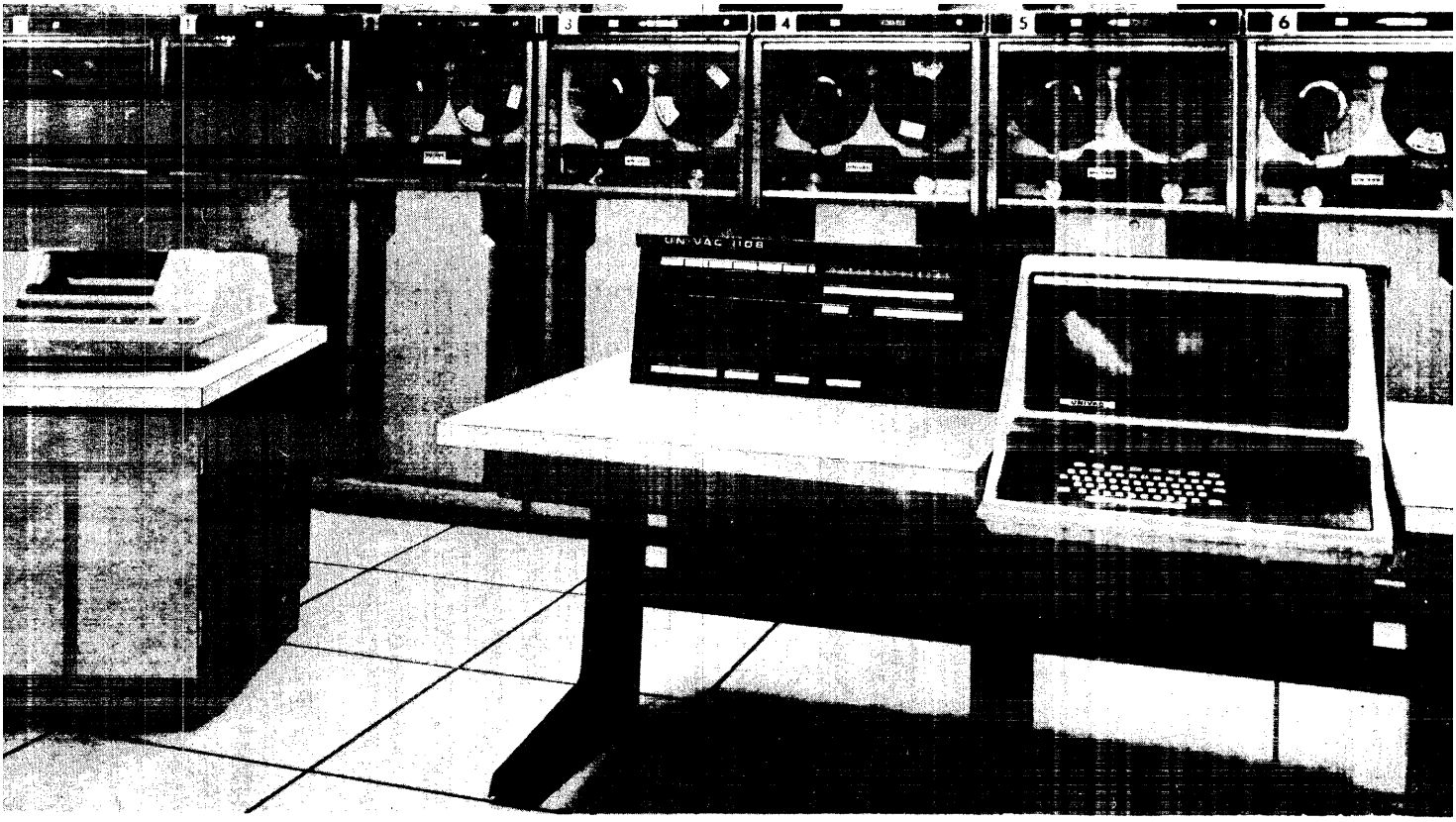


UNIVAC

1108 EXECUTIVE

USERS GUIDE

SPERRY RAND

UNIVAC 1108 EXECUTIVE USERS GUIDE

prepared by

ROBERT W. MOORE

Published by
Univac Marketing Education
UE - 637

TABLE OF CONTENTS

To the User:

The attached UNIVAC 1108 User's Guide is intended to provide a general description of the basic functions performed by the UNIVAC 1108 Executive, and the means by which the worker may utilize such functions to attain his desired end.

Since the 1108 Executive is designed to provide a simple control mechanism for simple applications and yet include the ability to express a complex environment where necessary, the accompanying explanations have been held to a relatively simple level of use.

The User is referred to the UNIVAC 1108 Operating System Executive Programmer's Reference Manual for a more detailed explanation of the working and control of the Executive.

1 INTRODUCTION

1.1 GENERAL DESCRIPTION

The 1108 Executive System controls the operating environment of all activities performed within the 1108 hardware configuration. It produces optimum utilization of the 1108 Central Processors and all other hardware components and, at the same time, is sensitive to the particular requirements of individual users.

Some of the major features of the 1108 Executive System are as follows:

1. Optimize machine facilities usage, and at the same time optimize interaction for all users by means of multi-programming/multi-processing techniques.

2. Make available to remote users the complete facilities of the 1108 System.

3. Provide an executive control language whose structure will allow simple programs to have a simple means of expressing their requirements.

4. Provide the flexibility to express a complex environment for complex programs.

5. Provide a broad and easily-used spectrum of program construction, manipulation, and checkout aids, including

the permanent storage of program elements on random-access devices.

6. Provide for tasks to be executed in either batch, demand, or Real-Time mode.

7. Provide a simple and flexible means of complete software system generation and maintenance at the individual installation.

8. Provide system invulnerability to programming error, and, as far as is reasonable, hardware errors.

9. Provide the simplest possible operational characteristics consistent with full utilization of the capabilities of the system.

The UNIVAC 1108 Executive System is designed to provide effective and efficient utilization of the mass storage devices available with the 1108. Thus giving an unprecendented ability to relieve operators and programmers of responsibilities in maintaining and physically handling cards, magnetic tapes, etc. At the same time, the overall efficiency of operation is considerably improved.

Provision is made for the maintenance of permanent data files and program files on the mass storage devices, with full

facilities for modification and manipulation of these files. Security measures are invoked by the Executive System to insure that files are not subjected to unauthorized use. As unused mass storage space approaches exhaustion, provision is also made for automatic relocation of files of low usage-frequency to magnetic tape. When the use of files relocated in such a manner is requested, they are retrieved and restored, under control of the Executive System, with no inconvenience to the user. For the most part, dynamic assignment of mass storage space is available to the user via the Executive System. To facilitate efficient utilization of available facilities, the user is also able to return portions of mass storage to general use as he finishes with them.

The multi-programming capabilities of the Executive System imply that many unrelated programs may be residing in main storage at the same time. Such programs may be Real-Time runs, classified runs, or simple debugging runs. Infringement of privacy in such a mixture is highly probable, especially in cases where debugging runs are executing.

To combat this invasion, intentional or unintentional, the Executive System has unique features that automatically guarantee absolute protection for each program. The protection guards against two forms of invasion, direct and indirect.

Direct protection safeguards all programs in main storage from an active program that may attempt to read, write, or jump into another program area. This safe guard is effected by "locking out" any area of main storage that is not assigned to the presently active program, or in effect, "locking in" the active program. Any attempt to perform any of the above functions is immediately reported to the Executive System.

Indirect protection is realized by reserving certain control functions for the exclusive use of the Executive System. These functions are of the type that could cause a system malfunction and, in turn, a program malfunction if erroneously used.

In both forms of protection, the Executive System is, in reality, guarantying its own safety and that of other user programs from abuses by the active program that may prove catastrophic.

1.2 DEFINITIONS

Certain terms are referred to in this section with the assumption that the reader is acquainted with their meaning. The following definitions are for the convenience of the reader:

Activity - A division of a program which may be executed independent of other portions of the program.

Activity Registration - The act of registering with the Executive System an activity which can be executed asynchronously with other parts of a program.

Batch Processing - A mode of operation where several runs are grouped prior to processing. Transition from run to run is effected by the Executive System.

Breakpoint - The division of symbiont defined files. Allows those portions of the file to be queued independent of run completion. Maximum use of available printers and punches is achieved in this manner.

Central Site - The 1108 Computer and its attached peripheral equipment.

Collection - The process by which elements of a program are brought together by satisfying the external symbols of the initial element and all referenced elements. The resulting structure defines a program to be allocated and executed.

Communication Device - An input or output device which operates in a Real-Time mode. The central processing unit must be prepared to receive input at any time or the information may be lost.

Demand Processing - The manner of processing in which the Executive System or a processor spontaneously reacts to the inputs from a remote inquiry terminal which is sending messages as required. This is essentially a demand and response type of activity.

Element - The basic component of a program file usually defined and manipulated as a unit. The form of an element is dependent upon the program using it.

Executive Control Language - Specifically formatted input information which is used to direct the activity of the Executive System.

Facilities - The peripheral units, main storage, tape drives, drum storage, etc. of the Central Site.

File - An organized collection of data stored in such a manner so as to facilitate the retrieval of each individual datum.

Granule - The incremental size in which a mass storage unit is assignable.

Multi-programming - The concurrent execution of several programs which occupy main storage. This is accomplished by sharing the attention of the central processor.

Processor Call Statements - Specifically formatted input information which is used to direct the activity of a system processor. A subset of the Executive Control Language.

Program File - A file in which the data are the constituents of a program or of several programs. This data may consist of program elements in a symbolic, relocatable binary, or absolute binary form. Special information in the program file is used to aid the system in the manipulation of the program constituents.

Real-Time Processing - An operating environment in which the response to an external stimuli is sufficiently fast to achieve a desired objective. Depending upon the application, the response time may vary from seconds to microseconds. Generally, Real-Time processing is under the influence of asynchronous inputs from one or more communications devices.

Re-Entrant Coding - A set of instructions coded in such a manner that they may logically perform the same task on different data sets simultaneously.

Remote Site - A communications terminal which is capable of sending information to and/or receiving information from the Central Processor via some common carrier or transmission scheme.

Run - A run is the standard unit in which work is entered into the operating system. This consists of a run command followed by one or more control commands which cause the ordered execution of processors and/or user programs.

Swapping - The process of storing low priority or suspended programs on secondary storage in order to allow space to retrieve another program into primary storage for execution.

Task - A logical step in the processing of a run. For example, execution of a system processor or a user program.

1.3 COMPONENTS OF THE EXECUTIVE SYSTEM

The UNIVAC 1108 Executive System is composed of many different routines, each of which performs specific functions. These routines are organized into several separate groups. For introductory purposes, a brief description of each component group follows:

Supervisor - The supervisor controls the sequencing, setup, and execution of all runs. Among those routines included within the supervisor are the scheduling routines, interrupt processing routines, timing routines, and accounting routines.

Executive Requests - Executive Requests are entrances into the Executive System which provide functions for a user program. Depending on the function, it may be performed asynchronously, synchronously, or immediately. If it is not an immediate request, a queue is maintained.

Symbionts - Symbionts provide the interface between the primary unit record equipment and the user program. These routines are referenced by using executive requests for input and output. Input and output data are buffered on the mass storage devices.

Input - Output Device Handlers - The input/output handlers are responsible for controlling the activities of all I/O channels

and peripheral equipment attached to the UNIVAC 1108. These device handlers provide the user with a full capability of peripheral device operations.

Operator Communications - The Communications Section of the Executive System handles all communications between the operator and the operating programs. This communication takes place via the computer keyboard and on-line printer on the console channel. Neither the keyboard nor the console printer can be assigned exclusively to operating programs.

File Control System - The File Supervisor controls the creation and maintenance of all program and data files. It also maintains an up-to-date master directory of all files catalogued in the system and the availability of all mass storage and magnetic tapes.

Data Handling - The Data Handling routines are designed to process a wide variety of file formats using a general technique. Few restrictions are placed on the formats acceptable to the system.

Files may be processed at the item or block levels with general disregard for the physical characteristics of the I/O device assigned. Data is presented or accepted, randomly or sequentially, on request of the user thereby providing complete operational flexibility for efficient file manipulation.

File Utility Routines/Program Utility Routines (FUR/PUR) - To aid the user in the manipulation of program and data files, a set of file utility routines is provided by the Executive System. These routines perform a variety of functions for system and user data file maintenance.

Auxiliary Processors - A set of auxiliary processors is included in the Executive System. These processors complement the source language processors such as FORTRAN. This set of processors includes the collector for linking relocatable subprograms, and the procedure definition processor for inserting and modifying assembler, COBOL, or FORTRAN procedure definitions in a program-file.

Processor Interface Routines - The processor interface routines provide a simple, standard interface for all processors within the system. Complete facilities are provided for the input of source-language statements and the output of the resulting relocatable binary code.

