MISSION AND DATA OPERATIONS

IBM 360

USER'S GUIDE

Prepared for

Goddard Space Flight Center
Greenbelt, Maryland 20771

Prepared by

Mr. Jack Balakirsky
Operating Systems Maintenance Section
Goddard Space Flight Center
GSFC Code 543.1

and

Computer Sciences Corporation
8728 Colesville Road
Silver Spring, Maryland 20910

September 1971

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

Section 5 - Job Set-Up

Section 6 - Standard (IBM-Supplied) Processors

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

FOREWORD

The purpose of this document is to provide a central reference vehicle for all information relating to the use of the M&DO IBM 360 computer facilities. While the primary objective is to provide information relating specifically to the M&DO computer installation, which may not be found elsewhere, there has been an attempt to include sufficient additional information so that a user may either find the information he needs directly or be referred to the proper document for more detailed information.

It is also intended that this document be continually updated, as new versions of the operating system are installed, changes are made to the configuration and the operating environment; all pertinent information regarding these changes will be distributed as updates to this document.

The M&DO IBM 360 User's Guide is available to all users of the M&DO IBM 360 computers, including both GSFC and contractor personnel. Readers are encouraged to offer suggestions as to the content and organization of the document so that these may be considered for inclusion in future updates.

Revised: September 1971

PREFACE

The Mission and Data Operations User's Guide was developed incrementally, portions being made available as they were completed. In order to provide a context for the sections that are presented, a full Table of Contents is provided.

Written comments, suggestions, and constructive criticism will be helpful in producing a useful and accurate guide. Mr. Jack Balakirsky, Code 543.1, Goddard Space Flight Center, Extension 6796, will coordinate the incorporation of user responses into subsequent revisions of this document.

This document was prepared from an outline developed by Mr. Robert C. Danek of the Computer Systems Branch, Code 543. The document necessarily contains a considerable amount of information edited from other publications. We extend an appreciation to the many personnel at Goddard who contributed their expertise in the development of this User's Guide.

SECTION 1

INTRODUCTION

1.1 PURPOSE

Operating System 360 (OS 360) is a powerful control program amply documented - the manuals produced by IBM relating to OS 360 require more than six feet of shelf space. It is not the intention of this guide to duplicate IBM's efforts. Operating System 360 is quite flexible and usually no two implementations of OS 360 will be the same. It is because of this flexibility that the need arises for a local User's Guide. The purpose of this manual is to serve as an introduction to the M&DO computer systems and to supplement all other relevant documentation to be referenced. Because of its index structure, the user will be able to reference those sections pertinent to his needs. This is not a programming manual; where a user needs to learn a language or a system, it will be necessary to review the referenced documents.

It is intended that this User's Guide be complete (within its scope) and up to date. Periodic revisions will be made; however, it will be impossible to maintain the guide completely up to date. To fill the interim, refer to the M&DO 360 Computer Bulletins and the GSFC Computer Newsletter. Usually, pertinent information is also posted in the Programmer Assistance Center (PAC) and dispatch areas.

1.2 ORGANIZATION OF THIS DOCUMENT

This User's Guide has been developed, and will be maintained on magnetic tape through the use of the IBM MT/ST system (a Selectric typewriter with cassette tape units). It is organized in such a way as to permit updating without the need to renumber all of the preceding or succeeding pages; that is, the document is paginated according to two-digit subsections (e.g., 14.1-1, 14.1-2, etc.). Exception to this occurs in those places where the subsections are very short. In these cases, more than one two-digit subsection may occur on the same page, and the page numbers are relative to the entire section.

When the document is updated, the new or replacement pages will contain the date of revision at the top left corner, and any modified lines of text will be indicated by a vertical line in the outside margin.

To assist the user in locating a particular area of interest in the User's Guide, several facilities are offered:

1. Table of Contents

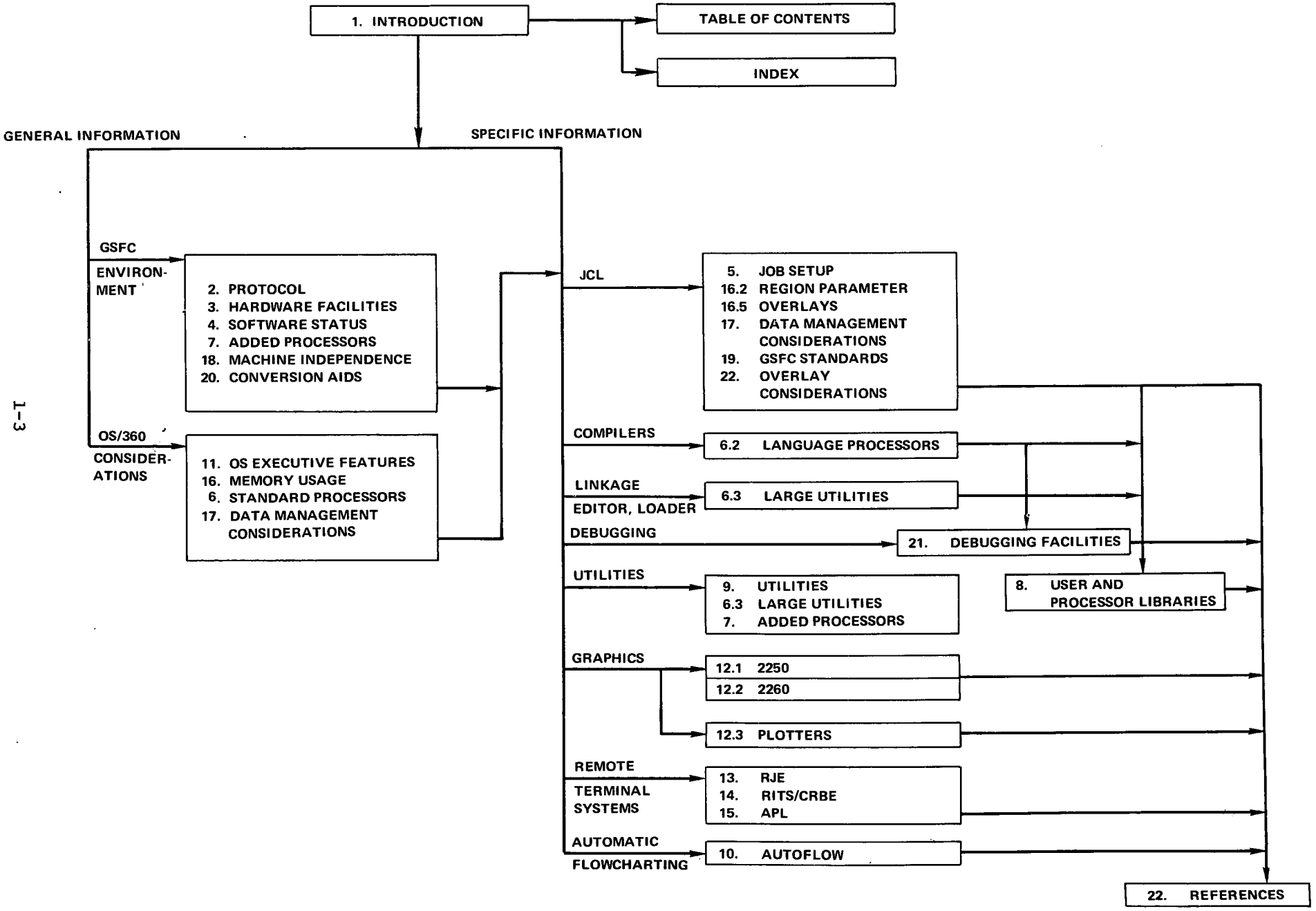
2. Index

3. Chart showing logical groupings of sections in the User's Guide and interrelationships between them (flyleaf)

In addition, the upper right hand corner of each page repeats the major section heading to which it belongs, to facilitate thumbing through the document.

Effort has been made, through redundancy and cross-referencing, to cover each area of interest in all contexts to which they are related. Finally, references to other sources for additional information are given throughout the text, and in Section 23.

M&DO IBM 360 User's Guide Reading Plan



SECTION 2

PROTOCOL

2.1 AUTHORIZATION FOR COMPUTER USE

2.1.1 GENERAL DISCUSSION

All programmers must use identifying accounting information when submitting jobs to be run on GSFC computers. This information must be punched into the JOB card (Subsection 5.2) and indicated on the Job Submission Slip (Subsection 5.1) when computer time is desired. A document (X541-69-379) issued by the GSFC Computer Services Section outlines the details for completing the registration form required of potential computer users. Copies can be obtained from Mr. Gerald R. Quigley, GSFC Building 3, Room 162, Extension 5791. The following paragraphs describe the accounting information required in the JOB card and Job Submission Slip.

2.1.2 PROGRAMMER IDENTIFICATION CODE (PROGRAMMER ID)

Format: ooiii

The first two characters identify the organization code; the last three are the programmer's initials. If the programmer has only two initials, or if another user has the same initials, the third character in the code will be a numeric digit.

2.1.3 SPONSOR NUMBER

Format: ssxxx

The first two characters identify the sponsor; i.e., the organization for whom the computer use has been authorized. The last three digits identify the specific problem being solved for that organization.

2.1.4 PROGRAM NUMBER

Format: pppppp

This six-character number identifies a particular program or system of programs, and is used by the GSFC Computer Program Library to register any program offered. Refer to paragraph 2.3.11 of this User's Guide for details on the function of the Library.

At the time a new programming effort is initiated, it is the responsibility of the programmer (or his sponsor) to obtain a program number from the Library, Extension 2186 or from a computer manager.

