOR	NEQ	SITE	- (TO)
EQL	NOSTOP	SIZE	
FACTOR	NUMBER		

APPENDIX C

B5700 FILE SECURITY SYSTEM

The B5700 Time Sharing System uses the file security system developed for the B5700 Data Communications System. This system recognizes one privileged user-code which is allowed access to all disk files in the system. On the Time Sharing System, it is defined to be that of the installation running the computer.

All other user-codes are subject to the constraints of the file security system. For them, there are four levels of file security:

A locked file may be accessed only under the user-code under which it was created. This type of security is created by the use of the SAVE and LOCK commands. All disk files are initially locked files when created.

A private file may be accessed by the user-code and programs listed in the GUARD file associated with the private file. A private file can be created by the use of the GUARD and the LOCK commands.

An unlocked file can be read (or executed, if the file has an object version) under any user-code, but it can only be changed under the user-code by which it was created. Unlocked files are created by the use of the UNLOCK command.

A public file can be read or written (or executed), if the file has an object version under any user-code. Public files are created by the use of the PUBLIC command.

The security status of a file is treated in much the same way as the file-name or the file-type. In effect, the security status is loaded with the file and, if neither the file nor the security status is changed by the user, it remains with the file when the file is saved, even if the file-name has been changed.

A GUARD file can specify two levels of access for either programs or users:

-Read-only - the program or user-code may only read the file (or execute if object code).

-Read/write - the program or user-code may read from or write into the file (not allowed on object code).

The general forms of the commands used to change the security status of files are as follows:

(LOCK| UNLOCK| PUBLIC)[SOURCE|OBJECT][file-name]...]...

and

LOCK [file-name] [WITH guardfile-name]

If the SOURCE or OBJECT parameter is empty, both source and object versions are implied. If the file-name parameter is empty, the work-file is implied.

The many combinations of these verbs and parameters are summarized in the following table:

command	degree of access for another user-code to		
	source file		object file
	READ	WRITE	EXECUTE
PUBLIC	YES	YES	YES
PUBLIC SOURCE	YES	YES	unchanged
PUBLIC OBJECT	unchanged	unchanged	YES
UNLOCK	YES	NO	YES
UNLOCK SOURCE	YES	NO	unchanged
UNLOCK OBJECT	unchanged	unchanged	YES
LOCK	NO	NO	NO
LOCK SOURCE	NO	NO	unchanged
LOCK OBJECT	unchanged	unchanged	NO
LOCK...WITH	GUARD file	GUARD file	GUARD file

APPENDIX D

CANDE MESSAGES

The following is a list of CANDE messages and a description of the meaning of each message.

Message	Meaning
A RUN OR EXECUTE OR CALL OR EQUATE MUST FOLLOW AN EQUATE COMMAND	User has entered one or more EQUATE commands, followed by a command illegal to follow EQUATE. Chain of EQUATE is forgotten.
ARRGH	User has attempted to CHANGE the file-type of a file belonging to another user.
B5700 TIME SHARING SYSTEM-LINE line-number	User has completed connection to B5700. He will be asked to log in.
B5700 TIME SHARING SYSTEM MARK level	User has entered WRU, ?, and ←. Response identifies system and gives level of operating system.
BASIC	User has attempted to RESEQ only a portion of a BASIC program.
BADCODE	A user-code or password which cannot be verified has been entered when user attempts to log-in on the system.
BADTYPE	A command is being used giving an invalid file-type.
BUSY	A message is being sent to a station which is running a program but the receiving user has not set the ALLOWMSG option.
BYE	A station is being logged-out of the system.
CC	User has attempted to change the carriage length of a device without a carriage.