The Diagnostic System - A comprehensive diagnostic system is available within the 1108 Executive to aid the checkout of user programs. Commands are available which can trigger snapshot dumps at the time of compilation or collection of a user routine. Post-mortem dumps are also available through an Executive Control Statement.

System Setup - The system generation routine provides the means of generating and maintaining a system tailored to the particular needs of each installation.

Utility Routines - Included within the utilities section of the Executive System are diagnostic routines, file conversion routines, and other programming aids.

1.4 FILE CONTROL FUNDAMENTALS

1.4.1 GENERAL

The file control routines exercise centralized control over operations on all files within the system. The primary functions performed by the file supervisor consist of:

1. Maintaining a directory of both catalogued permanent files and temporary files. (A permanent file is one which exists prior to and/or after the termination of a run which references it. A temporary file is one which is created during the course of a run and is not retained after the run terminates).

2. Control allocation of mass storage space as new files are assigned and existing files are expanded.

3. Provide an interface between the worker program and the mass storage device handlers to maintain the absolute

addresses of the various granules of each file.

4. Control the sharing of files.

1.4.2 MASTER DIRECTORY

For each file known to the system, other than temporary files, an entry containing the identification and characteristics of the file is maintained by the system in a master directory of files. The process of entering a file into the master directory is referred to as cataloguing and is effected by control statements. By use of the master directory the system remains cognizant of the usage of mass storage and magnetic tape reel numbers.

An entry exists in the master directory corresponding to each catalogued file. The information contained in each entry includes the following:

1. External name of the file, including qualifier.
2. Project identity from the run control statement.
3. Account number from the run control statement.
4. Date on which the file was catalogued.
5. Activity of the file (including date of last reference).
6. Usage authorization (read and write keys).
7. Recording mode, if tape.
8. Granularity and number of granules assigned if mass storage.

9. Number of reels of tape and tape reel numbers if a tape file.
10. Linkage to the various granules if a mass storage file.
11. F-cycle - absolute and relative.

The activity of the file is maintained to determine which files to roll out to magnetic tape if mass storage is nearing the overflow state.

The master directory shows the recording mode in effect at the time a magnetic tape is assigned. This includes density, parity, and noise constant. If magnetic tape files are catalogued, which were created outside of the system or which had been written and catalogued in separate runs, the recording mode must be entered as parameters at the time of cataloguing if other than the system standard. When a catalogued magnetic tape file is assigned to a run, the recording modes are set to the conditions saved in the directory.

The tape reel numbers are obtained from the operator response to the mount messages for the run which originally catalogued the file and are inserted into subsequent mount messages when catalogued files are assigned.

1.4.3 MASS STORAGE ALLOCATION

The term "Mass Storage" is taken to mean all types of magnetic

drum (FH 432, FH 1782, and FASTRAND). Mass storage is allocated by the 1108 Exec in three basic types:

1. A fixed-length area for the system's residence.
2. A contiguous FH 432 or FH 1782 area for programs.
3. FASTRAND format for both worker programs and the system usage.

The fixed length area of mass storage used by the Executive System is allocated at system generation time and remains fixed in size and location during operation. This area is loaded with a copy of the Executive routines, an area for storing executive tables, and a copy of the processors.

Statements can be used at system generation time to specify one or more areas of FH 432 and FH 1782 magnetic drum which are to be assigned to runs as fixed-length contiguous areas. These areas are intended for the special cases where worker programs need direct usage of high speed mass storage as a scratch pad, and as such, are expected to be a small percentage of the available area. This area is strictly scratching area and may not be catalogued in the master directory.

After satisfying the two requirements discussed above, the remainder of mass storage is treated as FASTRAND format and is allocated in granules of "tracks" and "positions". A track is defined as 1792 words of storage. A position is 64 tracks

or 114,688 words. As an extension to the master directory, the Executive maintains a table locating the various granules which are allocated to a given file name. This table is used by the device handlers to convert the relative locations to absolute hardware locations.

When a mass storage file is initially assigned, only the number of granules requested by the @ASG control statement are allocated. The file supervision routines automatically affect the assignment of additional increments of mass storage space as required to satisfy the needs of the worker programs. The space availability function also handles release of granules to the available status.

1.4.4 ROLLOUT OF FILES

Depending upon the amount of available FASTRAND format mass storage, the degree of usage given to cataloguing files on mass storage, and the manner in which FASTRAND files are assigned, there may be occurrences during normal operation when it is necessary to obtain additional space on FASTRAND by rolling out catalogued files to magnetic tape. This feature is provided automatically by the Executive.

The rollout routine utilizes the file activity and date of cataloguing to determine which files may be rolled out at a

given time. For this determination, all file activity with a frequency of less than one reference per 48 hours are considered eligible. File selection is started at the highest priority FASTRAND format units with the oldest files rolled out first.

The magnetic tape is left extended between references to allow future transfers. Each rolled out file is marked in the directory as to the tape number and file position on the tape. The magnetic tape unit remains assigned to the Executive until all rolled out files are returned to mass storage.

A request to assign a rolled out FASTRAND file causes the Exec to request mounting of the proper magnetic tape, unless already mounted, and automatically retrieve the file back to FASTRAND.

1.4.5 EXCLUSIVE USE OF FILES

The File Control Routines allow assignment of mass storage files to any number of runs at one time, providing exclusive use is not requested in an assignment. Such a request causes a delay in assignment of a file until no other run has the file assigned, and insures that other runs are delayed until a run releases any needed exclusively assigned files.

All magnetic tape files are always exclusively assigned.

1.4.6 FILE NAMES

In the format description of the various control statements, the "EXTERNAL" file name is indicated by "FILENAME" or simply by "NAME". It should be noted that although some of the control statements just specify the "EXTERNAL" file name specification field, the "READ" and "WRITE" keys sub-fields are always implied. (The keys are not part of the file name but are always associated with it). Normally, when a file is referenced in the control stream, it is the "EXTERNAL" name that is intended, although it can always be an internal name.

An "EXTERNAL" file name has the format:

QUALIFIER*FILE(F-CYCLE)

Where the "QUALIFIER", the "*" and the "F(-CYCLE)" are all optional and both the "QUALIFIER" and the "FILE" are limited to 12 characters each from the set A...Z, 0...9, -, and $. The omission of the "QUALIFIER" with the "*" present causes the last preceeding @QUAL statement to supply the qualifier used. If the @QUAL statement has not occured, the project field from the run statement is used as the qualifier. The omission of both the "QUALIFIER" and the "*" also causes the project field from the run statement to be used as the qualifier (provided the "FILE" is not an "ATTACHED" name which points to a particular external

name). The purpose of the qualifier is to allow an additional 12 character uniqueness in the catalogue directory - but more important - for any project, it insures the ability to catalogue the file without name conflict with other projects, as long as the user insures that the "FILE" designators are unique within his project. It should be emphasized that explicit definition of a 'QUALIFIER' for a Filename is the exception rather than the rule. This is due to the implicit qualification by "PROJECT". The only time a user should ever need to explicitly provide a qualifier is when he is referencing a catalogued file created by another project.

The "F-CYCLE" number serves to maintain successive versions of the same file (same "QUALIFIER" and "FILE"). Omission of the "F-CYCLE" implies that the most recently constructed file is intended. A file with a particular F-CYCLE number can be referenced by the absolute F-CYCLE number or by a relative F-CYCLE number. With the relative number, the last file to be produced and catalogued is referenced by "+0" or a blank, the one being produced and to be catalogued by a "+1"; and the backup files by "-1", "-2", etc. As an example, if the last file to be catalogued had an absolute F-CYCLE number of 28, it could be referenced by 28, +0, or a blank with the new file to be catalogued called 29 or +1. Prior to the cataloguing of 29, if a backup exists for 28, it is necessarily called 27 or -1.

When the new file (29) is catalogued, it becomes "+0" with 28 moved to -1 and 27 moved to -2. A plus or minus sign is illegal when the intention is for absolute notation. Absolute F-CYCLE numbers begin with 1 and continue to 999 at which point numbering begins with 1 again. The relative F-CYCLE number allows the user to access a particular relative backup, for example, two cycles prior to the last, with a "-2" at all times. Whereas with the absolute F-CYCLE the number changes with successive runs.

The F-CYCLE number for the new file must be specified as +1, or as the next higher absolute number. This is necessary so that the file can be detected as the one to be placed at the head of a cycle when it is catalogued. If cataloguing is called for at some backup level, such as -2 or the equivalent absolute, all newer files are deleted from the directory and the one being catalogued represents the highest absolute number (or "+0") and the next lower absolute number becomes "-1". If a file is deleted from the directory (other than being replaced as just described), all older files within the F-CYCLE set are also deleted. If a file is being renamed to something outside the set, all older files are given an F-CYCLE number greater than the old. The "OLDNAME" file is treated as though it were being moved outside the set (all older deleted), but when placed back

in the same set with the "NEWNAME", actually a different F-CYCLE, it is treated as a catalogue action (replacement where all newer are deleted).

The number of F-CYCLES maintained for a particular file is determined either as the system standard for all files, or as the number specified by the user via the file utility routine for this file. Automatic deletion of the directory entry for the oldest file occurs when the maximum number to maintain is exceeded. If the file itself is on FASTRAND, it too is deleted. If the file is on magnetic tape, the operator is notified of the directory deletion.

Within each run, the "EXTERNAL" names of the files assigned must be unique. In maintaining uniqueness, any two of the files are unique in one of three ways:

1. Unique by both "QUALIFIER" and "FILE".
2. Unique by either "QUALIFIER" or "FILE".
3. Unique by the F-CYCLE number only.

If one of these conditions is not met, the assignment is rejected by the system and the run is terminated in the error mode. Catalogued files are necessarily unique within themselves; however, the user may define additional files that cause a conflict. In the normal case, the user must guard only against a

conflict among the "FILE" specifications within his given qualifier (project). Two cases arise, however, where there is a conflict among the "FILE" portions of external names. This occurs when the cycling feature is being used and when the names are unique only by qualifier because the user must use a file outside of his project but with the same "FILE" portion as one of his own files.

The "INTERNAL" file name is used by the worker program on an I/O reference to the Executive and specifies the file to be used for the I/O operation. It is limited to a maximum of 12 characters from the set A...Z, 0...9, and $. (No-). The "INTERNAL" name must point to some "EXTERNAL" name before the I/O reference can be honored. This connection is automatic by having the "INTERNAL" name the same as the "FILE" portion of the "EXTERNAL" name (also 12 characters). As an example, if the "EXTERNAL" name on a control statement is J71107*AA001, then an "INTERNAL" name of "AA001" will point to the file.

The "INTERNAL" name need not be the same as the "FILE" portion of the "EXTERNAL" name. For example, an "INTERNAL" name of "AB" can be made to point to the file "J71107*AA001" by the @USE control statement:

```
@USE AB,J71107*AA001
```

This statement causes the name AB to be "ATTACHED" to the

"EXTERNAL" name. This feature allows "INTERNAL" names to be fixed and subsequently connected to any "EXTERNAL" file, depending on the particular run.

In the two cases mentioned above where the "FILE" portions of "EXTERNAL" names are not unique, a @USE statement is required in order to connect an "INTERNAL" name to a particular file involved in the conflict. When an assignment is made, if the "FILE" portion of the "EXTERNAL" name is the same as that of a previous assignment, the file being assigned is marked as not available for I/O reference except via an "ATTACHED" name supplied by a @USE statement to resolve the conflict. In other words, of all files that have the same "FILE" portion on an I/O reference, all others must have "ATTACHED" names. As stated earlier, this situation arises only when the F-CYCLE feature is being used and when the "FILE" portions are not unique because of the necessity to use files from different projects but with the same "FILE" identifiers.

An "ATTACHED" name may be used from within the control stream if specified by a @USE statement. The "EXTERNAL" name (where the project or qualifier is used) will always suffice, however, it may be desirable to use a shorter name or to use an "ATTACHED" name that had to be specified for some other reason. The system treats file specifications in the control stream as follows:

If any part of the "EXTERNAL" name other than the "FILE" portion is given, the name is always treated as "EXTERNAL". If only the "FILE" portion is specified and the "FILE" cannot be found in the "ATTACHED" list, the "EXTERNAL" list is searched for the "PROJECT*FILE". If a match is not found, it is then assumed that the file is a catalogued FASTRAND file yet to be assigned. If an "ATTACHED" name is not intended, but such an "ATTACHED" name exists, the "*" must be specified, even if the "QUALIFIER" is actually the project number. This is the abnormal case and occurs only when there is a conflict between "ATTACHED" names and the "FILE" portion of an "EXTERNAL" name.

1.5 PROGRAM FILE FUNDAMENTALS

The concept of a program file is fundamental to an understanding of the 1108 software system. A program file is essentially a named set of elements. The file name is the prime identifier for the set of elements. To identify and locate the elements within a program file, a table of contents is created, and maintained within the program file by the system.