2.1.5 PROJECT NUMBER

Format: pppp

This four-digit code identifies the individual spacecraft or experiment with a given program. (Refer to Document X-541-69-379 for a detailed description).

2.1.6 CATEGORY CODE

Format: c

This one-digit code identifies the type of work being processed. The categories are described in detail in Document X-541-69-379 and are summarized below:

Code	Title
1	Scientific and Engineering
2	Data Reduction
3	Mission Control
4	Simulation
5	Administrative

2.1.7 TYPE OF RUN CODE

Format: t

This one-digit code identifies a run as test (T), production (P), or an extended code for rerun and downtime (refer to Document X-541-69-379 for a detailed description).

2.1.8 RITS AUTHORIZATION

Use of the Remote Input Terminal System (RITS), which provides remote access to the IBM 360/95 system, requires registration with the Computer Services Section, GSFC Building 3, Room 171, Extension 6781. An applicant must have a programmer ID and sponsor number and must attend the RITS seminar, or show a proficiency in RITS or CRBE before he will be authorized as a RITS user. After his application is approved, he may then use any of the terminals, identifying himself by his programmer ID, password, and sponsor number.

2.2 APPROVAL REQUIREMENTS FOR DEDICATED USE OF RESOURCES

2.2.1 GENERAL DISCUSSION

The IBM 360/95 is the M&DO general support computer. Although all systems are available to any authorized user, the Model 65 and Model 75 are devoted almost entirely to their assigned projects and have little time available.

All computer facilities are operated on a limited access basis. The responsibilities for scheduling, operation, and maintenance of this equipment and supporting services are assigned to specified personnel and provided as a service to authorized users. Except for certain designated locations, access to the areas occupied by the equipment and supporting functions is limited to operating and management personnel assigned the responsibility of operating the facility. Users of the computing service, i.e., programmers and experimenters, are allowed access to these facilities only when the nature of their requirement is such that it cannot be satisifed by the normal operational mode. In such a case, special arrangements for access must be made with the GSFC computer manager. Mr. Harry G. Bitting, Building 3, Room 130, Extension 6886, is the computer manager for the 360/95, Orbit 360/75 and 360/65. Unauthorized personnel are not permitted to operate any part or component of the computers or other support equipment, as designated personnel are assigned this task.

2.2.2 USER DATA SETS

The Model 95 has ten scratch packs (VOL=SER=G1SCR1 through VOL=SER=G1SCRA) permanently mounted for the storage of temporary user data sets. A temporary data set is one which exists only for use within a job and which will be deleted upon completion of the job. To assign a temporary name, code the keyword parameter DSN=&name or DSN=&&name in the DD card. Replace the term name with any 1- to 8-character name not used by another temporary data set in the job. To assign the data set to one of the scratch packs, code the keyword parameter UNIT=DISK, and the system will assign the data set to a scratch pack having the amount of requested space available. The DISP parameter on the DD card should be coded as DISP=(NEW,DELETE) or DISP=(NEW,PASS,DELETE) if the data set is to be used in a subsequent job step. The use of temporary data sets is strongly encouraged, since it frees space for use by other users after the job has been completed.

It is, of course, essential for users to have the capability of retaining their data sets on-line from one day to the next or for an indefinite period of time. The computer manager has made a provision for this by allocating on-line storage for user data sets on the Model 95. Currently, two 2316 disk packs (VOL=SER=G1USR1, VOL=SER=G1USR2) and one 3021 bin on the 2321 data cell (VOL=SER=G1USR3) are allocated for this purpose. Because of the limited nature of these resources, procedures have been established to control their use. The following is extracted from the memorandum dated June 8, 1970, from the computer manager of the IBM 360/95:

- User data sets must be named in accordance with the centerwide convention:

  xx.yyyyy.zzzzzzzz

  where:

  xx = System Designation (360/95 = G1)
  yyyyy = User Identification (e.g., G3GRQ)
  zzzzzzzz = Optional Identification (e.g., MYSET1)

  Data sets which do not conform to the above convention will be eliminated without further notice.

- All allocations of user storage areas will be made only with the knowledge and approval of the computer manager. All such requests should be forwarded to the computer manager with a written explanation of the action desired. Unapproved allocations will be scratched without prior notice to the owner of the data set.

- On the first working day of each month, users will receive a memorandum listing the data sets that are available on the user packs. Users must indicate the disposition of each data set and return the memorandum to the computer manager. Failure to return the memorandum within one week will result in the user's data sets being scratched.

2.2.3 PRIVATE VOLUMES

A request for a disk pack or data cell must be made in writing by the sponsor - not the individual programmer - and approved by the Chief of the Computation Division. The programmer presents the approved request to the Library, Building 3, Room 171, where the packs and cells are dispensed. Disk packs and data cells are issued with the manufacturer's number, a permanent GSFC number, and a temporary volume label. The label contains a code representing the organization, user, and job. The device is identified by this code until it is returned to the Library and re-issued under a different label. Once issued, the devices are stored in the vicinity of the computer room under controlled conditions. Requests for moving the device from one system to another will be honored, but prior permission should be obtained from the computer manager. A device may be removed from the Building 3 area by obtaining permission from the computer manager and by completing a "charge-out" card. These devices are accountable equipment and are the responsibility of the Computation Division.

2.2.4 RITS SPACE ALLOCATION

After receiving authorization to use RITS (see Paragraph 2.1.8), a user will have allocated one cylinder (20 tracks) of space. This space will be on one of two 2316 packs - either RITS04 or RITS08. Refer to Paragraph 2.3.3 for detailed information on RITS services and hours of operation.

2.2.5 APL SPACE ALLOCATION

Each user of the 360/95 APL system is given 10 work spaces in which to SAVE his functions or data. Each work space has a length of 36,000 bytes. Unlike the RITS or CRBE user libraries, the APL user work spaces are not partitioned data sets; rather, they are structured together internal to the APL system. They are not readily distinguishable by anyone other than the user of the work space, and then, only when the work space is being used through APL. A backup to the APL system (including a daily dump of user work spaces) is maintained in the event of APL disk failure.

2.2.6 DEDICATED PERIPHERALS

One of the prime functions of OS 360 on the M&DO computers is to utilize efficiently all available system resources in order to maximize throughput and minimize turnaround time. The moment that any peripheral device (e.g., disk drive, tape drive, printer) is dedicated to an individual user's job, the system operates much less efficiently. Hence, the use of dedicated peripherals is discouraged, and requests for the use of dedicated peripherals should only be submitted when required for a launch or other emergency circumstance. All such requests must be submitted in writing to the computer manager.

2.2.7 SCHEDULING AND PRIORITIES

Scheduling of the computers is under the control of the Computer Services Section, Computation Division. A schedule is issued weekly, indicating blocks of time for specific divisions and/or programmers. The normal processing of a job is accomplished without obtaining a priority and is run in log number order, within the scheduled block of time for "general support".

Generally, individual jobs with a computing time of one-half hour or more will be retained for the night shifts to process. The exception to this would be priority or programmer-present work, operating within the scheduled block time. For convenience and speed in processing, IBM 360/95 jobs may be batched together and loaded onto tape off-line, utilizing the IBM 360/30. This action eliminates much of the card handling and setup time that would be required to process individual jobs on the computer. Similarly, the processed output may be stacked on one or more tapes for listing off-line.

Once a job has been entered into an M&DO computer, an addition to the operating system, known as the Job Stream Manager, further classifies jobs and establishes

priorities. Each job is automatically assigned to one of 26 classes (A through Z) on the basis of the amount of core (REGION size) and the number of tape drives required by the job. The Job Stream Manager uses the estimated run time as the basis for the assignment of priorities, with shorter runs being processed first. On all three computers, the priority (PRTY=) and class (CLASS=) parameters on the JOB card are ignored if they are present.

2.2.8 USER ADDITION TO PROCEDURE LIBRARY

The use of cataloged procedures (refer to the IBM Job Control Language User's Guide (GC28-6703) and Job Control Language Reference (GC28-6704) manuals) can considerably simplify the execution of many programs. Due to space limitations, the following conditions must be met before procedures can be placed in the 360/95 user PROCLIB (DSNAME=SYS2.USERPROC):

a. The procedure must be necessary for the execution of a user program. Procedures for compilations, assemblies, or link edits may not be entered.

b. The procedure must be checked out.

c. The procedure must be at least 15 cards, or it can be any size if it is used at least five times per day.

d. A written request accompanied by a listing of the procedure and the deck necessary to update SYS2.USERPROC must be submitted to the IBM 360/95 computer manager before space allocation can be granted. Subsequent updates to previously approved procedures must also follow the above rules.

2.2.9 APPLICATION-ORIENTED SYSTEMS

There are many specialized programming systems in use by the M&DO which are utilized by particular groups of individuals, but which are not for use by the general public. These processors are programs or series of programs designed to perform a particular function and assist particular projects. The following are examples of such processors.

2.2.9.1 Computer-Assisted Interactive Resource Scheduling (CAIRS)

This system assists the Operations Center branch in the control, production, and scheduling of the tracking and data acquisition stations in support of scientific satellite missions. The CAIRS system is utilized continuously throughout the day, and hence is resident on two 2316 disk packs permanently mounted on the Model 95. CAIRS is written in Assembly Language and PL/1 and functions on a real-time basis from remote terminals. Users requiring more information about the CAIRS system should contact Mr. Carl Gustafson, Code 512, Extension 4939.