Message	Meaning
CANNOT COMPILE THIS TYPE OF FILE--PLEASE INCLUDE FILE-TYPE NEXT TIME.	File-type of work-file is not a compiler-file-type; see Appendix B.
CHANGE IGNORED--FILE MAY BE IN USE BY ANOTHER USER.	A CHANGE command has been ignored since the file was not in a state capable of being changed.
CHANGE REQUIRES AT LEAST TWO PARAMETERS.	A parameter to CHANGE command is illegal, misspelled, or missing.
CHARGE	CHARGE command entered illegally.
COMMAND NOT YET IMPLEMENTED	User has entered an option of a command which has not been implemented yet.
COMPILING	User's program has begun to compile.
CONTROL RECORD ERROR OCCURRED ON ZIP COMMAND.	A programmatic ZIP contains an error.
COPY OR RE-LOAD REQUIRED TO CHANGE DATA FILE-TYPE	User has attempted to change the file-type of a type DATA work-file. This is illegal, because DATA files are referenced by record-number and other type files are referenced by sequence-number.
COPYING	COPY TO TAPE operation has begun.
DID NOT COMPILE...CHECK SYNTAX AND RETRY, PLEASE.	Compilation was incorrect because of syntax errors.
DISK SPACE FOUND. YOU ARE RUNNING AGAIN.	Notification that disk space has been found for a file which was in a no-user-disk situation. Processing is now continuing.
DONE.	Response to STATUS or STOP command.
ENTER USER CODE, PLEASE.	Response to a HELLO command or on initial call-up of the system.
ENTER YOUR PASSWORD	Response to a HELLO command or on initial call-up of the system.
ERR:	This is the prefix to the short error messages which are given by CANDE. Further explanation may be elicited by entering ?'s.

Message	Meaning
ERROR IN FIND OR REPLACE STATEMENT.	Improper parameters or delimiters appear in a FIND or REPLACE command. Refer to syntax of the command.
ESP DISK TABLE IS CURRENTLY FULL. PLEASE TRY AGAIN LATER.	An EQUATE command was entered, but not enough scratch-pad disk storage was available to store it, and it was ignored.
FILE NOT IN YOUR LIBRARY.	A file-name was specified which could not be found.
FILE SPECIFIED IS NOT A SCHEDULE OUTPUT FILE.	Refer to a specification parts for the SCHEDULE, STATUS,STOP command syntax. The SCHEDULE output file may be created with a SCHEDULE command interrogated with a STATUS command, and its line may be terminated with a STOP command.
FILENAM	File-name was expected but was not found in command.
FILE-NAMES CAN HAVE AT MOST SIX CHARACTERS.	An illegal file-name was specified.
FILTYPE	Illegal file-type specified in LIST FILES command.
FIX SYNTAX ERR @ sequence number	Illegal FIX command entered.
FLAG BIT: FILE file-name at DISK ADDRESS address-FILE DISCARDED.	Notification of trouble with work-file. CANDE will discard file and allow you to proceed. You should check the file you were processing to determine the extent of the damage.
FOUND MORE THAN ONE FILE-TYPE IN THE COMMAND.	Command specified two file-types such as the following: CHANGE ALGOL TO GTL.
FOUND "SIZE" MORE THAN ONCE IN YOUR INPUT.	The reserved word SIZE was used more than once in the last command.
FROM	Illegal user-code designation in LIST FILES command.

Message	Meaning
FUNNY. .	The user has attempted to bypass system security by not entering a user-code or password when the system asked for his credentials.
GUARD FILE MUST BE OF THE TYPE "LOCK".	User has attempted to LOCK a file or update a GUARD file which was not the output of the GUARD command.
I CANNOT ALTER THAT FILE- IT IS YOUR WORK-FILE	A CHANGE file-name... command has been used where a RENAME or TYPE command should have been used.
I CANNOT REMOVE THAT FILE -- IT IS YOUR WORK-FILE.	A REMOVE file-name command has been used where a REMOVE command should have been used, if that is what is desired.
I CANNOT FIND THAT FILE IN YOUR LIBRARY	The file-name indicated in the last CANDE command is not present on disk.
IGNORED	Notification that a command has been ignored due to the current disposition of a file or the user's terminal.
ILLEGAL PARAMETER IN SAVE.	Refer to the SAVE command syntax.
ILLEGAL PARAMETER IN REMOVE.	Refer to the REMOVE command syntax.
IMPROPER "FILE" SPECIFIER.	Illegal FILE option syntax in FIND or REPLACE command.
INCOMPLETE FIND OR REPLACE STATEMENT (COMMA OR END).	Check delimiters in the FIND or REPLACE command.
INCOMPL	Additional parameters or specification parts are required for this command.
INCORRECT COMMAND-PLEASE REFER TO THE B5700 TERMINAL USERS GUIDE.	A CANDE reserved word has been used improperly.
INPUT MUST START WITH A VERB OR SEQUENCE-NUMBER.	Last input was not a CANDE command or data for the work-file.