1.5.1 PROGRAM FILE ELEMENTS

Within the table of contents, each element within the program file is uniquely identified by the following four parameters:

1. Element type
2. Element name
3. Element version
4. Element cycle

Also included are various other parameters such as the date of element creation and the current relative location of the element on mass storage. These parameters are provided and maintained by the system.

The elements contained within a program file are of the following three types:

1. Source language
2. Relocatable binary
3. Absolute binary

Typical source-language elements are the following:

1. FORTRAN source language
2. COBOL source program
3. ASSEMBLER source program

Any of these elements may be introduced into a program file or manipulated within a file by the use of the appropriate processor (FORTRAN, COBOL, etc), or by certain utility routines (See Section 4).

The following elements may be thought of as being special-case source language elements:

1. ASSEMBLER procedure elements
2. COBOL procedure elements
3. FORTRAN procedure elements

Such elements are placed into a program file by the Procedure Definition Processor (PDP).

These elements are available to the language processors essentially as source-language library elements. Special elements are required by the system to facilitate the retrieval of source language library elements at compilation or assembly time. However, these elements are created and maintained by the system and require no concern on the part of the user.

In addition to the above source elements, sets of executive control statements may be entered as source elements. These elements may be called by the @START or @ADD statements.

Relocatable elements are the binary output of the processors such as FORTRAN, COBOL, and the ASSEMBLER. Absolute elements are placed in a program file by the COLLECTOR.

1.5.2 NOTATION FOR PROGRAM FILE ELEMENTS

A consistent notation is used throughout the system to reference

elements of a program file. A reference to an element has the form:

FILENAME.ELEMENT/VERSION(CYCLE)

The notation for the "FILENAME" subfield is identical to that described in Section 1.4.6, including F-CYCLE and read-write key specification. If the "FILE" subfield is omitted, then the run temporary program file is intended. The subfield "ELEMENT" must always be present when referring to an element. The "VERSION" subfield is required only in the case that more than one version of a particular element exists within the program file as is common when a program is in checkout.

On the various control statements which can specify either a "FILE" or an "ELEMENT" name a method is established which distinguishes between them. A period following the "NAME" will specify a "FILE", and no period will specify an "ELEMENT".

The cycle number serves to differentiate successive updates of a symbolic element. Omission of the cycle number when referring to a symbolic element implies that the most recently constructed copy is intended. A compacting method, as described later, is employed to prevent the retention of several cycles of a symbolic element from appropriating an excessive amount of space on whatever storage medium is employed. Some examples will help make all this a bit clearer:

PROG.EDIT	The element edit in the file PROG.
EDIT	The element edit in the run temporary program file.

Element Name and Version

Each element within a program file is given a name specified by the user. This name is referred to simply as the element name. To distinguish between elements of the same name and type, a user may specify a subname for an element, and this subname is called the element version.

Both an element name and an element version may be from one to twelve characters in length, and these two parameters together must uniquely identify one element among all elements of any particular type. Elements of different types (e.g., source language vs. relocatable binary) may, however, have the same name and version. An element name is required for all elements within a program file. (A name is supplied automatically by the Exec in many cases); however, the specification of an element version is not required.

"Cycle Parameter"

For differentiation among symbolic elements, an integer parameter called "CYCLE" is associated with each element. This allows several "copies" of the same version of an

element to be retained within a program file. Each item in a symbolic element has a cycle number indicating to which cycle it belongs, and, if deleted, a delete-cycle number to indicate in which cycle this item was deleted. When a symbolic element is updated, the update items are inserted where they belong in the element and given a cycle number one greater than the last cycle of the element. Any previous cycle items that have been deleted by this update are marked so. The user may make references by cycle number. This gives the same effect as though several different copies of the element were maintained. The user may set the number of update cycles to be retained at any level he desires; however, he need set that number only if he desires to change it from the standard system assumption.

In specifying a symbolic element for compilation or assembly, the user may reference a specific update from a sequence of retained updates by specifying the proper cycle number as part of the Executive Control Statement calling for the compilation or assembly. In compilation, the update entry will be combined with the element in its complete state, thereby creating a complete element as of that cycle.

As soon as the number of updates retained for an element exceeds the specified maximum, the update of the lowest cycle number (the original, complete element) is combined with the update

next lowest in cycle number; in effect, the oldest entry is discarded, and the next-oldest, in its completed form, becomes the oldest to make room for the latest cycle entry. These corrections thus become incorporated permanently into the basic elements and can only be removed by entering new correction statements.

This technique of handling symbolic elements offers two distinct advantages:

1. The user is allowed to keep many differing copies of the same element in a program file while requiring little additional storage over that needed for a single copy.

2. The user is able to refer easily to earlier copies of a specific element without having to prepare corrections deleting previously input corrections. However, if a set of corrections is applied to any cycle except the latest and the updated cycle is to be retained, all cycles that previously followed the cycle to be updated will be deleted. The new cycle number will be the updated cycle number plus one.

2. FUNDAMENTAL USE OF THE CONTROL STREAM

2.1 PURPOSE

Control of the operating environment on the UNIVAC 1108 is accomplished through a set of control statements. These statements direct the executive in scheduling, facility assignment, and in the disposition of program and data files. The control language is designed to provide ease of use for a wide range of applications from the simple to the complex.

2.2 CONTROL STREAM FORMAT

The basic format of the Executive Control Statements is quite simple and is amenable to a large number of input devices, however, since punched card input is the most common type, some of the explanations in the following sections employ card oriented phrasing for purposes of clarity.

2.3 GENERAL CONTENT

@COMMAND, OPTIONS Spec1, Spec2,----,Specn Comment

2.3.1 MASTER SPACE

@ or ▽ represents a "Master Space". It must appear in column one of the image. i.e. 7-8 multiple punch for punched cards or its equivalent for other types of input devices. (# pound-sign for TTY 35).

2.3.2 COMMAND FIELD

The command field must appear in the specific form required for the desired basic operation. The command field is terminated by one or more blank characters. For certain control statements, an options field is appended in which case the command field is terminated by a comma followed immediately by the option/s.

2.3.3 OPTIONS FIELD

The options field allows the user to control the performance and results of the operation by means of unsequenced alphabetic characters. The effect of any such character is determined by the particular operation. In some control statements the options field may be broken into subfields which are separated by a slash (/). The options field is terminated by one or more blank characters.

2.3.4 SPECIFICATIONS FIELDS

The specifications fields are separated by commas and are specified by the user as dictated by his requirements. The content, purpose and number of fields and whether each is required or optional varies with the command selected. Each specification field in turn may contain optional sub-fields which are separated by a slash (/).

2.3.5 BLANKS

Blanks, other than those required, are allowed following the master space (▽), the field separator (,), and subfield separator (/). A blank in any other position is taken to mean the end of a command, options or specifications field.

2.3.6 COMMENTS FIELD

The optional comments field must be preceded by at least one space. The comments field itself may contain any character except the semicolon (;) which is the continuation character. If any trailing specification fields are omitted the comments field must be preceded by a period and a space (.Δ). To avoid possible error, a space,period,space (Δ .Δ) sequence is recommended in all cases where comments are used.

2.4 CONTINUATION RULES

In certain situations, a statement may require more than one card or line. In such cases, coding of a semicolon (;) indicates continuation on the next card or line. A statement may be split at any point, after the options field, where a leading space is allowable or within the comment field. The semicolon is treated logically as a space. The next card or line may begin in any column but a master space character may not be placed in column one.

2.5 SUMMARY OF EXECUTIVE CONTROL STATEMENTS

2.6 THE RUN STATEMENT

The @RUN statement must be the first statement of a run. It identifies the run to the system and supplies accounting information as well as certain optional parameters for use by the scheduler in the system.

Format: (Underlined fields are required)

@RUN, priority/run-options RUN-ID,Acctg,Project,
run-time/deadline,pages/cards,start time

2.6.1 PRIORITY SUB-FIELDS

Contains one alphabetic character from the set A - Z. A represents the highest priority, B the next highest, etc.

2.6.1.1 At system generation time, (SYSGEN) there can be specified:

a. The highest priority allowable for each account number

b. The priority to be used if sub-field is omitted

2.6.1.2 A priority higher than that allowed for the account number is adjusted to the allowed maximum.

2.6.2 RUN OPTIONS SUB-FIELD

The run options sub-field may be used to put certain constraints on the run.

2.6.2.1 Permissible options are as follows:

T - Terminate the run if estimated run time is exceeded (See Section 2.6.6)

P - Terminate the run if estimated number of pages is exceeded (See Section 2.6.7)

C - Terminate the run if estimated number of output cards is exceeded (See Section 2.6.8)

S - This run is to be processed in sequence with the preceding run on the same input device. The priority, deadline and start time of this run will not be considered until the previous run has terminated.

B - This is a batch run being submitted from a terminal which is normally in demand mode, i.e. TTY35 etc.

D - This is a demand run being submitted from a terminal which is normally in the batch mode.

Note: The normal mode of a terminal (demand or batch) is established at SYSGEN.

Note: The priority and run options sub-fields, if used, may be coded in any of the following forms:

1. @RUN,P/O
2. @RUN,P
3. @RUN, /O
4. @RUN,/O

(P & O represent any allowable priority or option)

2.6.3 RUN-ID FIELD

The RUN-ID field can contain up to 6 characters from the set A-Z, 0-9. It uniquely identifies the run to the system.

2.6.3.1 If the RUN-ID duplicates that of a run that is currently in the run queue, the executive will alter the new RUN-ID to render it unique for this run time. The operator is advised of the change, however, the alteration has no effect on this or any subsequent execution.

2.6.4 ACCOUNTING FIELD

The accounting field can contain up to 12 characters from the set A-Z, 0-9,.(period), or -(hyphen). It is used by the system accounting routines to specify computer usage.

2.6.4.1 Those account numbers which are to be acceptable to the system, can be supplied at SYSGEN.

2.6.4.2 If an unknown account number is specified, the operator is advised of the RUN-ID and the account number. The operator can:

a. Abort the run.

b. Accept the new account number for this time only.

c. Accept the new account for this time and add it to the list of acceptable numbers.

d. Change the number to one which is acceptable.

2.6.5 PROJECT FIELD

The optional project field can contain up to 12 characters from the set A-Z, 0-9, -(hyphen), and $. It is used as an implied qualifier for filenames (see section 1.4.6) and for certain entries in the master log maintained by the system.

2.6.5.1 If omitted, the field is treated as 12 fieldata space characters.

2.6.6 RUNNING TIME/DEADLINE FIELDS

The running time field is optional and contains an estimate, in minutes, of Central Processor Unit (CPU) time required for the run.

2.6.6.1 If the field is omitted, the standard supplied at SYSGEN is used.

2.6.6.2 If the time is exceeded, the operator is notified and he can terminate the run or continue as required by the installation.

2.6.6.3 A SYSGEN parameter can be supplied to automatically terminate the run.

2.6.6.4 Deadline sub-field - specifies the time of day or the elapsed time from run submission by which the run must be completed.

2.6.6.5 Maximum deadline time is 24 hours. The time is written as hhmm. A 'D' prefix specifies time of day.

i.e. 345 = 3 hours and 45 minutes after run submission

D1430 = 2:30 P.M.

2.6.6.6 A SYSGEN parameter can be given to allow use of deadline only on specific account numbers.

2.6.7 PAGES SUB-FIELD

The pages sub-field specifies the programmer's estimate of the number of pages of print output to be produced by the run.

2.6.7.1 If omitted the system standard specified at SYSGEN is used.

2.6.7.2 If the pages estimate is exceeded the operator is notified and he can abort the run or continue as required by the installation.

2.6.7.3 A SYSGEN parameter can be supplied to automatically terminate the run.

2.6.8 CARD SUB-FIELD

The use of the card sub-field is identical to the pages sub-field (see section 2.6.7) except that it pertains to the number of output cards.

2.6.9 START-TIME FIELD

The start-time field is used to delay the consideration of the run for execution.

2.6.9.1 Start time is specified in the same manner as time in the deadline field (see section 2.6.6.4/5/6), except that the prefix S replaces the letter D to indicate time of day.

2.6.9.2 If a deadline field is also given, the deadline is not interpreted until the start time has been reached.

2.7 THE @FIN STATEMENT

The @FIN statement is used to signal that the end-of-run has been reached. It is required with all runs and must appear as the last statement. This statement cannot be continued on a second card or line.

The @FIN Statement's format is:

```
@FIN
```

When the @FIN statement is encountered by the Executive System,

the accounting routines are entered and all remaining facilities, temporary files, and core space are released.

2.8 THE @ASG STATEMENT

The @ASG (Assign) Control Statement is used to name an external file, state its I/O Facility requirements, and cause their assignment to the requesting run, under the given external file name. If the file is catalogued, the facility requirements are known and need not be specified in assigning the file as input.