2.2.9.2 Inter-Processor Tasking and Communications (ITAC)

The Inter-Processor Tasking and Communications (ITAC) system is a recently developed communications link between the MFLT 75 and the System/360 Model 95, and the M&DO 75 and Model 95. ITAC consists of two control programs: 1) the Dispatcher executes in supervisor state on a Model 75; and 2) the Monitor executes in problem state on a Model 95. Each Model 75 is linked physically to the Model 95 through a channel to channel adapter connected to a selector channel on each computer. Through the ITAC system, a user's task in execution on one of the Models 75 may attach and subsequently communicate with a subtask on a Model 95. This facility will allow large quantities of data to be transferred between computers and will permit off-loading of compute bound work from the Models 75 to the Model 95. The operator on the Model 95 has a moderate degree of control over ITAC operation through his console. He may monitor ITAC subtask execution and terminate any or all subtasks. He may also terminate ITAC operation itself. Users requiring more information on ITAC should contact Mr. Ron Larson, Code 833, Extension 4127.

2.2.9.3 Definitive Orbit Determination System (DODS)

This system is designed to meet the orbit determination needs and to support tracking prediction operations associated with scientific and applications spacecraft programs of NASA and the space community. The primary purpose of DODS is to compute orbits of various satellites for surveillance and research at GSFC, and includes a number of data processing functions to make this computation as automatic and effortless as possible. DODS operates on either the Model 75 or Model 95. Users requiring more information on DODS should contact Mr. Paul Shapiro, Code 541, Extension 2589.

2.3 JOB SUBMISSION PROCEDURES AND OPERATIONS

2.3.1 DISPATCHING STATIONS

2.3.1.1 General Processing Flow

The initial step in processing a computer work unit for the Models 75 and 95 is the submission of a computer job to the Dispatch Station in Building 3, Room 103, Extension 6733. The jobs for the Model 65 are submitted to the Dispatch Station in Building 14, Room S4, Extension 2195. The submission must be accompanied by the proper forms and instructions, program and data decks, if required, and all tape files which are not in the tape library associated with the operation. The work must be identified by an authorized programmer identification code.

Personnel at the Dispatch Station will examine the submission to determine that required forms, instructions, and data are provided. If the submission is determined to be incomplete, it will be returned to the requester immediately; if it is complete, it will be logged and processed. After processing, the appropriate input and output items are placed in the user's output box.

2.3.1.2 Problem Report

If it is felt by the programmer that his work has not been properly processed, a Problem Report form is available at the Programmer Assistance Center, Building 3, Room 133A, Extension 6768, (see Paragraph 2.3.10.1) to indicate the problem. The Problem Report form is self-explanatory.

2.3.1.3 Messenger Service

Users need not go directly to the Dispatch Stations to submit jobs. GSFC maintains a messenger service which will pick up and deliver jobs to remote locations, both at GSFC and off-site. The locations serviced and hours of delivery are posted at the dispatcher's office, or they may be obtained by calling Extension 6733.

2.3.2 REMOTE JOB ENTRY (RJE)

2.3.2.1 Job Submission

RJE services are normally available on the IBM 360/95 between the hours of 8 A.M. to 8 P.M., Monday through Friday. RJE may not be available if the 360/95 is being used to support a launch. Should a user have need to use RJE at hours other than the above, he may call the 360 operator at Extension 5395.

2.3.2.2 Output Routing

The output from a job submitted via RJE may be directed to the user's work station, to an alternate user, or to the system output writer for delivery through the Dispatch Station. Refer to Section 13 of this User's Guide for detailed information on the capabilities and uses of RJE.

2.3.3 REMOTE INPUT TERMINAL SYSTEM (RITS)

2.3.3.1 Hours of Service

RITS services are available on the IBM 360/95 from 8 A.M. to 8 P.M., Monday through Friday. RITS may not be available if the 360/95 is being used to support a launch. Should the user desire to use RITS at hours other than the above, he may call the 360 operator at Extension 5395.

2.3.3.2 Output

The printed results of a run can be obtained either from the 1403 printers located at the 360/95 (SYSOUT=A), at the user terminal (SYSOUT=R), or a combination of the two. Consult Section 14 of this User's Guide and the RITS User's Guide for complete details.

2.3.4 RITS/RJE TAPE MOUNTS

Jobs submitted to the IBM 360/95 via RITS or RJE which require the mounting of specific tapes cannot be processed unless these tapes are readily available in the machine area. In order to insure proper handling of such jobs, the user must complete the following:

a. Visit or call the tape library, Extension 6735 at least one-half hour in advance and ask that the tapes be placed in the machine room giving the dispatcher the tape numbers and specifying rings in or out, as desired. (If the tapes are in storage, several days notice is required.)

b. Include the tape numbers in the VOLUME=SER= fields of the appropriate DD statements of the submitted job.

If output tapes are to be retained, save a tape in advance (see the dispatcher or tape librarian and follow the same procedure).

Jobs which call for specific tape mounts will be CANCELED if the required tapes are not in the machine room. Scratch tapes may be used without notifying the dispatcher. When a job requires the use of a scratch tape, simply

omit the VOLUME parameter from the DD card. For jobs requiring more than one scratch unit, refer to Section 17.

Private volume disk packs or data cells may be referenced only if it is known that the required volumes are already mounted according to a predetermined schedule. RITS or RJE initiated mount requests for private volume direct access devices will not be honored.

Jobs submitted through the dispatcher that request the mounting of private volumes will be held until the volumes are scheduled to be mounted.

2.3.5 TAPE LIBRARY

The Magnetic Tape Library is maintained to issue, receive, and release magnetic tapes as requested or required. The tapes handled by the library are classified into one of three categories:

a. Utility Tapes (blanks)

b. Permanent Save Tapes

c. Storage Tapes (as requested from central storage)

The various jobs submitted for processing may require a tape from the tape library as input, and the resultant output may need to be filed. A Permanent Save Tape Label is submitted by the user/programmer to the tape library with the job and contains all the pertinent information concerning the tape, with the exception of the tape number, which is supplied by the tape librarian or the operator. This card is in two parts: one part is applied to the tape as a label, the other is sent to the Keypunch Section for punching into a permanent file card. Normally, magnetic tapes are obtained from the library by the operator or dispatcher as the work is being prepared for the computers. Remote terminal users must telephone the tape library to request that a tape be placed in the appropriate computer area prior to submitting a job calling for that tape.

Programmers and other personnel authorized to remove tapes from the library may do so by indicating such desire to the librarian who will, in turn, indicate on the tape file card that the tape has been removed and its destination.

Magnetic tapes that have not been requested for use for a 60-day period will be sent to permanent storage. This action is necessary to relinquish rack space, and make available new tapes for use. A four- to five-day notice is required for retrieval of tapes sent to storage. Storage tapes that are not used within 15 days after their arrival in the tape library will be returned to central storage.

A list of tapes assigned to users is generated at regular intervals and is forwarded to each person to whom magnetic tapes are assigned. Tapes that may be released or sent to permanent storage should be indicated by the user on this listing and returned to the librarian for action.

2.3.6 USE OF PRIVATE VOLUMES

During daytime operations (8 A.M. to 8 P.M.), 20 of the 24 disk drives available on the IBM 360/95 have permanently mounted packs. The remaining four are normally reserved for private user packs, such as ATSPAK, ATTDET, DODS01, DODS02, DOTTE1 and DOTTE2. The mounting of private disk packs and data cells could reduce the efficiency of normal operations and is not encouraged.

2.3.7 PLOTTER OUTPUTS

There are several types of plotters available at GSFC for use by qualified personnel. Programs run with the appropriate JCL will produce output tapes which may then be used on the desired plotter to furnish the necessary plots. Plotters are discussed more fully in Paragraph 2.4.6.

2.3.8 EAM AND RELATED SERVICES

Computing and Software, Inc. (C&SI), a GSFC contractor, is responsible for computer operations, dispatch service, the tape library, keypunching, and all EAM related services.

2.3.8.1 Card and Tape Processing Services

Two computers, an IBM 360/20 and 360/30, are available around-the-clock for card and tape processing services. The user need not prepare any control or JOB cards, but must complete a request slip (available in the Dispatch Station Building 3, Room 103, Extension 6733) indicating the service he wants performed and the density, mode, and DCB information for any tapes being used.

The Model 20 services provided include sorting, sequencing, interpreting, duplicating (up to four copies at once), reformatting, and printing card decks. It can also be used to convert 026 (BCD) punched cards to 029 (EBCDIC) cards and vice versa.

The Model 30 services provided include tape-to-print, tape-to-tape, tape-to-card, card-to-tape, and dumping a 7- or 9-track tape of any standard mode with 200, 556, or 800 BPI density. Nine-track tapes cannot be copied to another 9-track tape, as only one 9-track tape drive is available. Seven-track tapes may be copies to either 7- or 9-track tapes. The Model 30 can be used for both blocked and unblocked tapes, but users must indicate the block size and logical record length on the request slip.

2.3.8.2 Keypunching Services

A variety of keypunching services are provided to GSFC personnel in support of work directed to and through the large-scale computers. All work is keypunched and verified in sequential order, unless a priority or express run is specified.

All work submitted for keypunching must be logged through the central dispatcher. A log number will be assigned to the job and the job will then be forwarded to the Keypunch Operation. Completed work will be returned to the Dispatch area, where it may be retrieved by the submitter after it has been logged out.

An express submittal is defined as a job containing 25 cards or less to be keypunched and/or verified. A priority submittal can be of any length and the priority assigned is a function of the importance of the job. This priority may come from the Computer Services Section or from the C&SI Supervisor of the Operations Department. All keypunching work is verified as a standard practice. Normally, this verification is accomplished by an operator other than the one who has keypunched the job.