Message	Meaning
INPUT TOO LONG...RE-ENTER PLEASE.	The last input is physically too long to process; CANDE commands must be restricted to 200 characters.
INPUT RESTORED THRU....	Notification pertaining to the validity of the user's work-file following a system failure.
INSTRUCTION NOT RECOGNIZED.	Command does not meet syntax specifications. Refer to syntax of command.
LITERAL	Illegal LITERAL designation in LIST FILES command.
LOADING	Notification that the LOAD command has been accepted and initiated.
MERGE	MERGE attempted on type DATA file, usually.
MISSING DELIMITER OR STRING TOO LONG.	Strings in CANDE commands are restricted to 63 characters in length. Either this limit has been exceeded, or right delimiter of string has not been included in last command.
MISSING DELIMITER OR INCORRECT INSTRUCTION.	User has entered illegal parameters to a CANDE command.
MON.FIL	Missing MONITOR file.
NAME	Command requires a file-name as parameter.
NO CHECKPOINT FROM WHICH TO RESTART--PLEASE START FROM SCRATCH.	Notification pertaining to the invalidity of the user's previous work-file following a system failure.
NO CODE	Response to a DO, E, EXECUTE, R, or RUN command when the specified object file is not on disk, and no source file could be found from which to generate it.
NO ERRORS TO REPORT.	Response to a ? command with no errors resulting from the last operation.

Message	Meaning
NO FILE	A file-name has not been specified in the command or has not been found in the user's library.
NO NAME	A file-name has not been specified in the command.
NO OBJ.	A valid object file was not found, as required to process the last command.
NO ROOM	Not enough area for this file on disk or to complete the command.
NO SAVE	Work-file needs to be SAVEd or REMOVEd before that command may be processed.
NO WORK-FILE - USE MAKE OR LOAD.	A command was entered which requires a work-file, but the user had not yet declared one.
NO USER	No user was found with the specified user-code, or slash was not followed with a user-code.
NOFILE	A file could not be located with the specified file-name.
NOPARAM	A parameter which must be specified for this command has not been entered.
NOPRGRM	A ? TAPE command was entered, but the terminal was not attached to a program.
NOT ON	Given station has logged-off.
NOTDONE	Current command is still being processed; please wait a while longer.
NOT ENOUGH ROOM FOR YOUR FIX IN RECORD sequence-number	Fix command caused nonblank information to be truncated on record sequence-number.
OK.	Terminal is now in TAPE mode.
ONE OF YOUR PARAMETERS IS ILLEGAL.	A parameter to a command the user has entered is illegal.

Message	Meaning
OBJECT PROGRAM FILE IS NOT ON DISK.	A valid object version was not found by the specified file-name.
ONE OF YOUR PARAMETERS IS OUT OF SEQUENCE	User has entered a valid command with valid parameters, but they are out of order. Refer to syntax of command.
ONLY ONE FILE-NAME ALLOWED.	The last command may not be made compound, such as in the following: COPY AA,BB,CC TO PRINTER.
PARAMETERS MUST OCCUR IN PAIRS.	Check the parameter specifications for the last command.
PATIENCE--YOUR LAST REQUEST IS TAKING LONGER THAN I EXPECTED.	This normally indicates that the system is under heavy use and the request is being processed more slowly than normal. It may also indicate that the command has not been executed because a file is not currently available for use - so CANDE is waiting.
PLEASE CALL BACK AT YOUR SCHEDULED TIME.	Your user-code is restricted as to the time of day you may use the system.
P L O P	The system has failed and has been restarted. The user will be requested to re-enter his credentials.
P*L*O*P	A system failure occurred during the processing of the schedule time. The user should further check the output file and restart the line, if necessary.
PLEASE WAIT -NO USER DISK	The system disk tank requires expansion and no disk is available for such an action - wait for OK to proceed.
PROCEED	Continue entering commands and data.
RESEQ	RESEQ attempted on type DATA file, usually.