The general form of the FASTRAND @ASG Statement is:

```
@ASG,OPTIONS NAME/KEY1/KEY2,TYPE/RESERVE/GRANULE/MAXIMUM
```

The fields of the statement are explained in succeeding paragraphs and in the order of appearance on the statement.

2.8.1 OPTIONS

The options sub-field is used to cause a file to be catalogued (or decatalogued) and to place or remove constraints on the use of the file.

Cataloguing options are as follows:

C -Specifies that the file is to be catalogued if the run terminates normally.

U -Same as 'C' option except that the file is to be catalogued at run termination regardless of the manner of termination.

R -Specifies that the file is to be placed in the 'Read-Only' state when it is catalogued. A file catalogued with the 'R' option present cannot be over-written. The file can only be read or decatalogued. Any activity attempting to write on the file will be placed in the error mode.

P -Specifies that the file is to be catalogued as a 'PUBLIC' file rather than a 'PRIVATE' file. The distinction between them is that only runs which have the same project as the run which created the file, or which specify the proper Qualifier, can access a 'PRIVATE' file, while any run can access a 'PUBLIC' file.

W -Specifies that the file is to be catalogued as a write-only file. The file can only be written into, and in the process extended.

The above options are for use only with files that are not presently catalogued. If neither of the cataloguing options ('C' or 'U') appear, the file is treated as temporary and released at run termination. It will be released prior to run termination if @FREE is encountered. In the absence of a 'P' option a file is always catalogued as 'PRIVATE'.

Options to be used when the @ASG Statement names a file that is presently catalogued are as follows:

D -Specifies that the catalogued file is to be deleted from the directory (de-catalogued) if the run terminates normally.

K -Same as 'D' option except that the file is to be deleted at run termination regardless of the manner of termination.

X -Specifies that this run is to have 'EXCLUSIVE USE' of the file until the run has terminated or the file is released via the @FREE command. (If the file is not currently catalogued, the 'X' option is not needed because the run necessarily has 'EXCLUSIVE USE').

A -Specifies that the file is currently catalogued and insures that the Executive will terminate the run if the name cannot be found in the directory.

The above options are to be used only with files that are currently catalogued. If either the 'D' or 'K' option appears and the file has either or both keys, the key(s) (see section 2.8.3) must be specified. Failure to do so causes the run to be placed in the error mode.

An option to be used for a temporary file (not catalogued and not to be catalogued) is:

T -Specifies that the file is temporary and allows it to have a name the same as that of a catalogued file. No thought need be given as to whether a file by this name is currently catalogued. If this option is not present for temporary files, the system will attempt to find the file in the directory. If a find is made, the assignment will be made from the directory.

2.8.2 NAME

The field 'NAME' on the @ASG statement is used to specify the 'EXTERNAL' name of the File. The name must be present.

If the file has been created and catalogued by a project number different from that of this run, or with an explicit qualifier, the project or qualifier under which the file was created must be stated with the file name:

QUALIFIER*FILE

2.8.3 KEY1 AND KEY2 SUB-FIELDS

When cataloguing, the sub-fields 'KEY1' and 'KEY2' lock a file against indiscriminate reading and writing, respectively, by other users. They may contain up to six characters and all

characters are legal except the blank, the slash (/), the comma (,), and the semicolon (;). A file is catalogued with 'READ' and/or 'WRITE' lock by specifying the KEY1 and/or KEY2 sub-fields along with the 'C' or 'U' option. To gain read and/or write access to such a file, the appropriate key(s) must be specified at any future assign time or the request(s) will not be honored. If the key(s) are known, a 'LOCKED' file can be partially or completely 'UNLOCKED' or have its key(s) changed by using the File Utility Routine Statement created for this purpose.

A combination of the two keys is used for cataloguing. The following table shows the action allowed according to the key(s) given at cataloguing time and the key(s) given at assign or change time. When 'MESSAGE' appears as an action, a message will be printed indicating that the key is not present and, therefore, not needed:

Key(s) Specified at Cataloguing Time	Key(s) Specified at Assign or Change Time			
	Read	Write	Both	Neither
Read	Read Write	Write Message	Read Write Message	Write
Write	Read Message	Read Write	Read Write Message	Read
Both	Read	Write	Read Write	Abort
Neither	Read Write Message	Read Write Message	Read Write	Read Write

NOTE: The following 'TYPE', 'RESERVE', 'GRANULE', and 'MAXIMUM' fields are generally not required when a previously catalogued file is to be read.

2.8.4 TYPE

The sub-field 'TYPE', specifies that the statement applies to FASTRAND format and, in addition, points out the type of equipment to be used. It is ignored if given for catalogued files. The allowable types for the FASTRAND @ASG Statement format are:

F4 FASTRAND simulated on FH 432

F8 FASTRAND simulated on FH 880

F17 FASTRAND simulated on FH 1782

F2 FASTRAND Model II

FB FASTBAND

F FASTRAND, Type Independent

A file placed on drum simulated FASTRAND (F4, F8, F17) has all the characteristics of a FASTRAND file except for sector Zero Fill on write functions.

2.8.5 RESERVE/GRANULE

The sub-field 'RESERVE' is used to specify the approximate number of granules to be used by the file. The sub-field 'GRANULE' is used to specify the granule size. In certain cases, either or both sub-fields may be omitted. If the granule sub-field is specified it must contain either 'TRK' or 'POS'.

TRK -- Specifies a granule of one track (1792 words)

POS -- Specifies a granule of one position (114,688 words)

If the granule sub-field is omitted, it is assumed to be 'TRK'. The granule sub-field is ignored if the file is currently catalogued.

The reserve sub-field is ignored and need not be specified when the file is catalogued and is to be read only. If the file is to be created or updated, the reserve may contain an integer

specifying the number of granules to reserve for the file (on an update the reserve specification includes that portion of the file that already exists). If the reserve specification is omitted, no granules (or additional granules) are initially assigned, they are assigned dynamically as needed. When the reserve is supplied but exceeded, additional granules are also assigned dynamically as needed.

Note: When creating a file, the reserve sub-field should contain a reasonable estimate of the number of granules needed. If a file can be contained within the limits of the reserve, the run is assured of being able to create the file without delay (with dynamic expansion the requesting program may be removed from core while FASTRAND is being made available). In addition, the specification of a reserve aids the Executive in allocating FASTRAND area efficiently (if a reserve is used the track or positions will be adjacent, if possible).

For most efficient use of mass storage, all files that are to be program files should be allocated under track granules ('TRK'). A specification of 'POS' creates unused space in the program file in that 64 contiguous tracks will be assigned.

If the file takes fewer granules than reserved, the empty granules are returned to the available status when the file is catalogued. Furthermore, if 'POS' is specified in the granule sub-field and complete tracks in the highest referenced granule have not been referenced, this non-referenced space is put back into the available pool at run termination. The reserve value is placed in the directory and will be used on future updates unless a new reserve is supplied on the update @ASG Statement, in which case it replaces the previous value in the directory.

2.8.6 MAXIMUM

The sub-field 'MAXIMUM' is used to indicate that the run is to be terminated if the length of the file being created or updated exceeds the number of granules specified. It may also be used to override the 'SYSTEM-MAXIMUM' for all files, as specified by the particular installation at SYSGEN.

If a maximum was supplied when the file was catalogued, its value is retained and used when an update occurs. If a maximum is supplied on the updating @ASG Statement, it is placed in the directory, thereby replacing the previous maximum.

Consider the following examples of @ASG Statements for FASTRAND:

@ASG,CR FILEX,F/5

FILEX is to be catalogued in the permanent 'READ-ONLY' mode if the run terminates normally, five tracks are assigned initially and a maximum length is not specified.

@ASG,D FILEX/A2294B

FILEX is currently catalogued and is to be de-catalogued if the run terminates normally, the key A2294B is required to read the file.

@ASG,T FILEX, F/4/POS/5

FILEX is a temporary file requiring 4 FASTRAND positions to be reserved initially; termination is to occur if more than 5 positions are required.

@ASG,X FILEX /6//8

FILEX is currently catalogued and this run is to have 'EXCLUSIVE USE' of the file for updating; a reserve of 6 tracks is specified and the run is to be terminated if more than 8 tracks are used.

2.9 THE MAGNETIC TAPE @ASG STATEMENT

For Magnetic Tape the format of the @ASG Statement is:

```
@ASG,OPTIONS NAME/KEY1/KEY2,TYPE/UNITS/LOG/NOISE,;
REEL1/REEL2/...REELN
```

The NAME field and the 'KEY1' and 'KEY2' sub-fields are the same as for the FASTRAND @ASG Statement. The name must always appear.

Options to be used for cataloguing are:

C Same as for FASTRAND

U Same as for FASTRAND

P Same as for FASTRAND

R Same as for FASTRAND

Options to be used when the file is presently catalogued are:

D Same as for FASTRAND

K Same as for FASTRAND

A Same as for FASTRAND

The option to specify a file as temporary is:

T Same as for FASTRAND

The following options, called the 'MODE OPTIONS', are used to override the modes established at SYSGEN.

L Low Density (200 FPI)

M Medium Density (556 FPI)

H High Density (800 FPI)

E Even Parity

B Binary (No Translate)

I Decimal (Translate FIELDATA to BCD on WRITE, BCD to FIELDATA on READ). If hardware translate option is available the E option must also be specified.

If the equipment'TYPE' is nine-channel per frame (see below), the density is fixed at high, the parity is fixed at odd, and any attempt to change these settings is illegal. Hardware translation is not available on nine-channel per frame units.

When a file is to be catalogued, the options placed in the directory are those which were in effect at the time of first I/O reference by a program within the run (other than the 'SET MODE' Reference).

NOTE: The field 'TYPE/UNITS/LOG/NOISE' is called the 'FACILITIES' field and normally does not have to be specified if the file is currently catalogued.

2.9.1 TYPE

The sub-field 'TYPE' is used to show that the @ASG Statement is for Magnetic Tape and contains a symbol denoting the type of tape unit required.

2.9.1.1 The sub-field is required if the file is not presently catalogued, but is ignored if already catalogued.

Allowable types are:

T Tape, type independent

C UNISERVOS VIII C, VI C, OR IV C

U UNISERVOS VIIIC, OR VI C

8C UNISERVOS VIII C

6C UNISERVOS VI C

4C UNISERVOS IV C

3A UNISERVOS III A

2A UNISERVOS II A

The use of C or T is recommended to allow the system more freedom in efficient allocation of units.

2.9.1.2 If the hardware translate option is not available on all units it may be specified by a "B" following the type, i.e. 8CB. = UNISERVO VIII C with hardware translate.

2.9.1.3 If the installation has a mixture of 7 track and 9 track units, a "9" following the type will select a 9 track unit, i.e. 6C9 = UNISERVO VI C 9 track.

2.9.2 UNITS

The sub-field units is used to specify the number of magnetic

tape units to be assigned to this file. The integers "1 or 2" may be used. If omitted a "1" is assumed. On subsequent assignments of the file the units as indicated in the master directory are used unless changed by the current @ASG.

2.9.3 LOG

The log sub-field may contain a single letter. The system will, if possible, assign all files with the same log letter to the same physical channel. The log letter is not placed in the directory entry.

2.9.4 NOISE

The noise sub-field may be used to override the system standard noise constant.

NOTE: CAUTION - Changing of the noise constant should be used only in extreme cases and then with great care.

2.9.5 REEL

The reel sub-field may be used to list the specific tape reels to be used. Each sub-field (Reel1/Reel2/ etc.) contains a reel identifier hereafter called reel number.

2.9.5.1 Each reel number can be 1-6 alphanumeric characters. (A-Z, 0-9).

2.9.5.2 Files Being Catalogued

2.9.5.2.1 Reel numbers omitted cause Executive to request loading of blank reels and operator supplies reel numbers for the directory entry.

2.9.5.2.2 Reel numbers specified are placed in the directory whether or not all reels are used. Mounting is requested in the sequence of the numbers as they appear on the @ASG Statement. When all numbers are used the system will request blanks for any additional reels and will enter their numbers in the directory.

2.9.5.3 Files Currently Catalogued

2.9.5.3.1 Field is normally blank indicating reels are to be used in the sequence in which they were created.

2.9.5.3.2 If reel numbers are supplied they must be of the set listed in the directory but may be only part of the set and/or in any sequence.

NOTE: In either of the above cases, when the known reels are exhausted, blank reels will be requested and their numbers added to the directory.

2.9.5.4 Temporary Files

2.9.5.4.1 If reel numbers are supplied they will be requested in sequence.

2.9.5.4.2 When specified reels are exhausted, or if field is omitted, blank reels are requested but their numbers are not required of the operator.

2.9.6 EXAMPLES OF MAGNETIC TAPE ASG STATEMENTS

@ASG FILEA

FILEA is catalogued and all necessary options, facility requirements and reel numbers are taken from the directory.