All work submitted for keypunching must be coded on standard coding forms, such as those for FORTRAN, Assembler Language, or Eighty Column General Purpose. All entries on the coding sheets must be printed. Handwritten requests will be returned without being punched. Submitters are requested to erase any errors thoroughly - do not scratch through. Proper columns and spacing should be indicated. Requests for rearrangement of cards cannot be accommodated. All jobs will be keypunched and returned in the same order as they appear on the coding sheets. The EAM work request has provisions for indicating a sorting operation, and jobs requiring a rearrangement of cards may be handled in this way.

To avoid confusion, the following conventions have been established in coding forms at GSFC:

LETTERS: I, O, Z
NUMBERS: 1, Ø, 2

2.3.8.3 Notes on IBM 026/029 Card Punches

There are two types of keypunches available to users: the IBM 026 Printing Card Punch and the IBM 029 Card Punch. The IBM 026 is controlled by a program card and by the keyboard switches and keys. The program card controls automatic skipping and duplicating, field definition, and alphabetic shifting. The IBM 029 has a two-program card which can be alternated at will either between punched cards or in the course of punching a single card. The IBM 026 has a 48-character keyboard for punching data in BCD format. The 64-character keyboard on the IBM 029 is intended for punching data in EBCDIC. The card codes produced by these keypunches are discussed in paragraph 3.6.4.3.

2.3.8.4 Report Finishing

Report finishing includes the decollating, reproducing, and binding of computer printouts. This service is available 24 hours per day. The decollator separates multiple-ply printouts from their carbons and refolds them individually. The removal of margins is optional. Since only the first four copies of computer printouts are legible, copies in excess of four are made on the Xerox 2400. Document binding may take one of two forms. A continuous listing is glued along the top edge and secured between front and back covers. Side-edge glueing is feasible only with single-sheet listings reproduced on the 2400.

2.3.9 AUTOFLOW SERVICES

AUTOFLOW is a proprietary software system of Applied Data Research, Inc., that automatically translates the source language of a program into flowcharts, and then prints them out on the printer. The programmer may choose to have his flowchart put on an output tape for later use.

AUTOFLOW can assist in debugging a program by means of a flowchart in the early stages of the program, or can provide final documentation. Accompanying the flowcharts are tables of contents and cross references; tables of diagnostic messages pointing out program-logic errors, syntax errors, missing references, etc.; and an optional listing of the source program.

In addition, AUTOFLOW has special features that enable the programmer to adjust details in the flowcharts by means of additional coding. Since AUTOFLOW makes use of program comments in the flowcharts, generous use of comments in the program results in more meaningful flowcharts.

AUTOFLOW accepts as inputs either decks or tapes in COBOL, FORTRAN, PL/I, or assembly language for the S/360, and assembly language or FORTRAN decks or tapes for CDC 3200, DDP 224, XDS 930, and Univac 1108 computers. AUTOFLOW, and its preprocessor for computers other than the S/360, follows all the rules in the programming manuals of the particular computer being flowcharted.

See Section 10 for a more detailed description of AUTOFLOW.

2.3.10 USER ASSISTANCE

Goddard Space Flight Center utilizes both Government employees and contractor personnel to provide individual programmers and users with the best assistance possible.

2.3.10.1 Programmer Assistance Center (PAC)

Three senior programmers are on duty in the Programmer Assistance Center, Building 3, Room 133A, Extension 6768, from 8 A.M. to 4:30 P.M., Monday through

Friday, to provide users with error analysis and correction. Users will be requested to complete a Computer System Problem Report while one of the programmers examines the user's problem. To aid the programmer, users should request a SYSUDUMP dump for the job steps which terminate abnormally.

The personnel in the PAC will also provide operational assistance for users unfamiliar with the 360 Operating System, Job Control Language, or GSFC standards and procedures.

2.3.10.2 RITS/APL Assistance

Users with problems related to RITS or APL should contact the PAC.

2.3.10.3 GSFC Systems Programmer Assistance

The GSFC systems programmers are not available for general assistance. All matters related to programming and system difficulties should be directed to the programmers in the PAC. If the individuals on duty are unable to resolve the question satisfactorily, they will consult with the GSFC Technical Representative, who will determine if a GSFC systems programmer should be consulted.

2.3.10.4 Field Engineer Assistance

No user, whether a Goddard or a contractor employee, may contact the Field Engineers directly. Any problem which is caused by a hardware failure must be brought to the attention of the computer manager or the PAC programmers.

2.3.11 PROGRAM LIBRARY SERVICES

The Goddard Space Flight Center has established a Computer Program Library, a repository for computer programs and related documentation generated by and for GSFC. As such, the GSFC Computer Program Library is a center for the collection, storage, and retrieval of all computer programs, systems, subroutines, and their attendant documentation. This library also has access to programs and documentation from sources outside the Center. In establishing this library, the GSFC has several objectives:

- Avoid duplication of effort
- Reduce programming time
- Reduce programming cost

A means toward achieving these objectives is to make previously programmed material more widely available. Although previous efforts may not have fulfilled current needs, access to techniques employed by other programmers is

often helpful in shortening the time required for new developmental activities.

GSFC Report X-540-69-107 describes the functions and activities of the Program Library in greater detail. A copy of this may be obtained from Mrs. Pat Barnes, Code 543, Extension 6796.

2.3.12 GSFC MANUALS LIBRARY

2.3.12.1 Location, Hours of Service, and Approval Requirements

GSFC maintains a manuals library where Government personnel may obtain copies of computer manufacturers' manuals and related forms. This service is also available to contract personnel when their contract stipulates that GSFC will supply computer manuals. Manuals may be obtained by completing a Manual Request form and obtaining the approval of the responsible Division Chief.

The Library is located in Building 16 Annex, Room 115, which is most easily reached from the rear entrance (the end of the Annex that faces Building 16). Library hours are 8 A.M. to 4:30 P.M. daily. Requisitions will be filled from stock on the shelves or will be ordered. Users may telephone the Library, Extension 4672, to obtain additional information.

2.3.12.2 Types of Manuals Stocked

In general, supplies include those items needed by users of Goddard computers, and include the following types of publications:

- Manuals on IBM S/360 computers
- Manuals on IBM 7090/7094 computers
- IBM coding forms
- Other items, such as the SD4060, CalComp and Gerber plotter manuals.

The Librarian maintains a current bibliography of manuals at the desk, so that users can check titles and current revision dates of manuals requested.

2.4 OTHER FACILITIES AT THE GODDARD SPACE FLIGHT CENTER

2.4.1 SPACE AND EARTH SCIENCES DIRECTORATE (SESD) IBM 360/91K

The SESD 360/91K computer is located in the basement of Building 1. The 360/91K is currently operating under Release 18 of the IBM S/360 Operating System with MVT. It has on-line graphics, and serves a number of RITS and RJE users. This computer has the regular assembler and compilers, plus some of the special programs available on the SESD Model 75J. (NOTE: The Model 91 has no decimal arithmetic feature; consequently, all decimal-instruction programs must be directed to the Model 75.) For further information concerning the SESD 360/91 or SESD 360/75 consult the SESD User's Guide.

2.4.2 SESD 360/75J

The SESD 360/75J computer is located on the second floor of Building 21. The 360/75J is currently operating under Release 18 of the IBM S/360 Operating System with MVT. It has on-line graphics, and serves several remote terminals (CRBE) both at Goddard and off-site. This computer has in its system library a number of special programs of general interest, in addition to the regular assembler and compilers.

2.4.3 MFLT 360/75

2.4.3.1 Location

The IBM 360/75 (MFLT) computer, located in Building 14, Room 100, is the primary support system for the manned flight project. Its system hardware and on-line and off-line peripheral devices are described below.

2.4.3.2 Operations

The Computer Manager for the Model 75 (MFLT) is Mr. Calvin A. Packard, Building 3, Room 172B, Extension 4022. The system programmers are Mr. John Morton and Mr. Fred King, Building 14, Room E140B, Extension 4848. Questions regarding the Model 75 (MFLT) may be directed to them or their staff.

2.4.3.3 Hardware Configuration

The system components for the Model 75 (MFLT) are as follows (additional hardware to be added October 2, 1970, will be reflected in the next update to this section):

a. A Model 2075-I Processing Unit (equipped with the standard, decimal, and floating point instruction sets)

b. Two Model 2365-3 Processor Storage units, with 1024K bytes of high-speed storage

c. One Model 2361-1 Core Storage with 1024K bytes of LCS

d. One 2860-2 Selector Channel with:

   1. One 2314-1 Direct Access Storage Facility, with a capacity of 233,408K bytes of storage

   2. One 2321-1 Data Cell containing 400,000K bytes of storage

   3. One 2303-1 Drum Storage device, with a capacity of 3900K bytes

e. One 2870-1 Multiplexer Channel with three selector subchannels with:

   1. Two switchable 2250-1 Display Units

   2. One hard wired 2250-1 Display Unit

   3. Eight 2401-6 9-track tape drives (the 4 drives 0D1-0D4 are switchable with the Model 75 (ORBIT))

   4. Two 2401-3 7-track tape drives (drive 0D0 is switchable with the Model 75 (ORBIT))

   5. Five 1403-N1 Printers (3 are shared with the Model 75 (ORBIT))

   6. Two 2540-1 Card Read Punches (one is switchable with the Model 75 (ORBIT))

f. One 2150/1052 Typewriter Console

g. Various communication channels for interfacing with the Model 95, the Univac 494, and IBM 1050 remote terminals.

2.4.3.4 Software

The Model 75 (MFLT) is currently operating under Release 18 of the IBM S/360 Operating System with MVT.