Message	Meaning
RESERVED WORDS MAY NOT BE USED AS FILE-NAMES	Refer to appendix B for a list of CANDE reserved words. They cannot be used as file-names.
RUNNING.	Response to indicate object program has started.
RUNNING(n).	Response to STATUS or STOP command.
SAVE UNNECESSARY - NO CHANGES SINCE LAST SAVE OR LOAD	File on disk corresponds to workfile - no SAVE required.
SAVED.	Response to a SAVE command in CONCISE mode.
SCHEDULED.	Response to indicate RUN/DO/EXECUTE/CALL/SCHEDULE command has been scheduled to run, rather than starting execution immediately.
SECURITY	Illegal security designation in LIST FILES command.
SRC-OBJ	Illegal security designation in LIST FILES command on object version of file.
SSFILE	User has attempted to enter on SSFILE command.
SEQUENCE-NUMBER TOO LONG.	A sequence-number was generated which was longer than eight digits.
SORRY, YOU ARE NOT SCHEDULED FOR TIME AT THIS HOUR.	Your user-code is restricted as to time of day you may use the system.
SORRY, BUT YOU ARE PAST YOUR SCHEDULED TIME AND WE MUST DISCONNECT YOU.	Your user-code is restricted as to time of day you may use the system.
SYSTEM OK - YOU MAY PROCEED.	System trouble (not enough disk) has been corrected and processing has continued.
TASK WAS DISCONTINUED.	Response to a STOP file-name command referencing a schedule output file, or last program executed from terminal terminated abnormally.

Message	Meaning
THAT COMMAND IS NOT COMPATIBLE WITH YOUR TERMINAL.	User has entered a command which is impossible to perform on his class of terminal device.
THAT COMMAND REQUIRES A FILE-NAME	User has entered a command which requires a file-name parameter, but none was given.
THAT CONSTRUCT CANNOT BE USED WITH TYPE DATA FILES	User has entered a command, such as RESEQ, or MERGE which is impossible to perform with files of file-type DATA.
THAT CONSTRUCT IS NOT AVAILABLE FOR YOUR USE AT THIS TIME.	User has entered a command which he is not capable of using.
THERE IS NO OBJECT CODE AVAILABLE... TRY RUN OR COMPILE.	A valid object version was not found by the specified file-name.
THE FILE IS NOT OBJECT CODE	File specified was not compiled on B5700 Time Sharing System, or at all, but is present.
THE PARAMETERS ARE IMPROPER-CHECK THE B5700 TERMINAL USERS GUIDE.	User has attempted a valid CANDE command with improper parameters.
THIS IS IMPROPER FOR SCHEDULE TASKS.	Illegal syntax for Schedule command, or for commands in a SCHEDULE line. Refer to syntax of SCHEDULE command.
TIME	In AFTER option of SCHEDULE command; illegal time was entered.
TIME MUST BE BETWEEN 800 & 2400. LAST TWO DIGITS MUST BE LESS THAN 60.	In AFTER option of SCHEDULE command, illegal time was entered.
TO	Illegal output designator in LIST FILES command.
TOOBIG.	A sequence-number or sequence range in a command is too large or illegal.
TOOLONG.	Last input to CANDE is physically too long to handle.
TOOMANY.	Only 9 parameters are allowed for the last command.

Message	Meaning
TOOMUCH	A command has been given to CANDE which contains too many elements.
user-code	In last command, user attempted to reference a file to which he was not allowed access.
"VERB" REQUIRES AT LEAST 1 PARAMETER.	User has entered a command without parameters, and parameters are required.
"VERB" CONTAINS TOO MANY PARAMETERS.	User has entered a command with more parameters than are allowed.
WE MUST DISCONNECT YOU.	User-code is limited according to the time of day - time is up.
WE HAVE TEMPORARILY RUN OUT OF DISK SPACE. PLEASE WAIT FOR OK.	Your command has requested more disk space than is currently available. Either enter WRU to discontinue the command or wait for OK.
WHAT?	A CANDE command has been given which may not be compound; correct and reenter the command.
WHAT.	Illegal parameter has been entered to TAPE command.
WORK-FILE HAS UNSAVED RECORDS IN IT - PLEASE SAVE OR REMOVE IT.	An attempt has been made to log-out, or LOAD another file without cleaning up the current work-file.
WORK-FILE DOES NOT HAVE FILE-TYPE DATA.	User has entered ?DATA← for work-file with file-type not DATA.
WORK-FILE OK.	Notification pertaining to the validity of the user's work-file following a system failure.
YOU ALREADY HAVE A FILE BY THAT NAME.	User has entered a command with a file-name as parameter which must not be already on disk.
YOU ARE SCHEDULED TO USE THE SYSTEM UNTIL time.	Your user-code is restricted as to the time of day you may use the system.
YOU ARE IN DATA MODE	?DATA has been entered, but ? ENDS has not yet been entered by the user.