@ASG,TEL FILEB,8C/2,N432

FILEB is a temporary file to be recorded with even parity, at low density. It requires two UNISERVO VIII C's and reel number N432 will be requested first.

@ASG,U FILEC/492671/RA1234,8C/2,707/708/709/710

FILEC is to be catalogued regardless of type of run termination, two units are required for reels 707 through 710 and the 'READ' and 'WRITE' keys will be required in future assignments.

2.10 THE @FREE STATEMENT

The @FREE Control Statement makes provision for the de-assigning

of a file and the release of its input/output facilities. In the absence of a @FREE Statement, the file and its facilities are held until end-run. Files should be de-assigned as soon as they are no longer needed so as to allow facilities, reels, and 'EXCLUSIVE USE' areas to be assigned to other runs. The format of the @FREE Statement is:

```
@FREE,OPTIONS   NAME1,NAME2,...,NAMEN
```

where 'NAME1', 'NAME2', etc., are the 'EXTERNAL' names of files to be de-assigned. All 'EXTERNAL' names must have been previously assigned.

A file that is named on a @FREE Statement can no longer be referenced by the run; it can, of course, be re-established by an @ASG Statement, provided its facility requirements can be met.

2.10.1 OPTIONS

S. Will cause the physical assignment for the file to be held for subsequent use by the same run through another @ASG Statement.

The actions taken by the system when a file is named on a @FREE Statement are discussed below.

2.10.2 FOR A TEMPORARY FILE (NOT CATALOGUED OR TO BE CATALOGUED):

FASTRAND - The FASTRAND area is made available as file space for other runs.

DRUM -- Same as FASTRAND. Always temporary.

TAPE - Units are released for use by other runs. The operator is notified that the reels are to be removed and that the file was not catalogued.

OTHER EQUIPMENT - The device is released for use by other runs. Always (Comm's.Gear,etc.) temporary.

2.10.3 FOR A FILE BEING CATALOGUED (C OR U OPTION ON @ASG):

FASTRAND - Catalogue entry is made in the Master Directory and FASTRAND area containing the file is held. The file can now be referenced by other runs.

TAPE - Catalogue entry containing reel numbers is made, units are released for other runs. The operator is told to remove the file (REELS) and that the file was catalogued.

2.10.4 FOR A FILE BEING DE-CATALOGUED (D OR K OPTION ON @ASG):

FASTRAND - Same as for a temporary file except that the file area is not released until all runs currently using the file have also finished. It is no longer available for assignment.

TAPE - Same as for a temporary file.

2.10.5 A typical @FREE Statement is shown in the following example of a partial control stream:

```
@ASG,C      FILEX,F/3
@ASG,T      FILEY,8C
.......
.......
@FREE       FILEX, FILEY
```

FILEX is a FASTRAND file to be catalogued and requires 3 tracks initially.

FILEY is a temporary tape file requiring 1 VIIIC unit.

When the @FREE Statement is encountered, FILEX is catalogued with the file area held for future reference. For FILEY, the tape unit is made available to other runs and the operator is notified to remove the reels and follow the user's instructions as to their disposal.

2.11 THE @USE STATEMENT

The @USE Control Statement provides the ability to refer to a file by two or more names. The need for this ability arises from three principal conditions:

a. To simplify file referencing by equating a short internal name to a long or complex external name.

b. to resolve duplications of external names.

c. To allow constant internal names to reference different external names in different environments.

2.11.1 FORMAT

@USE Δ INTERNAL,EXTERNAL

2.11.1.1 OPTIONS

No options are allowed in the @USE Statement.

2.11.1.2 INTERNAL

The internal name can be 1-12 characters from the set A-Z, 0-9, -, and $. It represents the name by which a file is referenced within the run.

2.11.1.3 EXTERNAL

The external name is the name of the file as specified in the @ASG Statement. The external name is of the form:

QUALIFIER*FILENAME(F-CYCLE)/KEY1/KEY2

2.11.2 NOTES

1. More than one internal name may be attached to one external name.

2. All internal names are maintained as applicable to an external file.

2.12 PROCESSOR CALL STATEMENTS

The system processors will process a source language element to produce a relocatable binary element. The processor call statement calls the appropriate processor into main memory and starts execution.

2.12.1 FORMAT

@PROCESSOR,OPTIONS EL1,EL2,EL3

2.12.2 PROCESSOR FIELD

The processor field contains the specified acronym applicable to the desired processor (if available within the particular system)

a. COB = COBOL

b. FOR = FORTRAN V

c. ASM = ASSEMBLER

etc.

2.12.3 OPTIONS

The options field may contain any number of the following letters to specify the desired actions.

A - Accept the results of the processor even if errors are detected.

X - Abort the run, if errors are detected.

NOTE: If neither X nor A are given the run will continue but execution of the processed program in error will be inhibited.

U - Update (correction runs only). Produce a new cycle of the source element named in Field 1.

I - Insert an element into the Program File named in Field 1.

L - Produce a complete listing (source and object code).

N - Suppress all printing of processor output (overrides any other print option).

S - Produce a partial listing.

W - List correction lines in a separate listing (Used with 'U' option).

NOTE: Absence of all print options will suppress all printing.

2.12.4 ELEMENT 1

The element 1 field names the source language element to be used by the processor.

2.12.4.1 It may be omitted if there is no 'I' option present and the source language statements immediately follow the processor call. If so, only the RB output is placed in the run temporary program file (TPF) under a system generated element name.

2.12.4.2 If the 'I' is present ELEMENT 1 must appear and the source language statements must follow the processor call. If so, both source language and RB output are placed in the TPF with the same element name.

2.12.4.3 If the 'U' option is present, ELEMENT 1 must appear to identify the element which is to be updated, and the correction cards must follow the processor call.

2.12.5 ELEMENT 2

The element 2 field is optional and is used to provide a specific element name for the RB element produced by the processor.

2.12.6 ELEMENT 3

The element 3 field is optional and is used to privide a specific element name for an updated source element. If element 3 is used, the 'U' and "I" options may not be present.

2.13 FORMAT OF CORRECTION LINES

Each processor provides a list of all source language statements

submitted to it and places successive integral numbers by each statement. These numbers are used to indicate where corrections are to be applied.

2.13.1 Correction lines are identified by a minus sign in column one, followed by one or two integral numbers separated by a comma.

2.13.2 DELETIONS

The form -76,81 indicates that lines 76 through 81 inclusive of the element are to be deleted. No source language statements may follow this card. (See replacements). The form -53,53 will delete line 53 only.

2.13.3 REPLACEMENTS

The form -107,132 followed by any number of source statements will replace lines 107 through 132 inclusive, with the source statements which follow. The number of old and new lines have no interrelation, i.e. replace one old with ten new or ten old with one new.

2.13.4 INSERTIONS

The form -432 followed by source statements will insert the new statements following line 432. Any number of statements may be inserted.

2.14 THE XQT STATEMENT

The XQT (Execute) Statement is used to initiate execution of an absolute program prepared by the collector (See Section 2.15). In the absence of an absolute element, the relocatable elements in the file will be collected and executed.

The general form of the XQT statement is as follows:

@XQT,Options ELEMENT

2.14.1 OPTIONS

The options field may be used to contain options for use by the program by means of the ER OPT$ function.

Option letters generate a 26 bit mask with bit position 25 set to a 1 bit for an A option letter, bit 24 for a B, etc.

2.14.2 ELEMENT

The element field (FILE.ELT/VER) names the specific element (absolute element only) to be executed.

2.15 THE MAP STATEMENT

The casual user of the 1108 System does not need to prepare a @MAP control statement unless his run embodies one of the following conditions:

1. All of the relocatable elements to be allocated (other than system library routines) are not contained within a single file.

2. The program requires overlays (segments).

3. The program file contains structural ambiguities such as duplicate entry points, multiple starting addresses, etc.

✓ 4. The absolute element is to be retained so that re-allocation will not be required for subsequent executions.

It should be noted here that the @MAP control statement is in reality a processor call statement for the Collector. The Collector must be executed to combine a set of relocatable elements into a single absolute element before the program may be executed. Only absolute elements produced by the Collector can be executed.

If the user does not provide a @MAP control statement, the system simulates the statement and calls the Collector on the occurrence of an @XQT control statement.

For performing the collection of complex programs which require relocatable input from many sources, construction of overlay segments, or the use of multiple libraries, the user must prepare a set of source language control statements. These

statements immediately follow the @MAP executive control statement.

Use of the system relocatable library to satisfy external references is automatically implied; the use of user libraries is under control of a source language control statement to the Collector. Any specified user libraries are always searched before the system relocatable library.

The outputs of the Collector are as follows:

1. An absolute or relocatable element.
2. A source language control element as discussed above.
3. Listing information.

The primary output of the Collector is the relocatable or absolute element which results from the collecting and linking of the various relocatable elements. This element is given a name and placed within a program file for subsequent use. Both the element name and the file in which the element is placed may be dictated by the user.

For any error condition encountered, the Collector produces an error message which is placed in the user's listing output file. If a storage allocation map is required, the listing of such is also placed in the user's listing output file.

The general form of the @MAP Statement is as follows:

@MAP,Options NAME1,NAME2,NAME3

NOTE: Names of the form FILE.ELT/VER(CYCLE) as applicable.

2.15.1 OPTIONS

I = Introduce source language element into program file from the control stream. The first field identifies the element. The third field never appears.

U = Produce a new cycle of the source language element.

L = Produce a complete listing containing a summary of the core space used by the program, the space allocated to each element, and the absolute definition of all external definitions.

N = Produce no listing. Diagnostic messages are always printed. If neither L nor N are coded, only summary information will be printed.

A = Accept the results of the collection so long as an absolute element is produced.

X = Do not execute the remainder of the run if errors are detected. If neither A or X is coded, the occurrence of an error will inhibit execution of the program but will allow continuation of the run.

Z = Inhibit generation of the diagnostic information normally provided to the diagnostic system.

R = Produce a relocatable element rather than an absolute element.

2.15.2 NAME1

The field "Name1" normally identifies the source input element and the file in which it is located. Whenever the I option is used, this field instead identifies the source output element and the field "Name3" does not appear.

2.15.3 NAME2

The field "Name2" is used to specify the output file, and the element identification, for the resulting absolute or relocatable element.

2.15.4 NAME3

The field "Name3" is used to specify the output file and the element identification, for a source language output element.

2.15.5 Some typical examples involving this form of the @MAP Statement are as follows:

```
@MAP
    (Source language follows this card and the absolute output
     is to the temporary file).
```

@MAP,I FILEA.JOE

(The statements following the @MAP statement are used to direct the collection and are output to FILEA as element JOE, CYCLE1; the absolute output is to a temporary file).

@MAP,U FILEA.JOE (6)

(CYCLE 7 of JOE is produced using corrections following the @MAP statement with both outputs to a temporary file).

@MAP JOE/XYZ,FILEA.JOE

(Element JOE, VERSION XYZ, latest cycle, from a temporary file is used to direct the collection of the absolute element JOE written into FILEA).

@MAP,R JOE,XX

(Element JOE in a temporary file is used to direct the production of a relocatable element output to a temporary file).

A description of the Collector Control Language is contained in a following part of this manual entitled "The Collector".

2.16 THE EOF STATEMENT

The @EOF Statement is used as a file divider or file terminator for punched card files input to a worker program.

The format of the @EOF Statement is as follows:

@EOF 's'

Where 's' is a one character sentinel which is transmitted to the worker program when the @EOF statement is found in the input data stream in response to the workers ER READ$ instruction.

An example of the use of the @EOF statement follows:

```
@XQT     PROGCR
-----
DATA
-----
@EOF  A
-----
DATA
-----
@EOF  B
@FIN
```

2.17 THE @PMD STATEMENT

The @PMD Statement may be used to dump core memory after execution of a task. Dumps may include all or part of the user's core as long as they are currently in core at the time the task terminates.

The @PMD Statement may be used in two formats as follows:

1. @PMD,Options NAME1,NAME2,....,NAMEN
2. @PMD,Options NAME,START,LENGTH,FORMAT

2.17.1 OPTIONS

2.17.1.1 The options available for both form 1 and 2 are:

E = The @PMD Statement will be honored only if the previous task terminated in error.

C = Only those words which were changed during execution of the task will be dumped.

B = Dump all of blank common storage after the rest of the @PMD has been honored. If used with the 'C' option the 'C' is ignored for blank common storage.

2.17.1.2 Options available only for form 1 are:

A = Dump all of memory from elements named in the specification field.

D = Dump only the D bank for the named elements.

I = Dump only the I bank for the named elements.

2.17.1.3 Options to be used only with the A, I or D options are:

X = Dump all active (in core) elements except those named in the specification field.

L = Dump all active elements from the system library.

P = Dump the user's PCT (Program Complex Table).