2.4.4 OTHER SUPPORTING SERVICES

Supporting services are available and include:

1. IBM S/360, Model 30 computer in Building 26, used for off-line support of the SESD 7094.

2. IBM S/360, Model 20 computer in Building 21, used for off-line support of the SESD 360/75J. It is used for such operations as card-to-printer listing, 026-to-029 conversion, reproduction, sequencing, gang punching, and sorting.

3. Keypunching service.

4. Model 029 keypunch machines for self-service, in Buildings 21 and 26.

In rooms near the SESD 360/91K computer facility, the following support services are available:

1. IBM S/360, Model 20 computer for off-line support to the SESD 360/91K. It is used for such operations as card-to-printer listing, 026-to-029 conversion, reproduction, interpretation, sequencing, gang punching, and sorting.

2. Keypunching service.

3. Model 029, keypunch machines for self-service.

More detailed information on these services may be found in the SESD User's Guide.

2.4.5 OTHER COMPUTERS

Other "category A" (general purpose) computers available for use at the Goddard Space Flight Center include:

1. The SESD IBM 7094-II computer. With 65K words of core and 1301-2 disk capability, this computer is located in Building 26 and intended primarily for use by the National Space Science Data Center personnel. More detailed information may be found in the SESD User's Guide.

2. The Information Processing Division's Univac 1108, which has 196,000 word memory, 512,000 words of fast drum storage, 419,200 words of medium-speed drum storage, and 44,000,000 words of low-speed drum storage. This computer, located in Building 23, has two CPU's sharing the resources, which include 33 IBM-compatible 7-track tape drives. The Univac 1108 supports various satellite projects, including the Orbiting Geophysical Observatory (OGO), Orbiting Solar Observatory (OSO), Applications Technology Satellite (ATS), and Orbiting Astronomical Observatory (OAO) projects. More detailed information on the Univac 1108 may be obtained from Mr. Eugene Grunby, Extension 6428 or Mr. Mike Mahoney, Extension 6028.

2.4.6 PLOTTERS

2.4.6.1 Stromberg - Carlson 4020 Plotter (SC 4020)

The SC 4020 is a cathode-ray-tube microfilm plotter having options for 35 mm or 16 mm microfilm, or 7" x 7" hardcopy. It is no longer available at Goddard; having been replaced by the Stromberg-Datagraphics 4060 Plotter.

2.4.6.2 Stromberg - Datagraphics (Formerly Stromberg - Carlson) 4060 Plotter

The SD 4060 Stored Program Recording System represents an improvement over the SC 4020 unit described in the previous paragraph. It is also located in Building 23, and plot tapes are submitted in the same manner.

The 4060 system contains a Product Control Unit, which is a modified Honeywell DDP 516 computer that controls information flow and formats the output. This results in greater flexibility and higher operating speed. The major advantages of the SD 4060 over the SC 4020 are:

1. Hard copy is made available through the xerox copyflow process, which allows the production of 8½" x 11", 11" x 14", and strip chart.

2. The graphics portion of a 4060 job takes only 25 percent as much execution time as a comparable 4020 job.

3. The resolution of the 4060 output is 12.5 times greater than that of the 4020 output.

Additional information about the SD 4060 may be obtained from the Universal 4060 System and Software Manual available through Mr. Don Kennedy, Extension 6346, or Mr. George Fleming, Extension 5129, both in Building 23.

Software packages for generating SD 4060 plot tapes on the M&DO S/360 computers are described in Section 12 of this document. Tapes generated for the SC 4020 may also be used on the SD 4060. However, this is inefficient since it makes the 4060 simulate the 4020 and this procedure is not encouraged.

2.4.6.3 CalComp 570 Plotting System

The CalComp Model 570 Magnetic Tape Plotting System is located in the S/360-91 computer room in the basement of Building 1. This equipment consists of a Model 565R digital recorder, a tape transport, and a tape control unit. The Model 565R digital recorder is a drum plotter with a plotting surface of 10 inches by 120 feet. The pen moves in .01 inch steps, with a maximum speed of 300 steps per second.

To use the system, a seven-track 200 BPI tape is produced on the computer. There are a number of program packages which facilitate the production of the plotting tape. Those packages available on T&DS computers are described in Section 12. For further information, refer to the CalComp Digital Recorder User's Manual, prepared by Computer Sciences Corporation, dated January 1967, with Update Packages A, B, C, and D. An operating manual is available at the plotter. For additional assistance, see personnel in Building 1, Room 159, Extension 5436.

2.4.6.4 CalComp 770 Plotting System

The CalComp Model 770 Magnetic Tape Plotting System is located in Building 21, Room 273, Extension 6277. This equipment consists of a Model 763 digital recorder, a tape transport, and a tape control unit. The Model 763 digital recorder is a drum plotter, similar to the 565R unit used with the 570 System, but with several added features--the plotting surface is 30 inches by 120 feet, and the plotter is capable of operating in the "ZIP" mode. This mode is used to reduce plotting time when the plot consists of long, smooth lines without abrupt changes in direction.

The 770 tape transport accepts 7-track tapes recorded at 200 BPI or 556 BPI. These tapes may be prepared on IBM 360 computers through the use of the 770/780 plotter packages, as described in Section 12. For further information, refer to the CalComp Digital Recorder User's Manual.

SECTION 3

M&DO HARDWARE FACILITIES

3.1 GENERAL DISCUSSION

This section discusses, in general terms, the Operating System (OS) used on the 360 computers at Goddard. Later sections discuss specific machines and hardware characteristics. Refer to the Computation Division ADP Equipment Guide for greater detail on characteristics of peripheral and hardware components.

3.1.1 OPERATING SYSTEM

The Models 95, 75, and 65 operate under a program package referred to as OS/MVT (Operating System with Multiprogramming with a Variable Number of Tasks). Multiprogramming is that operation of a processor which permits programs to be executed in such a way that none of the programs needs be completed before another is started or continued. This capability is directed toward minimizing periods of idleness in any one part of the system. The scheduling program assigns one activity to the central processor while other activities are awaiting the completion of the input/output operations, and these activities are executed in such a way that those components that can function simultaneously are put to the fullest possible use. With MVT, the processor can service up to 15 jobs, each containing several tasks.

The operating system consists of a control program and a variety of processing programs. The latter include language translators, service programs, and user-problem programs. The exact composition of the operating system is flexible. All OS programs are stored in libraries on direct-access devices. Those basic to the system remain in memory; infrequently-used programs are stored on-line on direct-access storage devices and copies brought forward as needed. Furthermore, user-designed programs can be incorporated into the operating system for the duration of a single job or they may be stored in libraries and remain a part of the system for an extended period of time.

3.1.2 CONTROL PROGRAM

The control program directs the order in which jobs are processed, the work flow within the system, and input/output operations. It has three major parts: the job scheduler, the master scheduler, and the supervisor. These parts remain in core indefinitely. In addition to these, a group of supplementary routines is also available. These are brought into core from auxiliary storage, as needed.

The job scheduler reads job definitions from input/output devices; allocates input/output devices to each job; initiates the execution of the processing program specified for each job; processes selected output produced during each job; and provides records of work processed. In addition to performing these standard functions, the job schedulers for Goddard's 360 systems are uniquely tailored to classify jobs and establish their priorities. Refer to Paragraph 2.2.7 of this User's Guide for further details on the job schedulers.

The master scheduler is a two-way communication link between the operator and the system. The operator can issue commands to the master scheduler, alerting the system to a change in the status of the input/output devices; altering the operation of the system; and requesting information on the status of the system. Goddard's 360 systems provide extended communication with the operator: the master scheduler keeps the operator informed of where jobs are coming from and where they are going.

The supervisor is the control center of the operating system. It provides a number of services for other parts of the system, either in response to a specific request (e.g., request for storage space) or in response to some contingency (e.g., hardware malfunction). These services include allocating main storage space required by programs during their execution; sharing areas of main storage among routines that need not be in main storage at the same time; loading programs into main storage; controlling the concurrent execution of programs and routines; scheduling and controlling input/output operations; providing the time of day and other timing services; and providing standard procedures that assist in diagnosing exceptional conditions.

3.2 M&DO IBM 360/95

3.2.1 LOCATION

The M&DO Model 95 computer is located in Building 3, Room 153. Its systems hardware and on-line and off-line peripheral devices are described below. Refer to the Computation Division ADP Equipment Guide for more detailed information on the system and peripheral hardware.

3.2.2 SCHEDULING AND OPERATIONS

The Computer Manager for the M&DO 360/95 is Mr. Harry G. Bitting, Building 3, Room 130, Extension 6886. The Systems Programmers are Mr. Curtiss Barrett, Building 3, Room 127, Extension 6798, and Mr. Frank Pajerski, Jr., Building 3, Room 133E, Extension 6710.

Users may submit jobs to the Model 95 through the Dispatch Station, Building 3, Room 103, Extension 6733, or through remote terminals.

The systems programmers responsible for the remote terminal systems such as RITS (Remote Input Terminal System) and APL (A Programming Language) are Mr. Harry E. Crispell and Mrs. G. May Wilson both located in Building 3, Room 129, Extension 6797.

3.2.3 HARDWARE CONFIGURATION

As shown in Figure 3.2-1:

a. Model 2095J Central Processing Unit (CPU). This unit addresses main storage, performs logical and arithmetic functions, and initiates communication between main storage and external devices. The CPU has a basic machine cycle time of 60 nanoseconds.

b. Model M-120-J Processor Storage. This has 1024K bytes of thin-film memory. It operates at a basic machine cycle time of 120 nanoseconds.

c. Model 2395-2 Core Storage. This has 4096K bytes of high-speed core, with a basic cycle time of 750 nanoseconds.

d. Model 2150/1052 typewriter console.

e. One 2860-1 Selector Channel with a 2314-A1 Direct-Access Storage Facility attached. The 2314-A1 has a capacity of 233,408K bytes.