Message	Meaning
YOU ATTEMPTED TO ACCESS A SECURED FILE.	The user has entered a command which attempted to access a file belonging to another user.
YOU HAVE ENTERED AN ILLEGAL FILE-TYPE.	A file-type must be one of those in the file-type table in Appendix B.
YOU HAVE PUT TOO MUCH ON ONE LINE. PLEASE SEPARATE.	Input message was too long for CANDE to handle, either in terms of number of characters or commands and parameters.
YOU MUST SPECIFY A FILE-NAME IN AN EQUATE COMMAND.	Refer to the syntax of the EQUATE command.
YOU MUST SPECIFY AN OUTPUT FILE FOR SCHEDULE.	Refer to the syntax of the SCHEDULE command.
YOU MAY USE THAT CONSTRUCT ONLY WITH THE RESET COMMAND.	User has entered a command which attempts to move records in a file to another position in that same file. This may be done only with the RESEQ command.
YOU MAY NOT SPECIFY A USERCODE ON THE OUTPUT FILE NAME.	Refer to the syntax of the FIND and REPLACE commands.
YOUR JOB NEEDS DISK THAT IS NOT AVAILABLE NOW. WAIT OR DS IT.	User's program is requesting more disk space than is currently available. Either enter WRU to discontinue the program or wait for OK.
YOUR NEXT SCHEDULED TIME IS FROM time to time.	Your user-code is restricted as to the time of day you may use the system.
YOUR PROGRAM IS BEING COMPILED.	User has entered a command which required that a program be compiled, and the compiler has not yet finished.
YOUR PROGRAM IS RUNNING.	User has entered a command which required that a program be run, and it is not yet finished.
YOUR REQUEST IS INCOMPLETE.	Additional parameters are required. for your last command.
ZIPERR:	Following line contains first 72 characters control record which was zipped by a program and contains an error.

APPENDIX E

I/O ERROR MESSAGES

Most error termination messages are self-explanatory. However, I/O error messages are identified only by an integer error number and have the following form:

-I/O ERROR integer file-name, NEAR LINE line

The table shown below lists each possible I/O error number (they are not consecutive integers) and its meaning:

I/O Error Number	Meaning
1	A program attempted to OPEN an input file that was not closed.
3	A program attempted to OPEN REVERSE a file that was not closed.
5	A program attempted to OPEN REVERSE a file that was not blocked properly.
6	A program attempted to OPEN an output file that was not closed.
11	An attempt was made to CLOSE an input file which was closed or never opened.
12	An attempt was made to CLOSE an output file which was closed or never opened.
15	An attempt was made to READ a file for which AT END has already been processed.
16	The record count on an input tape does not agree with the internally accumulated record count. The external record or block count is printed out first in the error message; then the internal record or block count is printed.
17	The block count on an input tape does not agree with the internally accumulated block count. The external record or block count is printed out first in the error message; then the internal record or block count is printed.
18	The HASH TOTAL on an input tape does not agree with the internally accumulated HASH TOTAL.

I/O Error Number	Meaning
19	An irrecoverable parity error has occurred during reading of a file assigned to disk or tape. The message is typed once for each block which is in error unless a USE procedure has been specified. The USE procedure (if any) will be executed and control will be transferred to the statement following the READ statement.
20	An irrecoverable parity error occurred on an output tape or disk file. The USE procedure has been executed, allowing programmatic closing of files which must be saved.
21	An attempt was made to READ from a file opened as OUTPUT.
22	An attempt was made to read from a row of a disk file which was never referenced. To get this error, the record number must be less than the highest record number written and greater than 1. For example, when a RANDOM file is written, but records fall only in rows 1 and 3 of a 3-row file, attempts to access records in row 2 will cause I/O ERROR 22 instead of executing INVALID KEY statements. A row of disk space will be assigned, and the appropriate record will be made available. The contents of the record will be unpredictable.
23	An attempt was made to write on a file which was opened as INPUT.
24	An attempt was made to WRITE on a file which was opened REVERSED.
25	Improper code was passed to the COBOL IO intrinsics.
26	A block of less than eight characters has been read, or a zero record size has been encountered during the reading of a variable-record-length which utilizes the SIZE DEPENDING option.
28	While operating under the Time Sharing System, a SEEK has been issued for a data communications device.
29	An irrecoverable parity error has occurred while reading a tape file which was opened REVERSED. The message will be typed once for each block which is in error unless a USE procedure has been specified. The USE procedure (if any) will be executed, and control will be transferred to the statement following the READ statement.