2.17.2 SPECIFICATIONS

2.17.2.1 The specification sub-fields for form 1 constitute a list of element or segment names. If a segment name is given, the segment must be in core at termination in order to be dumped.

2.17.2.2 The specification sub-fields for form 2 are as follows:

Name = The required name of an element.

Start = The Start sub-field is of the form N/M. Where M is the location counter of the element to be dumped and N is an address, relative to the beginning of M, at which the dump is to start. If either value is omitted, Ø (zero) is assumed.

Length = The number of words to be dumped. If omitted, all words for the location counter will be dumped.

Format = The user specified format (using FORTRAN specifications) in which the dump will be taken.

User defined formats are not allowed for changed word dumps (C option). If a special option (see 2.17.1.2) is used or if the format is omitted, octal format is assumed.

2.18 THE @MSG STATEMENT

The @MSG Statement is used to type a message on the central site console and/or place the same message in the run print file.

The format of the @MSG Statement is as follows:

@MSG,Options Message

2.18.1 OPTIONS

W = Type the message on the console typewriter followed by 'RUN-ID WAIT'. This option is used when the message indicates that some operator action is required. The execution of the run is suspended until the operator answers 'GO'. If the W option is not present the run continues without requiring an answer.

N = Suppress typing on the console. The message is placed in the run print file only.

2.18.2 SPECIFICATIONS

The message may be a maximum of 122 characters starting with the first non-blank character after the COMMAND,Options field and ending with the end-of-line, comment field or the 132 character maximum.

The semicolon ; and the space-period-space sequence may not be included as part of the message.

A carriage return character may be included within the message and will cause a line feed and carriage return.

2.19 THE @HDG STATEMENT

The @HDG Statement allows the user a means of automatically printing a heading on each succeeding page of the print file.

The format of the @HDG Statement is as follows:

@HDG,Options Heading Text

2.19.1 OPTIONS

P = Begin page number with 1. If omitted, and a previous @HDG statement has appeared in the run, the page numbers for this heading will be in sequence with those of the previous heading.

X = Do not print date or page count.

N = Terminate printing of heading.

2.19.2 HEADING TEXT

The heading field may contain a maximum of 96 characters starting with second position of the statement after the @HDG,Options. The date and page number will appear at the right of the heading unless suppressed by the 'X' option.

3. EXAMPLE DECK SETUPS

The following sections will cover some of the most common examples of program deck setups. It should be noted that these examples are general in nature and do not cover all possible variations.

3.1 COMPILE ONLY

```
@RUN  NIMBLE,J711040,$HOPE
@COB
.....
COBOL Source Language
.....
@FIN
```

In this example the COBOL source language program is compiled and the resulting relocatable element put into the run-temporary file. The specifications on the RUN statement refer to the RUN-ID, ACCOUNTING field PROJECT, respectively, reading from left to right.

3.2 COMPILE AND EXECUTE

```
@RUN  TEST,J71104,ZEUS
@FOR,I SQRT
.....
```

```
FORTRAN Source Language
.....
@XQT
@FIN
```

In the above example the FORTRAN source language is compiled and the resultant relocatable element SQRT, is executed. The same results would be attained, in this example, if "SQRT" were deleted from the @FOR card.

3.3 COMPILE AND EXECUTE MAIN PROGRAM WITH TWO SUBROUTINES

```
@RUN  TEST,J71104,ZEUS
@FOR,I MAN
.....
FORTRAN Source Language
.....
END
@FOR,I SUBR
Subroutine SQRT
.....
FORTRAN Source Language
.....
END
@ASM,I DIVIDE
```

```
.....
ASSEMBLER Source Language
.....
END
@XQT
@FIN
```

The above example illustrates the compilation of a main program and two subroutines, which will be allocated together and executed.

3.4 COMPILE AND CATALOG ORIGINAL PROGRAM

```
@RUN TEST,J71104,APOLLO
@ASG,U PROGFILE/READK/WRITEK,F
@FOR,I  PROGRFILE.MAIN
.....
FORTRAN Source Language
.....
@FOR,I  PROGFILE.SUBP
Subroutine SQRT
.....
FORTRAN Source Language
.....
@ASM,I  PROGFILE.DIVIDE
```

.....

ASSEMBLER Source Language

.....

@FIN

The Assign (ASG) statement is used to name an external file, set-up its I/O requirements and catalog the file for future reference. The "U" option on the ASG card specifies that the file is to be catalogued regardless of the manner of termination of the run. "PROGFILE" is the name of the file to be assigned to the run. "READK" and "WRITEK" are specification fields which prevent reading and writing of the user's file by other users. To gain access to the file the appropriate keys must be specified at assign time or the assignment will not be made.

"F" indicates that the file will be located on Fastrand.

The processor control statement (@FOR) specifies that a source language element will be introduced into the file "PROGFILE" from the control stream and also given to the processor for compilation. The "I" option on the @FOR card directs this introduction of source language.

3.5 UPDATE EXISTING PROGRAM AND EXECUTE

```
@RUN,/TP  TEST,J71104,APOLLO,20,300
@ASG  PROGFILE/READK/WRITEK
@FOR,UW  PROGFILE.MAIN
Correction Statements
.....
MAP,I PROGFILE.MINT2
@XQT  PROGFILE.MINT2
@FIN
```

On the RUN statement, the "20" in the specifications field is programmer estimated run-time. If the run exceeds this time the run will be terminated. The "300" in the last specification field means that a maximum of 300 pages of output is expected from this run. If exceeded, the run will be terminated automatically.

The Assign (ASG) statement is used to name an external file, "PROGFILE," which contains the program source language for this run.

In the above example corrections will be made to FORTRAN element "MAIN" before compilation. The correction statements must immediately follow the processor call statement. The "U" option on the @FOR statement specifies that an updated source

language element will be produced by applying corrections to the input source language. The "W" option specifies that all correction statements will be listed.

Subroutines "SUBR" and "DIVIDE" are also part of this program (see previous example). Since there are no modifications to these routines, they are not recompiled.

The main element "MAIN" with subroutine "SUBR" and "DIVIDE" will be collected together and an absolute element "MINTI", which will also be placed in the file, "PROGFILE". The absolute program is to be executed; data cards follow the @XQT statement.

3.6 EXECUTE EXISTING PROGRAMS USING CATALOGUED DATA FILES

```
@RUN  MINT,J71104,APOLLO
@ASG  PROGFILE/READK
@ASG  J71100*MASTER-FLT
@USE  MASTER,J71100*MASTER-FLT
@ASG,T TEMP,F/3/POS
@XQT  PROGFILE.MINI
.....
Data Cards
.....
@FREE  J71100*MASTER-FLT
@ASG  MINT
```

```
@USE  OLD,MINT
@ASG,CR  MINT(+1),F
@USE  NEW,MINT(+1)
@XQT  PROGFILE.MINT2
@FIN
```

In the above example two programs, MINT1 and MINT2, are to be executed in respective order. The programs are currently contained as absolute elements in file PROGFILE.

The program, MINT1, reads data cards and a file, MASTER. For this run a file created under account, J71100 is to be used as MASTER. A temporary file, TEMP, is created.

The program, MINT2, updates the file, MINT, referencing the current cycle of the file as OLD and the updated cycle of the file as NEW. The temporary file created by MINT1 is used as the basis for the update.

Note that the file, J71100*MASTER-FLT, is not required by the run after MINT1 is terminated. The @FREE statement releases the file so that another run might gain exclusive access to the file during the time that MINT2 is being executed.

4. FILE UTILITY ROUTINES

In addition to the Executive Control Statements previously described, there is a set of commands which are recognized as calls for the File Utility Routine and the Program File Utility Routine (FUR/PUR).

When the Executive encounters one of these commands, the FUR/PUR Processor will be loaded into core. (An @XQT statement is not required). When FUR/PUR receives control it will access the command which caused the processor to be loaded. After completing the action required by the first command, FUR/PUR will continue to read and process commands until it is determined that the next command is not directed to FUR/PUR at which time FUR/PUR will terminate and relinquish its core space.

4.1 STATEMENT FORMAT AND RULES

```
@COMMAND,Options   Spec1,Spec2,....,Specn
```

Where Spec1 --- Specn are of the form:

```
Qualifier*Filename(F)/Key1/Key2. Element/Version (C)
```

4.1.1 The run temporary file (TPF) may be referenced by omitting the file name and giving only the element name.

4.1.2 All rules for referencing a file by name apply as for Executive control statements. (Except as specified in 4.1.4).

4.1.3 The options field serves the same purpose as in the Executive Control Language.

Options and their meanings vary between commands except for the following, which are applicable to most commands:

C = Attempt to continue processing in case of error detection.

S = Process only symbolic elements (including PROCS).

R = Process only relocatable elements.

A = Process only absolute elements.

4.1.4 Files referenced in FUR/PUR commands must be assigned to the run via an ASG statement except that the ASG may be omitted if the file is a catalogued mass storage file.

4.1.5 Certain FUR/PUR commands manipulate elements within files. In these statements the Spec field includes the file name and the element description.

4.1.6 FUR/PUR manipulates elements within program files or element files. A program file is a random file on mass storage. (Hereafter called FASTRAND). When a program file is removed from

FASTRAND for any reason by means of a @COPOUT command it is placed on magnetic tape in a sequential equivalent called an Element file. An element file must be restored to FASTRAND program file format with a @COPIN statement before any language processor or the collector can access the data.

4.1.7 The commands which are directed to FUR/PUR are listed below with a brief explanation of the functions to be performed; each is described in detail in subsequent paragraphs.

COMMAND	FUNCTION
@COPY	To copy a file or a program file element from one device to another.
@COPIN	To copy an element file onto FASTRAND and reformat it as a program file.
@COPOUT	To copy a program file and reformat it as an element file.
@DELETE	To delete an element from a program file or a file from the master directory.
@CHG	To change the name and/or version of an element in a program file.
@PACK	To purge a program file of deleted elements.

@PRT	To list the table of contents of a program file or text of symbolic elements.
@PREP	To create an entry point table for a program file.
@CYCLE	To specify the number of cycles for a symbolic program file element.
@PCH	To punch symbolic or relocatable elements from a program file.
@FIND	To locate a specific element of an element file on tape.
@MOVE	To position a tape (element file) past a specified number of end-of-file marks.
@ERS	To return the program file to its initial condition and make its space available for reuse.
@REWIND	To rewind a tape (element file).
@MARK	To write an end-of-file mark on a tape (element file).
@CLOSE	To write an end-of-file mark and rewind a tape (element file).

4.2 THE COPY STATEMENT

The @COPY command copies an entry file by reading from tape or FASTRAND and writing to tape or FASTRAND. @COPY manipulates both program and data files. The only partial file operation allowed is to add an element from one FASTRAND program file to another FASTRAND program file.

The format of the @COPY Statement is as follows:

```
@COPY,Options   Spec1,Spec2
```

4.2.1 OPTIONS

(None) An entire data file is to be copied from the input file named in Spec1 to the output file named in Spec2. The input file Spec1 must have been written sequentially with the data handling routines and must be catalogued.

An entire file is to be copied. Spec1 and Spec2 are the input and output files. The input file is in System Data Format (SDF) as produced by the @COPOUT command and will be written in the same format.

P An entire program file is to be copied from Spec1 to Spec2 (both of which are FASTRAND files). Deleted elements are not copied (see @DELETE statement), and

all copied elements are packed. The table of contents for Spec2 is updated. The entry point table of Spec2 is not reconstructed and the file must be prepared by the @PREP statement after any subsequent @COPIN.

The following options are available for dealing with elements:

I = The input file named in SPEC1 is source code data in System Data File Format (SDF). These data are to be added to a program file. The program file and the identification to be given to the new element are described in SPEC2 as PF.EN/V. The symbolic element is given a cycle number of 0 when it is inserted into the program file.

S
R
A These three options may be used singly or in combination for copying elements from one program file to another. (Refer to 4.1.3)

4.2.2 SPECIFICATION FIELDS

The specification fields are written as PF.EN/V(c), with cycle only applicable to symbolic elements. In SPEC1, the element name is required, but the version need be specified only if required to identify the element. If cycle is omitted for a symbolic element, the latest cycle will be assumed.

In SPEC2, the entire EN/V(c) may be omitted if the SPEC1

EN/V(c) is to be used for the output. If SPEC2 contains an EN without a version (and cycle for option S), the output will be given a blank version (and cycle 0 for option S).

4.2.3 Some typical @COPY statements are given below:

1. @COPY,P PROG1.,PROG2.

   The program file PROG1 is reproduced onto PROG2. Deleted elements are not copied and the table of contents is revised to include only the non-deleted entries (same as in PACK).

2. @COPY,I SDFILE.,PFILE.ELT1

   A file of source code data in System Data File Format (SDF) on file SDFILE is added to the program file named PFILE and identified as symbolic element ELT1, no version name, Cycle 0.