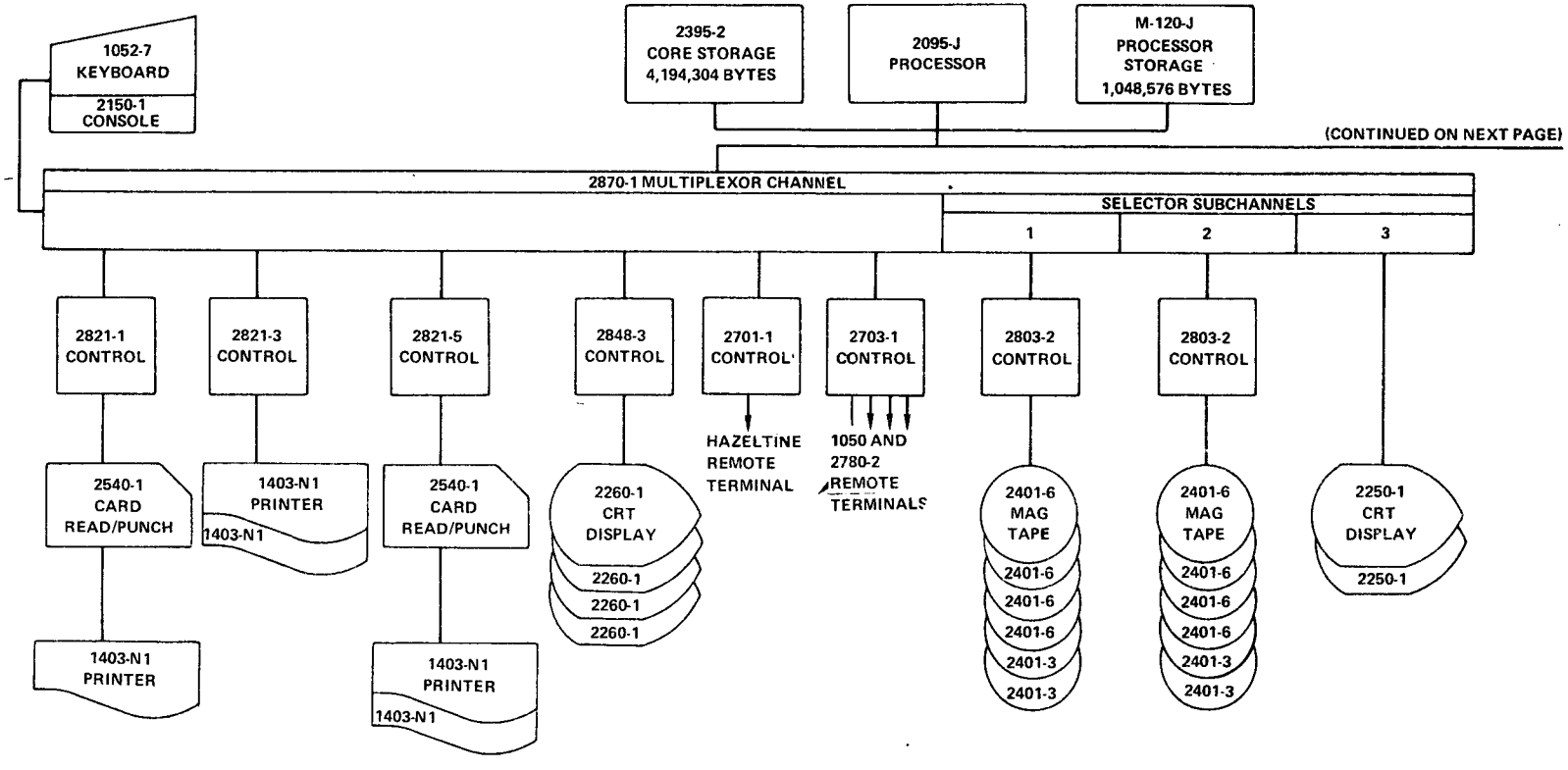


Figure 3.2-1. Equipment Configuration: IBM 360 System, Model 95 June 1970
(Sheet 1 of 2)

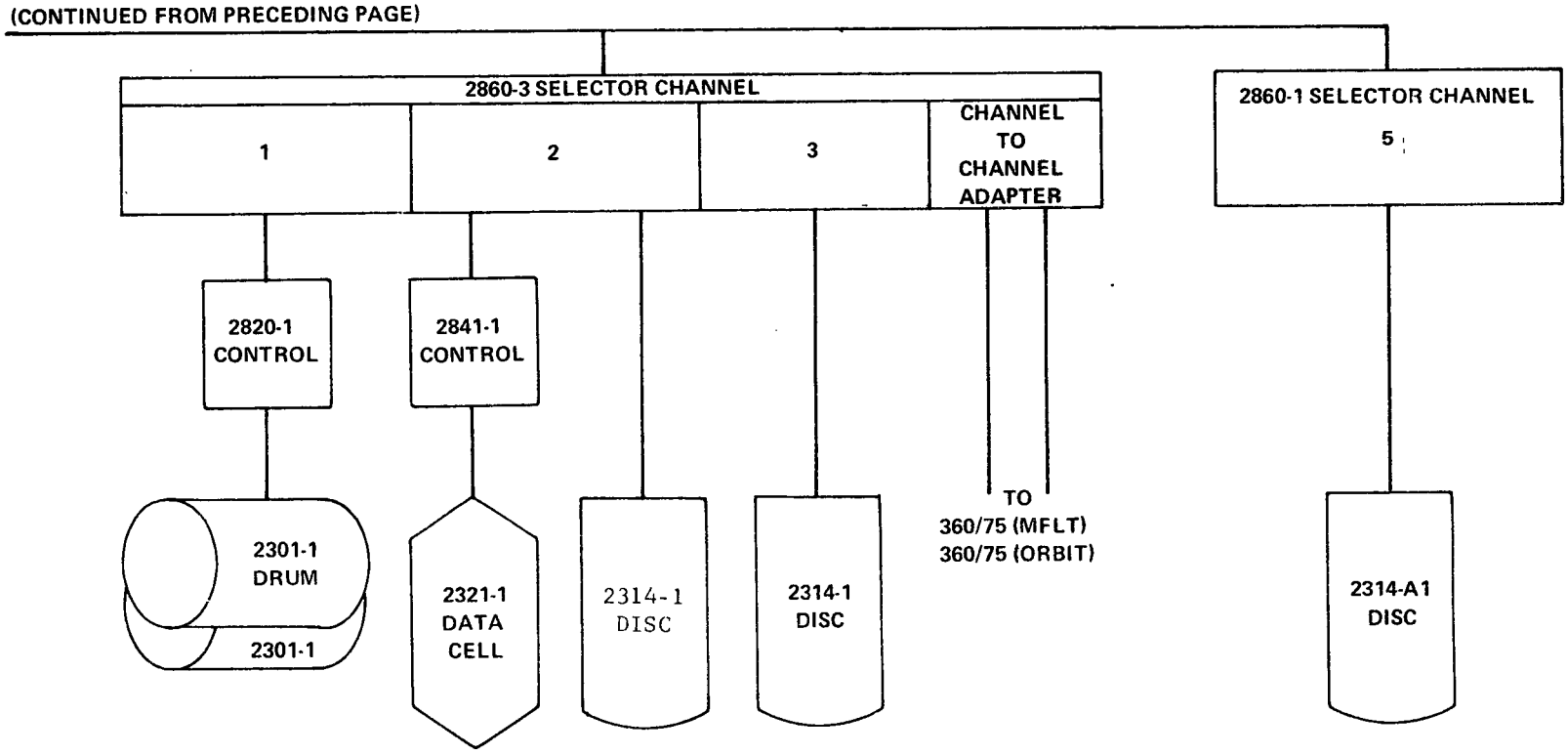


Figure 3.2-1. Equipment Configuration: IBM 360 System, Model 95 September 1970
(Sheet 2 of 2)

f. Three 2860-3 Selector Channels with:

   1. Two 2314-1 Direct Access Storage Facilities, containing 233,408K bytes each.

   2. One 2321-1 Data Cell, containing 400,000K bytes.

   3. Two 2301-1 Drum Storage Units, containing 4,000K bytes each.

   4. A Channel-to-Channel adapter which can be used to tie the Model 95 together with the 360/75 (ORBIT) or 360/75 (MFLT).

g. One 2870-1 Multiplexor Channel, including three selector subchannels with:

   1. Two 2250-1 Graphic Display Units

   2. Eight 2401-6 9-track tape drives

   3. Four 2401-3 7-track tape drives

   4. Two 2540-1 Card Read Punches

   5. Five 1403-N1 Printers with an HN Print Chain

   6. One 2701-1 Data Adapter Unit

   7. One 2703-1 Transmission Control Unit for attaching IBM 1050 and 2780-2 remote terminals to telephone lines

   8. Four 2260-1 Alphameric CRT Displays

3.2.4 UNIT ADDRESS

The following are the device addresses for units attached to the Model 95. They are presented here for the reader's edification; however, unit addresses should never be used in the UNIT field on DD cards.

Device Type	Address
2150/1052	*01F
2314-A1	330-337
2314-1	230-237, 540-547
2321-1	2E3
2301-1	1C0, 1C1
2250-1	*0E7, 0E0-0E1
2401-6	0C2-0C5, 0D2-0D5
2401-3	0C0, 0C1, 0D0, 0D1
2540-1	00C, 01C, 00D, 01D
1403-N1	001-003, 00E, 00F
2260-1	0A0-0A3

*Reserved for operators consoles.