I/O Error Number	Meaning
31	An attempt was made to READ from a file which is closed or never opened.
32	An attempt was made to WRITE to a file which is closed or was never opened. An attempt was made to SEEK on a file which is closed or was never opened.
34	An attempt was made to WRITE BLOCK on an INPUT file.
35	An attempt was made to WRITE BLOCK on a file opened REVERSED.
37	An attempt was made to WRITE BLOCK on a file which is closed.
69	An attempt was made to write on disk at an address less than 1000. The program will hang in a loop. The program must be discontinued by the user.
71	The number of records within a string on a tape read or written by a SORT was incorrect. This was due to an incorrect READ or WRITE on that tape.
72	Parity or blank tape on write in the SORT has occurred.
74	Parity or blank tape on write in the MERGE has occurred.
76	An error occurred within a string being written by a SORT: the number of records that should have been written did not equal the number written on the designated unit.
79	The number of records that should have been read from other tape units in the final merge pass of a SORT did not equal the number of records written onto the final output tape. However, after action was taken to type this message, the SORT closed the final output reel or executed the user's output routine, signaling end of file. Consequently, the output tape may be used in spite of this error message. The tape unit indicated in this message is meaningless.
80	The total number of records entered as input to the SORT was not equal to the number of records produced as output from the SORT in the final merge pass. However, after action was taken to write this message, the SORT closed the final output file or executed the user's output routine, signaling end of file. Consequently, the output tape may be used in spite of this message. The tape unit indicated in this message is meaningless.

I/O Error Number	Meaning
81	The amount of available disk is insufficient for a SORT.
82	The number of records read from the input does not match the number written io the final output in a SORT.
83	A disk file was passed as an output file to a SORT which was not large enough to hold all of the sorted output data.
84	The amount of disk specified is insufficient to do a disk-only SORT. The program must be rerun with a disk and tape SORT.
85	The number of records read from a string on tape is not the same as the number written by a SORT.
86	No records have been passed to the SORT in a program.
87	A sort record description is greater in length than the record descriptor of a file which is passed as an output file to a SORT.

APPENDIX F

B5700 CANDE COMMANDS--QUICK REFERENCE INFORMATION

sequence-list is defined as follows:

[(sequence-number|END)[(TO|-)(sequence-number|END)]] ...

resequence-info is defined as follows:

[sequence-list[base-sequence-number][+resequence-increment]]

program-parameter-info is defined as follows:

[WITH((PROCESS|IO|STACK|COMMON) = integer) ...]

(ADD|APPEND) file-name[/user-code]sequence-list[RESEQ resequence-info]
- Adds specified records from file-name onto end of work-file.

APPEND - see ADD

BYE
- Logs user out and disconnects terminal.

C - see COMPILE

CALL program[/suffix]program-parameter-info
- Executes a program in the public library.

CC(SHORT|LONG)
- Changes assumed length of terminal carriage.

CHANGE [SOURCE|OBJECT] file-name FACTOR TO integer
- Changes save-factor of existing disk file.

CHANGE file-name TO file-name
- Changes existing disk-file's file-name.

CHANGE [file-name] TYPE [TO] file-type
- Changes disk file's file-type.

(COMPILE|C) [file-name][compiler-file-type]program-parameter-info
- Compiles source version of a disk file to produce object version.

COPY [file-name[/user-code]] TO (PRINTER|PUNCH|TAPE|file-name)
file-name[/user-code]sequence-list[RESEQ resequence-info])

- Copies a disk file or a portion of it to another file or peripheral unit.

CREATE - see DISPLAY

?DATA

- Places terminal in DATA mode.

DELETE [file-name] (ALL|sequence-list) [RESEQ resequence-info]

- Clears some or all of a disk file by sequence-number.

DISPLAY - see LIST

DO - see EXECUTE

E - see EXECUTE

?END

- Places terminal in normal status from TAPE or DATA mode.

EQUATE internal-name = file-name[/user-code][unit ...]

- Changes program file attributes.

(E|EXECUTE|DO) [file-name[/user-code]]program-parameter-info

- Executes specified file-name's object version.

FILE - see FILES

(FILE|FILES)

- Lists names of user's disk files.

(FIND|REPLACE)[FILE file-name[/user-code]][FIRST][LITERAL]
(delimiter sought-string delimiter|mnemonic)
[WITH(delimiter replacement-string delimiter|mnemonic)]
sequence-list [PRINT(SEQUENCE|TEXT|SITE|FILE file-name)] ,)...