3. @COPY,S FILEA.ELTA,FILEB.ELTB/VERSB(2)

   The latest cycle of symbolic element ELTA in the program file named FILEA is reproduced in the program file named FILEB where it is identified as element name ELTB, version name VERSB, Cycle 2.

4. @COPY,S .ELTA,FILEB.

   Element ELTA in the temporary program file is copied to

FILEB, and can be referenced later as FILEB.ELTA with a version name of blanks (none necessary), and the cycle number is 0. (R and A options are applicable also).

5. @COPY,P ,FILEX.

   The entire temporary program file (excluding deleted elements) is copied to FILEX.

6. @COPY,S ,FILEX.

   The symbolic elements in the temporary program file are copied to FILEX.

4.3 @COPOUT

The @COPOUT command copies a program file or program file element from FASTRAND onto magnetic tape and reformats the data into element file format. Procedure name table entries are preserved so that the program file can be fully reconstructed with @COPIN. Deleted elements are not copied into the element file. If a packed file is copied out, the entry point table is not preserved. After a @COPIN, the user should execute @PREP to allow the program file to be used as a library.

The @COPOUT Statement is in the following format:

```
@COPOUT,OPTIONS  SPEC1,SPEC2
```

4.3.1 OPTIONS

The options available to the user are (S), (R), and (A) and they may be used to copy the symbolic, relocatable, or absolute elements from a program file on mass storage to an element file on tape.

4.3.2 SPEC1,SPEC2

The file named in SPEC1 must be a program file or, if the field is blank, the run-temporary program file. SPEC2 must name an element file on tape.

4.3.3 EXAMPLES OF THE USE OF @COPOUT

The @COPOUT Statement is typically used in the following manner:

1. @COPOUT PROGRAM.,HOLDPROG.

   The program file named PROGRAM will be copied onto the output file HOLDPROG. It will be reformatted as an element file.

2. @COPOUT,S PROGFIL.ELT1, ELTFIL.ELT1

   The symbolic element ELT1 from the program file PROGFIL is copied onto the element file ELTFIL. The element name remains the same.

4.4 @COPIN

The @COPIN command copies an element file from magnetic tape onto FASTRAND and reformats the data into program file format. The table of contents is rebuilt to include element table and procedure tables if any were present in the program file when @COPOUT was executed.

The format for using the COPIN Statement is as follows:

```
@COPIN,OPTIONS    SPEC1,SPEC2
```

4.4.1 OPTIONS

The S, R, and A options apply as with @COPOUT.

4.4.2 SPECIFICATIONS

This command will cause the file or element indicated by SPEC1 to be reformatted and written out on SPEC2. SPEC1 must indicate an element file on magnetic tape. SPEC2 must indicate a program file on FASTRAND.

4.4.3 EXAMPLE OF THE @COPIN STATEMENT

The @COPIN Statement is typically used in the following manner:

```
@COPIN   HOLDPROG.,PROGRAM.
```

In this example, the element file HOLDPROG is copied and re-

formatted on the FASTRAND area assigned to file PROGRAM. When the @COPIN operation is complete, file PROGRAM will be in the standard program-file format and may be treated as a program-file in any subsequent operation.

```
@COPIN,R   TEMP.ELTA,PF1.
```

The above example causes the relocatable element (ELTA), to be read from the element file assigned the external name 'TEMP' to be added to the program file PF1. The element file must be positioned at ELTA if ELTA is not the first element of file 'TEMP'. When adding to the program file, care must be used if it has been prepared. If it has been, it will have to be prepared again (via a @PREP Statement).

4.5 @DELETE

The DELETE command may be used to delete one or more entries from the master file directory or one or more elements from a program file on FASTRAND.

If a whole file is deleted from FASTRAND, the storage area is released for re-use. If the file is on tape, the reels are released. If a program file element is deleted the element table entry is flagged but the physical storage area on FASTRAND remains assigned until a @PACK, @COPY or @COPOUT command is found.

The @DELETE Statement has the following format:

@DELETE,OPTIONS SPEC1,.....SPECN

4.5.1 OPTIONS

The available options for program file element deletion are 'S', 'R', and 'A'.

4.5.2 SPECIFICATIONS

Several elements of the same type may be deleted by the same command. Each element must be described by name and must be from the same file. Version name should be included as needed to further identify the element.

Deletion of multiple elements in a file may be accomplished by adding additional specification fields to the control card (viz. @DELETE SPEC1,.SPEC2,.SPEC3,....) where SPEC1 is the form FILE.EL/VR, and SPEC2 through SPECN are of the form .EL2/V2, .E3/V3 etc. (see example 2 and 3 below).

4.5.3 EXAMPLES OF @DELETE STATEMENTS

Some typical @DELETE Statements are as follows:

1. @DELETE,S PROGRAM.ELT1

   Symbolic element ELT1 is to be deleted from the file PROGRAM.

2. @DELETE,A PROGFILE.SAM/XYZ,.JOE

Absolute elements SAM (Version XYZ) and JOE (Version not needed for uniqueness) are to be deleted from the file PROGFILE.

3. @DELETE,S PF1.ELTA,.ELTB,.ELTC,.ELTD

This control card will cause symbolic elements ELTA, ELTB, ELTC and ELTD in Program File PF1 to be deleted.

4.6 @CHG

The @CHG command changes the name and security keys of a catalogued file or optionally changes the element name and/or version name of an element in a program file on FASTRAND.

4.6.1 The format to change the name of a catalogued file is as follows:

@CHG SPEC1.,SPEC2.

4.6.1.1 If SPEC1. was originally catalogued with security keys they must be specified in SPEC1. If SPEC2 contains no keys then the file is effectively "unlocked" (previous keys are removed). If SPEC2 does contain keys, they will become the applicable keys for the file (whether different or not).

4.6.2 The format to change the element and/or version names is as follows:

@CHG,OPTIONS FILE1.EL1/V1,.EL2/V2

4.6.2.1 OPTIONS

The allowable options are S, R, and A.

4.6.2.2 FILE1 identifies the program file in which EL1/V1 is located. The sub-file V1 is required only when needed to uniquely identify the element. The field EL2/V2 contains the new names for the element and version. If the field EL2 is omitted, only the version will be changed.

4.6.3 EXAMPLES OF THE @CHG STATEMENT

1. @CHG OLDF/RDZ/WT$., NEWF//WT$

   The file name OLDF is changed to NEWF and the read key RDZ is removed to allow free reading of NEWF. The write key WT$ is retained.

2. @CHG,SRA PROGF.SUBR/ONE,.STAND/PRIME

   The symbolic, relocatable and absolute elements of program file PROGF, element SUBR, version ONE will be renamed element STAND, version PRIME.

4.7 @PRT

For a program file, there are two special options for @PRT which list either the table of contents of a program file or the text of a specified symbolic element.

The @PRT Statement has the following general format:

@PRT,OPTIONS SPEC1,SPEC2,...SPECN

4.7.1 OPTIONS

T - List the table of contents for the program file named in SPEC1. Additional program files may be specified in SPEC2, etc.

No Option - List a symbolic element. SPEC1 is FN.EN/V(c). Only one element may be listed by this option. Relocatable and absolute elements can not be listed.

4.7.2 EXAMPLES OF @PRT STATEMENTS

Some typical uses of the @PRT Statement are shown below:

1. @PRT,T PROGFILE

   The table of contents of the program file named PROGFILE are listed. The entries flagged as deleted are listed as well as the active entries. Procedure table and entry point table entries follow the element table entries.

2. @PRT PROGFILE.SAM/XYZ

   The latest cycle of symbolic element SAM, Version XYZ in PROGFILE, is listed.

4.8 @PACK

The @PACK command causes all deleted elements to be physically removed from a program file on FASTRAND and revises the table of contents to include only the non-deleted elements. Procedure table entries are removed if they are flagged as deleted or point to deleted element table entries. The complete entry point table is removed. If the newly packed file is to be used as a library, it must be re-prepared (@PREP).

The @PACK Statement is used in the following format:

```
@PACK          FILEA.,------,FILEN.
```

4.8.1 OPTIONS

No options are used.

4.8.2 SPECIFICATIONS

FILEA is the name of the file to be packed. More than one file may be named in a pack statement. The actual packing function will consist of the rewriting of some, though not necessarily all, elements, and a rewriting of the table of contents.

4.8.3 EXAMPLE OF THE @PACK STATEMENT

A typical use of the @PACK Statement is shown below:

```
@PACK     PROGR1.,PROGR3.,UPROGRAM.
```

The program files PROGR1, PROGR3, and UPROGRAM are to be packed. Deleted elements are to be dropped, the tables of contents are to be reconstructed and any assignable granules of mass storage released by the packing function are to be returned to the system for reassignment.

4.9 @PREP

The @PREP command is used to 'prepare' a program file, on FASTRAND for subsequent referencing as a library by the collector. The @PREP statement causes an entry point table to be generated which contains all the entry points (external definitions) of all relocatable elements in the file.

A file must be prepared when elements in the file are to be added to a program collection from that file merely by referencing an entry point in the element. The file must also be named on a 'LIB' collector control statement for the automatic inclusion. If all desired elements from the file are included in the collection as a result of source language statements, the file need not be prepared.

The @PREP Statement is used as follows:

```
@PREP   FILEA.,FILEB.,-----.,FINEn
```

4.9.1 OPTIONS

No options are used.

4.9.2 SPECIFICATIONS

FILEA is the name of a program file. If this field is blank, the run temporary program file is assumed. In its processing, @PREP will review all relocatable elements, extract all entry points and create a new entry point table. More than one program file may be named in a @PREP statement.

4.9.3 EXAMPLES OF THE @PREP STATEMENT

1. @PREP

   All the entry points in the run-temporary program file are put into the entry point table.

2. @PREP A.,B.

   An entry point table is created for File A and also for File B.

4.10 @PCH

The @PCH command causes a symbolic or relocatable element of a program-file on FASTRAND to be punched out in 80 column punch card format.

The @PCH statement is used in the following formats:

@PCH,OPTIONS FILENAME.ELEMENT/VERSION

4.10.1 OPTIONS

The allowable options are 'S' and 'R'.

4.10.2 SPECIFICATIONS

The field 'FILENAME.' will be a program file. If omitted, the TPF will be assumed. The field ELEMENT names the element to be punched. The field VERSION is required only to guarantee uniqueness to the element name.

Only one element may be punched by a single @PCH command.

4.10.3 EXAMPLES OF THE @PCH STATEMENT

Some typical uses of the @PCH statement are shown below:

1. @PCH,SR PROGA.SAM/XYZ

   The symbolic and relocatable elements named SAM, version XYZ of program-file PROGA are to be punched.

2. @PCH,S .DUMP/10K

   The latest cycle of the symbolic element dump, version 10K, of the run-temporary program-file is to be punched.

3. @PCH,RS MAINPROG.LISTING (10)

   The relocatable element and cycle 10 of the symbolic element named LISTING of program-file MAINPROG are to be punched.

4.11 @ERS

This directive is used to remove all elements from the named file and restore the file to its initial condition. The file will be considered as empty and available for use just as it was at the beginning of the run.

This statement is provided primarily for use on temporary program file TPF which may be used as scratch area during a run.

```
@ERS   SPEC1
```

Where SPEC is the name of a file assigned to this run.

4.11.1 EXAMPLE OF AN @ERS STATEMENT

```
@ERS   FILEA.
```

Remove all traces of data in 'FILEA' such that if it were to be copied a blank file would result on the output side.

4.12 @CYCLE

The @CYCLE statement is used to specify the number of update (c) cycles to be maintained for a symbolic program file element or to change the number of F-cycles to be retained for a data file.

A standard number is established at system generation for the retention of both (c) and (f) cycles. The @CYCLE command need

not be used unless the standard is to be changed for a specific file or element.

If the number of cycles in existence at the time the @CYCLE command is executed is greater than the new number specified, the number in existence will be reduced with the oldest cycles being deleted in order of age.

The @CYCLE statement is used in the following format:

```
@CYCLE    FILEA.ELEMENT/VERSION,N
```

N is the maximum number of cycles to be retained.

4.12.1 EXAMPLE OF THE @CYCLE STATEMENT

```
@CYCLE    BASEPROG.SYMBOL/A221,10
```

This command tells the system that ten (10) update cycles are to be retained for the symbolic element symbol, version 'A221' of program file BASEPROG.

```
@CYCLE    MAINDATA,3
```

Three cycles are to be retained in the catalogued file MAINDATA.

4.13 @FIND

The @FIND command will locate a specific element of an element file on magnetic tape. When the element is found the tape will

be positioned so that a @COPIN will process the element.

Formatting the @FIND Statement:

The @FIND statement is used in the following format:

```
@FIND,OPTION   FILEA.ELEMENT/VERSION
```

The allowable options, denoting element type are 'S', 'R', and 'A'. One (and only one) must be present.