3.2.5 VOLUME SERIAL NUMBERS

In the volume label of each Direct-Access Storage Device (DASD) is a unique serial number, referenced by the keyword parameter VOL=SER=volume in the DD card. The following is a list of the serial numbers assigned to the DASDs normally mounted on the Model 95:

Device Type	Use	Volume Serial Number
2314	Work or scratch packs	G1SCR1 through G1SCRA
2314	Storage of user data sets	G1USR1, G1USR2
2321	Storage of user data sets	G1USR3
2301	Storage of system and user libraries and data sets	G1DRM1, G1DRM2
2314	Storage of system and user libraries and data sets	G1SYS1, G1SYS2
2314	Used by the RITS and the APL system	RITS04, RITS08 APLR19

The other seven packs are private user packs, such as CAIRS1, DODS packs, etc.

3.2.6 SOFTWARE

The 360/95 is currently operating under Release 19.6 of the IBM S/360 Operating System with MVT. The Model 95 is not equipped with a decimal instruction set and can only simulate these instructions through software techniques. Since the simulation is extremely slow, lengthy decimal operations should be performed on Models 65 or 75, which are equipped with the decimal instruction set.

3.3 M&DO IBM 360/75 (ORBIT)

3.3.1 LOCATION

The Model 75 (ORBIT) is located in Building 14, Room 100. It is used primarily to perform orbit determination work and to support the manned flight project. However, when the work load warrants it, general purpose jobs will be run when so indicated by the user on his submittal form. Its systems hardware and on-line and off-line peripheral devices are described below. Refer to the Computation Division ADP Equipment Guide for more detailed information on the system and peripheral hardware.

3.3.2 SCHEDULING AND OPERATIONS

The Computer Manager for the M&DO 360/75 is Mr. Harry G. Bitting, Building 3, Room 130, Extension 6886. The Systems Programmers are Mr. Frank G. Ross, Building 3, Room 127, Extension 6798, and Mr. Herbert Durbeck, Building 3, Room 133E, Extension 6710.

Users may submit jobs to the Model 75 through the Dispatch Station in Building 3.

3.3.3 HARDWARE CONFIGURATION

As shown in Figure 3.3-1, the system components are:

a. Model 2075-I CPU, with a basic machine cycle time of 195 nanoseconds (equipped with the standard, decimal, and floating point instruction sets).

b. Two Model 2365-3 Processor Storage Units with a capacity of 512K bytes, each of high-speed storage, and a basic machine cycle time of 750 nanoseconds.

c. One Model 2361-1 Core Storage unit, with a capacity of 1024K bytes of Large Core Storage (LCS), and a cycle time of 8 microseconds.

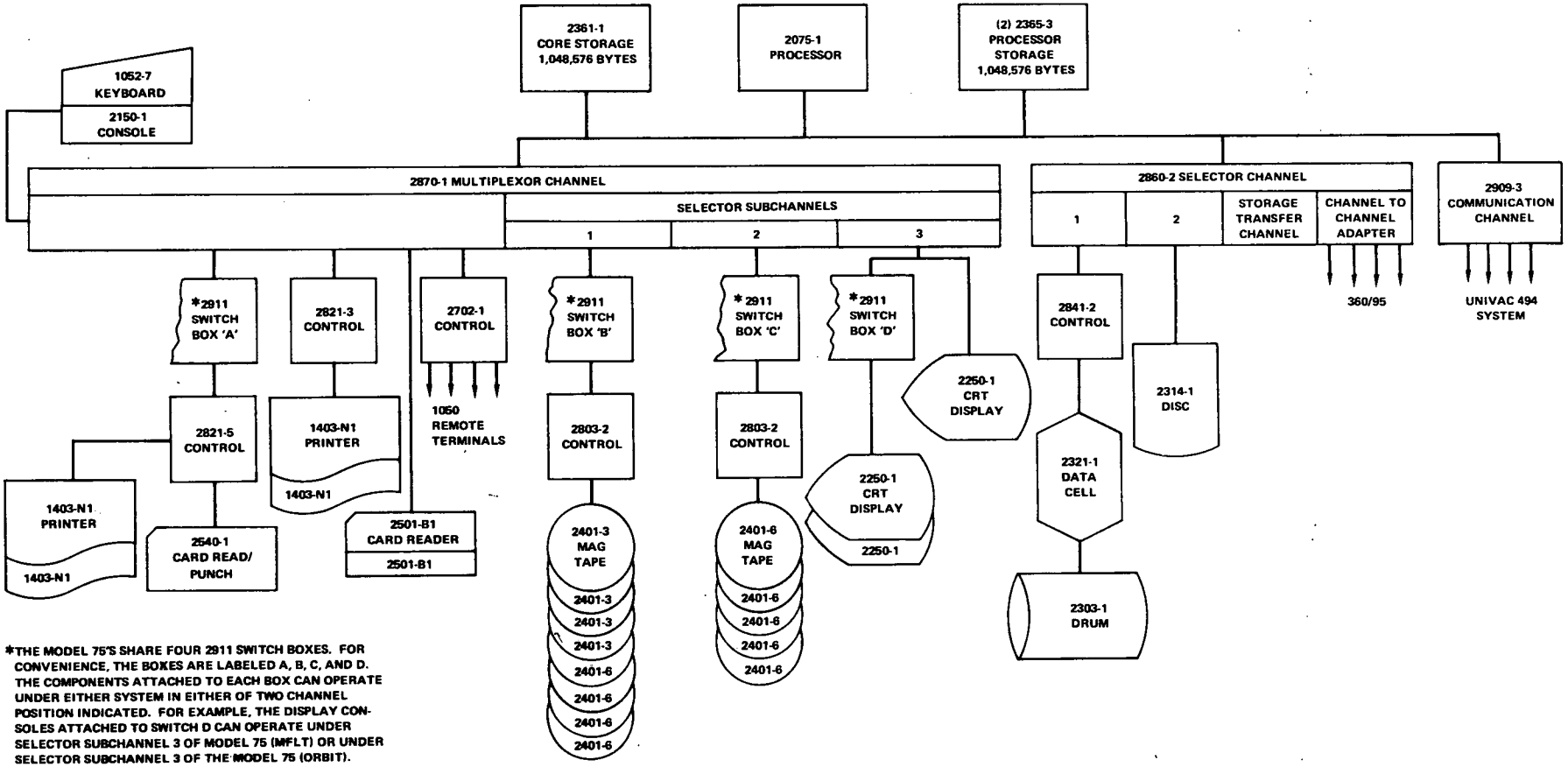
d. One Model 2150/1052 Typewriter Console.

e. One 2860-2 Selector Channel with:

   1. One 2314-1 Direct-Access Storage Facility, with a capacity of 233,408K bytes.

   2. One 2321-1 Data Cell, with a capacity of 400,000K bytes.

   3. One 2303-1 Drum Storage Unit, with a capacity of 4,000K bytes.



*THE MODEL 75'S SHARE FOUR 2911 SWITCH BOXES. FOR CONVENIENCE, THE BOXES ARE LABELED A, B, C, AND D. THE COMPONENTS ATTACHED TO EACH BOX CAN OPERATE UNDER EITHER SYSTEM IN EITHER OF TWO CHANNEL POSITION INDICATED. FOR EXAMPLE, THE DISPLAY CONSOLES ATTACHED TO SWITCH D CAN OPERATE UNDER SELECTOR SUBCHANNEL 3 OF MODEL 75 (MFLT) OR UNDER SELECTOR SUBCHANNEL 3 OF THE MODEL 75 (ORBIT).

Figure 3.3-1. Equipment Configuration: IBM 360 System, Model 75 (ORBIT) June 1970

f. One 2870-1 Multiplexor Channel, including three selector subchannels with:

1. Three 2250-1 Display Units
2. Four 2401-6 9-track tape drives
3. Four 2401-3 7-track tape drives
4. One 2501-B1 Card Reader
5. One 2540-1 Card Read Punch
6. Two 1403-N1 Printers with a QN print chain
7. One 2702-1 Transmission Control Unit for the attachment of 1050 remote terminals.

For units switchable between the MFLT 75 and the ORBIT 75 see paragraph 2.4.3.3. In addition, various communication channels are present for interfacing with the Model 95, and Univac 494 system.

3.3.4 UNIT ADDRESSES

The following are the device address for the units attached to the Model 75 (ORBIT). They are presented here for the reader's edification; however, unit addresses should never be used in the UNIT field on DD cards.

Device Type	Address
2150/1052	01F
2314-1	230-237
2321-1	193
2303-1	197
2250-1	0E0-0E2
2401-6	0C1-0C4
2401-3	0C0, 0C5-0C7
2501-B1	04C
2540-1	00C,00D,01C,01D
1403-N1	002,003,00E,00F
2702	020-027, 030-037

3.3.5 SERIAL NUMBERS

The volume serial numbers for the DASDs on the Model 75 are as follows:

Device Type	Use	Volume Serial Number
2314-1	Scratch or work packs	G3SCR0 - G3SCR3
2314-1	System pack	G3SYS1
2321-1	User Libraries	
2303-1	System storage	G3DRUM

3.3.6 SOFTWARE

The 360/75 (ORBIT) is currently operating under Release 19.6 of the IBM S/360 Operating System with MVT.