- Searches disk file for sought-string, optionally replacing (REPLACE only) by replacement-string.

(FIX|*) sequence-number delimiter sought-string delimiter[replacement-string]

-Corrects a portion of a record in work-file.

GUARD

- Assists in constructing GUARD disk file to be associated with private disk file.

HELLO [user-code[password]]

- Logs out old user, may log in new user.

L - see LIST

(LIST|L|PRINT|P|DISPLAY|D)[($|CHANGES)|file-name[/user-code]]
sequence-list[*|SQUASHED][#|NUMBERED]

- Lists part or all of contents of disk file on terminal.

LIST FILES [TO(PRINTER|TELETYPE|file-name)]
[[file-type][SOURCE][OBJECT]
[LOCKED][UNLOCKED][PUBLIC][SOLEUSER]
[LITERAL string][file-name][/user-code]] ...

- Lists much information concerning user's disk files.

LIST PROGRAM FILES [file-name[/user-code]]

- Lists program file attributes.

LOAD file-name

- Loads existing disk file into newly-created work-file.

(LOCK|UNLOCK|PUBLIC)[[SOURCE|OBJECT]file-name ...] ... WITH guard-file-name]

- Changes disk file security status of specified disk files (WITH used only with LOCK).

(MAKE|CREATE) file-name[file-type]

- Establishes new work-file named file-name having given file-type.

(MERGE|RMERGE) file-name[/user-code]sequence-list[RESEQ resequence-info]

- Merges specified disk file, or portions of it, with work-file.

MONITOR file-name

- Causes changes to work-file to be saved in file-name.

P - see LIST

PRINT - see LIST

PUBLIC - see LOCK

PUNCH file-name[/user-code]sequence-list[RESEQ resequence-info]

- Punches specified disk file, or portions of it, on paper tape in a form suitable for reading back into system.

```
R          - see RUN

REMOVE[[SOURCE|OBJECT] file-name...] ...
           - Removes specified files.

RENAME file-name
           - Changes file-name associated with work-file.

REPLACE    - see FIND

RESEQ [file-name]resequence-info
           - Changes sequence-numbers and moves records about within a file.

RESET      - see SET

RMERGE     - see MERGE

(RUN|R) [file-name[/user-code]][compiler-file-type]program-parameter-info
           - Attempts to execute specified file.

S          - see SEQ

SAVE [save-factor][file-type]
           - Stores a copy of current work-file on disk.

SCH        - see SCHEDULE

(SCHEDULE|SCH) [file-name[/user-code]] TO file-name[AFTER integer]
           - Schedules a series of CANDE commands to be processed.

(SEQ|S) [base-sequence-number][+ resequence-increment]
           - Initiates automatic sequencing mode for input.

(SET|RESET)(ALLOWMSG|BUSY|CONCISE|HELPFUL|MONITOR|NOSTOP|QUICKBYE|QUICKLOG)
           - Changes remote terminal options.

[?]SS      - see TO

[?]STATUS[file-name]
           - Returns status of SCHEDULE line with output file file-name, or
             status of currently-running program or compiler.
[?]STOP file-name
           - Terminates SCHEDULE line with output file file-name.

(?TAPE|TAPE [SEQ[base-sequence-number][+ resequence-increment])
           - Places terminal in TAPE mode.
```

TIME

[?](TO|SS) (logical-line-number|user-code|SITE|SPO)[message]
- Sends message to operator or to another terminal.

TYPE file-type
- Changes file-type of work-file.

TYPE OPTIONS
- Prints remote terminal options.

U - see UPDATE

UNLOCK - see LOCK

(UPDATE|U)
- Updates work-file.

WHATS[SOURCE|OBJECT][file-name]
- Returns file-name, file-type, size, creation-date, creation-time, and save-factor for a disk file.

?[command]
- Notifies CANDE that special action is required.

* - see FIX

command ...; command ...
- Concatenates CANDE commands.

, or blank
- Concatenates parameters to CANDE commands.

"[comment]
- Introduces a comment for CANDE.

[message]←
- Ends current line of input to system.

[message]!
- Causes system to delete current line of input.

[message]
- Logically backspaces one input character.

(BREAK)

- Causes terminal to stop typing and accept input.

(WRU)

- Terminates currently executing programs (if any).

(EOT)

- Terminates currently executing program (if any), logs user out, detaches terminal from system.