The field FILEA must be the name of an element file on tape. The fields ELEMENT and VERSION identify the particular element sought. Version may be eliminated if not needed to uniquely identify the element.

4.13.1 EXAMPLE OF THE @FIND STATEMENT

A typical use of the @FIND statement is shown below:

```
@FIND,S    ELTFILE,BLITZ/CLOTH
```

Find the symbolic element BLITZ, version CLOTH on element file ELTFILE. When the element is found, the file will be positioned so the next @COPIN will bring in the element BLITZ, version CLOTH.

4.14 @MOVE

The @MOVE command will move any tape file forward or backward over a specified number of end-of-file marks.

The @MOVE statement is used in the following format:

```
@MOVE,OPTION      FILEA.,N
```

4.14.1 OPTIONS

If the option field is omitted the tape is moved forward. If the tape is to be moved backward, the option is 'B'.

4.14.2 SPECIFICATIONS

FILEA is the name of the tape file to be positioned, and N is the number of end-of-file marks (decimal) past which the tape is to be moved.

4.14.3 Some typical uses of the @MOVE statement are:

1. @MOVE ELTFIL.,2

   This statement will cause the element file, ELTFIL., to be moved forward past two end-of-file marks.

2. @MOVE,B HARRY.,1

   This statement moves data file HARRY backward past one end-of-file mark. A forward @MOVE can be used to move over the file mark to the beginning of the next data file on the tape.

NOTE: The system makes no check of the information passed when searching for an end-of-file mark. The user must be sure that the tape contains at least 'N' end-of-file marks.

4.15 @REWIND

The @REWIND Statement rewinds any magnetic tape file with or without interlock.

The format of the @REWIND statement is as follows:

```
@REWIND,OPTION FILEA.,FILEB.,...
```

4.15.1 OPTIONS

The only allowable option for the REWIND command is 'I' which is used to specify a rewind-with-interlock. The absence of the 'I' option denotes rewind to load point.

4.15.2 SPECIFICATIONS

Parameters FILEA and FILEB are the names of the element files to be rewound. More than one file may be referenced in a @REWIND statement. Each file referenced must be on tape with a mounted reel.

4.15.3 EXAMPLE OF THE @REWIND STATEMENT

A typical example of the @REWIND statement is as follows:

```
@REWIND,I ELTFIL.
```

The file ELTFIL is to be rewound-with-interlock.

4.16 @MARK

The @MARK command writes 2 EOF marks on an element and backspaces over the second EOF mark.

The format for using the @MARK statement is as follows:

```
@MARK    FILEA.,FILEB.,----.,FILEn
```

The field FILEA must contain the name of a tape file. More than one file may be called in a @MARK statement, but each must be assigned and pre-positioned.

4.16.1 EXAMPLE OF THE @MARK STATEMENT

A typical use of the MARK processor call statement is shown below:

```
@MARK    DATA1.,TPFLE.,ELTFLE.
```

The three named files currently mounted, are to have end-of-file marks written at their present positions.

4.17 @CLOSE

The @CLOSE command combines the functions of @MARK and @REWIND.

The @CLOSE statement is used in the following manner:

```
@CLOSE,OPTION    FILEA.,FILEB.,----.,FILEn
```

4.17.1 OPTIONS

The 'I' option will cause the named tape file to be marked and rewound with interlock.

4.17.2 SPECIFICATIONS

The field FILEA is the name of a mounted element file. More than one file may be named in a close statement. Two EOF marks will be written on the file at the point where it is positioned when the @CLOSE call is given.

4.17.3 EXAMPLE OF THE @CLOSE STATEMENT

The following is a typical use of the @CLOSE statement:

```
@CLOSE    ELTFILE.
```

Two end-of-file marks will be written on the element file ELTFILE and it will be rewound without interlock.

5 THE COLLECTOR

5.1 GENERAL

The Collector is a system processor designed to provide the user with a means of gathering (collecting) and interconnecting one or more relocatable elements to produce a program in a form ready to be loaded into memory and executed. This program form is called an absolute element. Optionally the collector can be used to produce one relocatable element from a collection of several relocatable elements.

5.1.1 INPUTS TO THE COLLECTOR

The three basic inputs to the collector are:

1. Parameters from the executive control statement causing the collection (@MAP).

2. Source language control statements (IN, SEG, etc.).

3. Relocatable elements from a variety of sources.

5.1.2 THE COLLECTION PROCESS

The Collector collects relocatable elements out of program files according to the source language control statements.

The program files may be:

1. The run-temporary file (TPF$)

2. User program file libraries

3. The relocatable system library

5.2 THE @MAP STATEMENT

The @MAP Statement is used to load the collector in order to combine one or more relocatable elements into an absolute element or a single relocatable element.

The format of the @MAP statement is as follows:

```
@MAP,OPTIONS   SPEC1,SPEC2,SPEC3
```

5.2.1 OPTIONS

I =Initial Insertion: The following source language control statements will direct the collection. SPEC1 names the output source element. SPEC3 is not used. If no source statements follow, IN TPF$ is assumed. (See source control statements below).

U =Update: Produce a new cycle of source language element using the corrections which follow. SPEC3 is not used.

L =Produce a complete listing. Will contain a summary of core space used by the program, the space allocated to each element, the program address of all definitions, and the external references of each element.

N =Produce no listing. Diagnostic messages are always printed. If neither L nor N are coded, only summary information is printed.

X =If any errors are detected, inhibit execution of the run. (Except a @PMD statement). Normal action is to accept results of the allocation as long as an absolute element is produced.

R =Produce a relocatable element rather than an absolute element.

5.2.2 SPECIFICATION FIELDS

The "SPEC" fields are of the form FILE.ELT/VER(c)

SPEC1 normally identifies the source input element. When the I option is present it identifies the source output element.

SPEC2 normally identifies the absolute output element. When the R option is present it identifies the relocatable output element.

SPEC3 is optional. If used it names the output source element and the I and U options may not be present.

5.2.3 EXAMPLES OF THE @MAP STATEMENT ARE AS FOLLOWS:

@MAP SYMIN/LAT,BACKUP.ABSOUT

Element SYMIN/LAT in TPF$ is used to direct the collection of element ABSOUT which is written to file BACKUP. If any corrections follow they will be used but not saved because no output source element is produced.

@MAP,I BACKUP.SYMOUT,.ABSOUT

The statements following the @MAP are used to direct the collection and are output as SYMOUT in file BACKUP. The absolute element ABSOUT is also output to file BACKUP.

@MAP OLDFIL.OLDELT,A,NEWFIL.NEWELT

The source statements in element OLDELT of file OLDFIL, as amended by correction statements following the @MAP, are used to direct the collection, and are output to file NEWFIL element NEWELT. The absolute output element 'A' goes to TPF$.

5.3 COLLECTOR CONTROL STATEMENTS

The source language collector control statements can be used to control the building of an absolute element by specifying many relocatable elements from many sources.

The control statements recognized by the collector include the following:

IN Include specific files or elements in the collection.

NOT Exclude specific elements from the collection.

LIB Specify user libraries to be searched.

SEG Direct the segmentation of a program.

5.3.1 THE 'IN' STATEMENT

The IN control statement allows the inclusion of full files or only specific elements from named files. Particularly the user can specify the elements to be included in a segment named in a preceding SEG statement.

The format of the IN statement is as follows:

IN FILE1.ELTA/VER1,FILE2,FILE3.E3.E4

Include only VER1 of ELTA of FILE1, all elements of FILE2 and elements E3 and E4 of FILE3.

If no file is named, TPF$ is assumed. An element name may appear in only one IN statement in any collection.

5.3.1.1 EXAMPLES OF THE IN STATEMENT ARE AS FOLLOWS

IN FILEA.,FILEB.

All elements of FILEA and FILEB are included in the collection.

IN FILEB.AA,.BB,DD

Elements AA and BB of FILEB and element DD of TPF$ are included in the collection.

5.3.2 THE 'NOT' STATEMENT

The NOT control statement is essentially the inverse of the IN statement. It is used to state explicitly which elements of a file are to be included in the collection.

The format of the NOT statement is as follows:

NOT FILE1.E1,.E2,FILE2.E1

The specified elements are to be omitted from the collection, i.e. E1 and E2 of FILE1 and E1 of FILE2.

If only an element name is given, then all elements of the same name will be excluded.

5.3.2.1 EXAMPLES OF THE NOT STATEMENT ARE AS FOLLOWS

NOT AA,BB

All elements in the TPF$ except AA,BB are included in the collection.

IN FILEA

NOT FILEA.AA,.BB

All elements in FILEA are included except AA and BB.

IN FILEA.,FILEB.

NOT FILEA.AA,.BB,FILEB.CC,.DD

All elements of FILEA except AA and BB and all elements of FILEB except CC and DD.

5.3.3 THE LIB STATEMENT

The LIB statement allows the user to specify program libraries to be searched to satisfy external references and/or to find elements specified without filenames which were not found in the TPF$.

The format of the LIB statement is as follows:

LIB FILE1,FILE2

5.3.3.1 The names of files to be treated as libraries are specified in successive fields. The libraries are searched in the sequence given and before the system library. The same file may be searched more than once by naming it in more than one LIB statement. Files will not be searched for external definitions unless prepared by the FUR/PUR @PREP statement.

5.3.3.2 EXAMPLES

LIB USER1

File USER1 will be searched before the system library.

```
LIB     USER1,USER2
```

Files USER1 then USER2 will be searched before the system library.

5.3.4 THE SEG STATEMENT

The SEG statement is used to define the relationship and contents of segments within a program.

The format(s) of the SEG statement is as follows:

```
SEG     NAME1,NAME2
 or
SEG   NAME1,(NAME2,NAME3,----,name_n)
```

5.3.4.1 The field NAME1 is required and is the name of the segment.

5.3.4.2 The first segment named in the source input is called the Main segment and is never overlaid by any other segment. It should be specified using only the NAME1 field.

5.3.4.3 If NAME2 is blank then segment NAME1 starts immediately following the preceding segment.

5.3.4.4 If NAME2 is present and is not inclosed in parentheses the NAME1 segment will start at the same location as NAME2 segment.

5.3.4.5 If NAME2, NAME3, etc. is inclosed in parentheses, NAME1 segment will start following the highest location of any of the NAME2, NAME3, etc. segments.

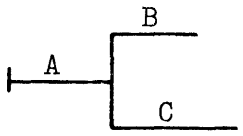
5.3.4.6 Each segment may have both an I Bank and a D Bank.

5.3.4.7 At least one IN statement must follow each SEG statement.

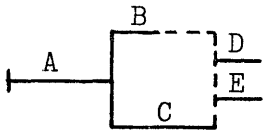
5.3.4.8 If NAME1 is suffixed by an asterisk, i.e. "NAME1*", the segment may be loaded indirect (automatic load).

5.3.4.9 Following a schematic diagram can best illustrate the results of various SEG statements.

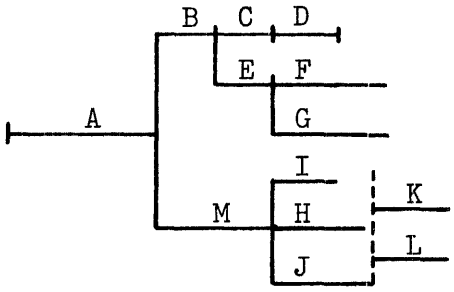
a.
```
SEG  A
SEG  B,(A)
SEG  C,B   or SEG C,(A)
```



b.
```
SEG  A
SEG  B, (A)
SEG  C,B
SEG  D, (B,C)
SEG  E,D
```



c.
```
SEG  A
SEG  B,(A)
SEG  M,B
SEG  C,(B)
SEG  D
SEG  E,C
SEG  F,(E)
SEG  G,F
SEG  H,(M)
SEG  I,H
SEG  J,H
SEG  K,(H,I,J)
SEG  L,K
```



5.3.4.10 SEGMENT LOADING

When a segmented program is called into memory for execution (@XQT control statement) only the main segment is initially loaded. Other segments may be loaded by the direct or indirect method as explained below.

5.3.4.10.1 DIRECT LOADING METHOD

A secondary segment is loaded directly by the procedure call:

```
L$OAD    NAME,JUMP,CLEAR
```

NAME is the name of the segment to be loaded (Spec. field 1 of the SEG statement). JUMP is the address to receive control when the segment is loaded. If omitted, control returns at the point after the call. CLEAR, if non zero, inhibits the clearing to zero of the area in which the segment is to be loaded.

5.3.4.10.2 INDIRECT LOADING METHOD

The indirect method of loading a segment is applicable only to segments whose names were suffixed by an asterisk in the SEG statement. This method provides for a segment to be loaded automatically when referenced by any jump command to an instruction area. The referenced segment will be loaded if not already in memory and the jump will be executed.

SPERRY RAND UNIVAC