3.4 M&DO IBM 360/65

3.4.1 LOCATION

The Model 65 is located in Building 14, Room E4. It is used primarily to support the Orbiting Astronomical Observatory (OAO) project. Its systems hardware and on-line and off-line peripheral devices are described below. Refer to the Computation Division ADP Equipment Guide for more detailed information on the system and peripheral hardware.

3.4.2 SCHEDULING AND OPERATIONS

The Computer Manager for the M&DO 360/65 is Mr. Harry Bitting, Building 3, Room 130, Extension 6886. The Systems Programmers are Mr. Dave Spiegel, Building 3, Room 127, Extension 6798, and Mr. Calvin Curlen, Building 3, Room 133E, Extension 6710.

Users may submit jobs to the Model 65 through the Dispatch Station, Building 14, Room S4, Extension 2195.

3.4.3 HARDWARE CONFIGURATION

As shown in Figure 3.4-1, the system components are:

a. Model 2065-I CPU, with a basic machine cycle of 200 nanoseconds. (Equipped with the standard, floating point, and decimal instruction sets).

b. Two Model 2365-2 Processor Storage units with a capacity of 256K bytes each of high-speed storage and a basic cycle time of 750 nanoseconds.

c. One Model 2361-1 Large Core Storage unit with a capacity of 1024K bytes and a cycle time of 8 microseconds.

d. One Model 2150 console with two Model 1052 typewriter attachments.

e. One 2860-1 Selector Channel with:

   1. Two 2314-1 Direct-Access Storage Facilities, with capacities of 233,408K bytes each.

   2. One 2303-1 Drum Storage Unit, with a capacity of 3,913K bytes.

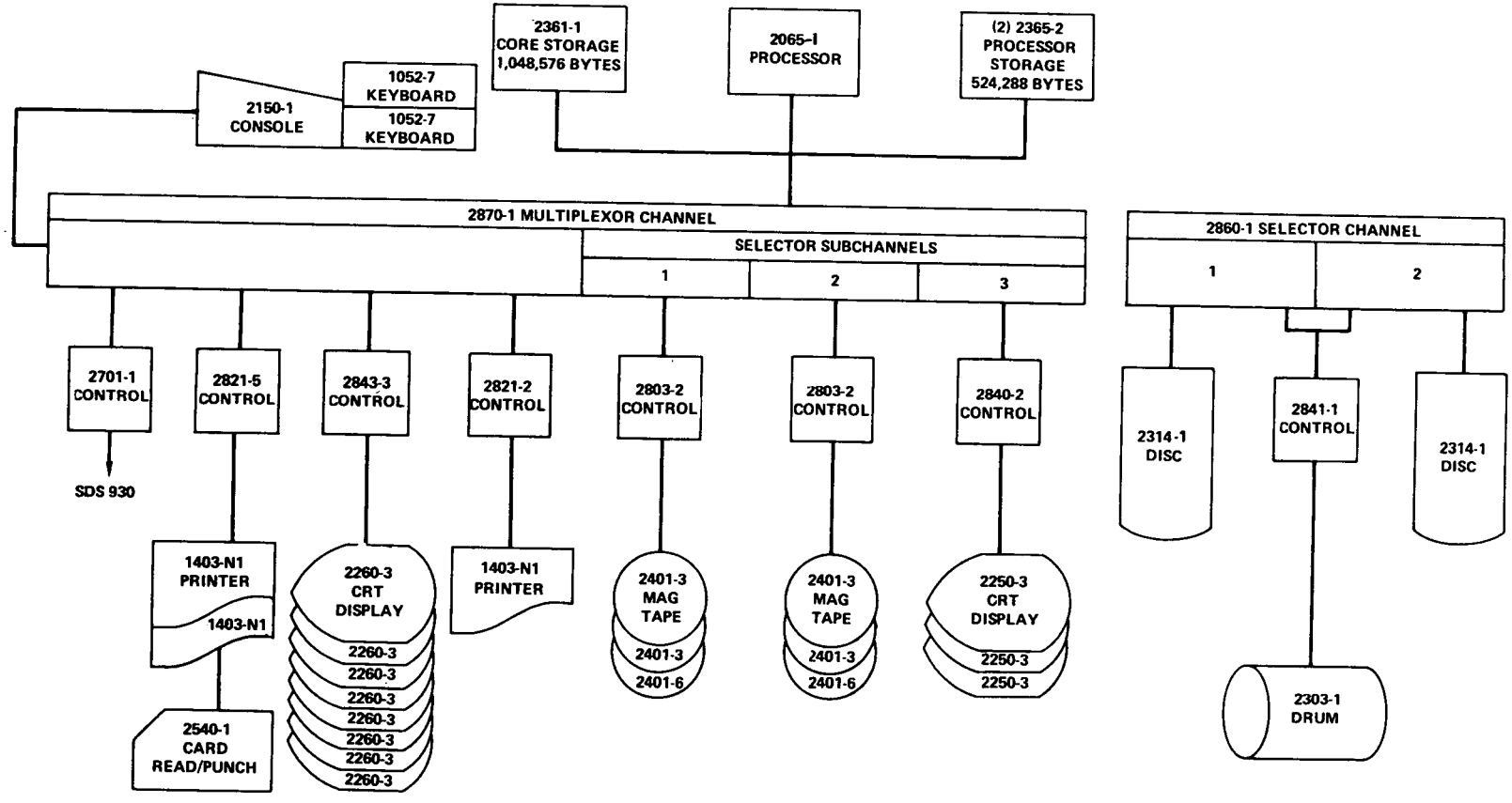


Figure 3.4-1. Equipment Configuration: IBM 360 System, Model 65

f. One 2870-1 Multiplexor Channel, including three selector sub-channels with:

1. Three 2250-3 Display Units
2. Two 2401-6 9-track tape drives
3. Two 2401-3 9-track tape drives
4. Two 2401-3 7-track tape drives
5. One 2540-1 Card Read Punch
6. Three 1403-N1 Printers with a QN print chain
7. Eight 2260-1 Display Units
8. One 2701-1 Transmission Control Unit, used for communication with SDS 930.

3.4.4 UNIT ADDRESSES

The following are the device addresses for the units attached to the Model 65. They are presented here for the reader's edification; however, unit addresses should never be used in the UNIT field on DD cards.

Device Type	Address
2150/1052	01F
2314-1	130-137
2314-1	230-237
2303-1	197
2401-6	0C2,0D2
2401-3 9-track 800BPI	0C1,0D1
2401-3 7-track	0C0, 0D0
2540-1	00C,00D
1403-N1	001,00E,00F
2260-1	050-057
2250-3	0E0-0E2

3.4.5 SERIAL NUMBERS

The volume serial numbers for the DASDs on the Model 65 are as follows:

Device Type	Use	Volume Serial Number
2314-1	Scratch packs	G2SCR1 - G2SCR4
2303-1	System storage	G2DRUM

3.4.6 SOFTWARE

The 360/65 is currently operating under Release 19.6 of IBM S/360 Operating System with MVT.

3.5 PERIPHERAL AND ACCESSORY EQUIPMENT

The information in this section is extracted from the Computation Division ADP Equipment Guide.

3.5.1 LOCATION

The IBM 360/30, 360/20, and the CDC 160A systems are located in the Main Computer Room, Building 3, Room 153. See Figure 3.5-1 for the floor plan. One IBM 360/30 is located in the Riggs Building. All requests for work to be performed on these systems are submitted to the Dispatch Station, Building 3.

3.5.2 PURPOSE

The smaller models of the IBM 360 system are used primarily to prepare input and output for the larger computing systems.

The Model 20 provides the card-processing services described in Paragraph 2.3.7 of this manual. The Model 30 in Building 3 provides off-line support for the IBM 7094 system. It performs tape-to-print, card-to-tape, tape-to-tape, and tape-to-card operations.

The S/360 Model 30 offers several utilities for operations such as tape-to-print, card-to-tape, tape-to-punch, tape copying, octal dump, and hex dump. The two most important utilities are DEBE and the System/360 Multiple Utility Program for 7094 Support.

DEBE is used for operations such as copying from card or tape to either card, tape, or print. DEBE will handle either 7- or 9- track tapes.

The Multiple Utility for 7094 Support is used for multiple card-to-tape, tape-to-print, and tape-to-punch operations with 7- or 9-track tapes.

The Octal Tape Dump Utility is used to dump 7-track tapes in octal format.

The Hex Tape Dump Utility provides a hex dump of 7- or 9-track tapes.

The Block Tape utility is used to print blocked tapes. The default blocksize is 7200; any other blocksize must be stated explicitly.

Execution of the utility programs is under the control of the operator. The user needs only to specify the function to be performed and any identifying information. For correct and efficient operation, the user must specify:

- 7- or 9-track tape (800 bpi only)
- Density

- Blocksize
- Number of files on tape
- Any special instructions required to further define or clarify the operation

Since there is only one 9-track drive on this system, 9-track tapes cannot be duplicated.

The Model 30 in the Riggs Building supports the IBM 360/75 (MFLT) system. It performs data transfer operations similar to those performed by the Model 30 in Building 3. In addition, it is uniquely equipped for printing documents; its printer has a text-train character set (i.e., a full upper and lower case and all special characters).

The CDC 160A system interfaces the local STADAN operations and the various tracking stations involved in those operations. Predicted orbit data generated by the 360/75 system are transferred from magnetic tape to paper tape and relayed via teletype.

3.5.3 REFERENCES

IBM Reference Manual, IBM System/360 Model 20, System Summary, Form GA26-5889.
IBM Reference Manual, IBM System/360 Model 30, Functional Characteristics, Form GA24-3231.

CDC Reference Manual, Control Data 160 Computer, Publication No. 60002300.

Revised: September 1971

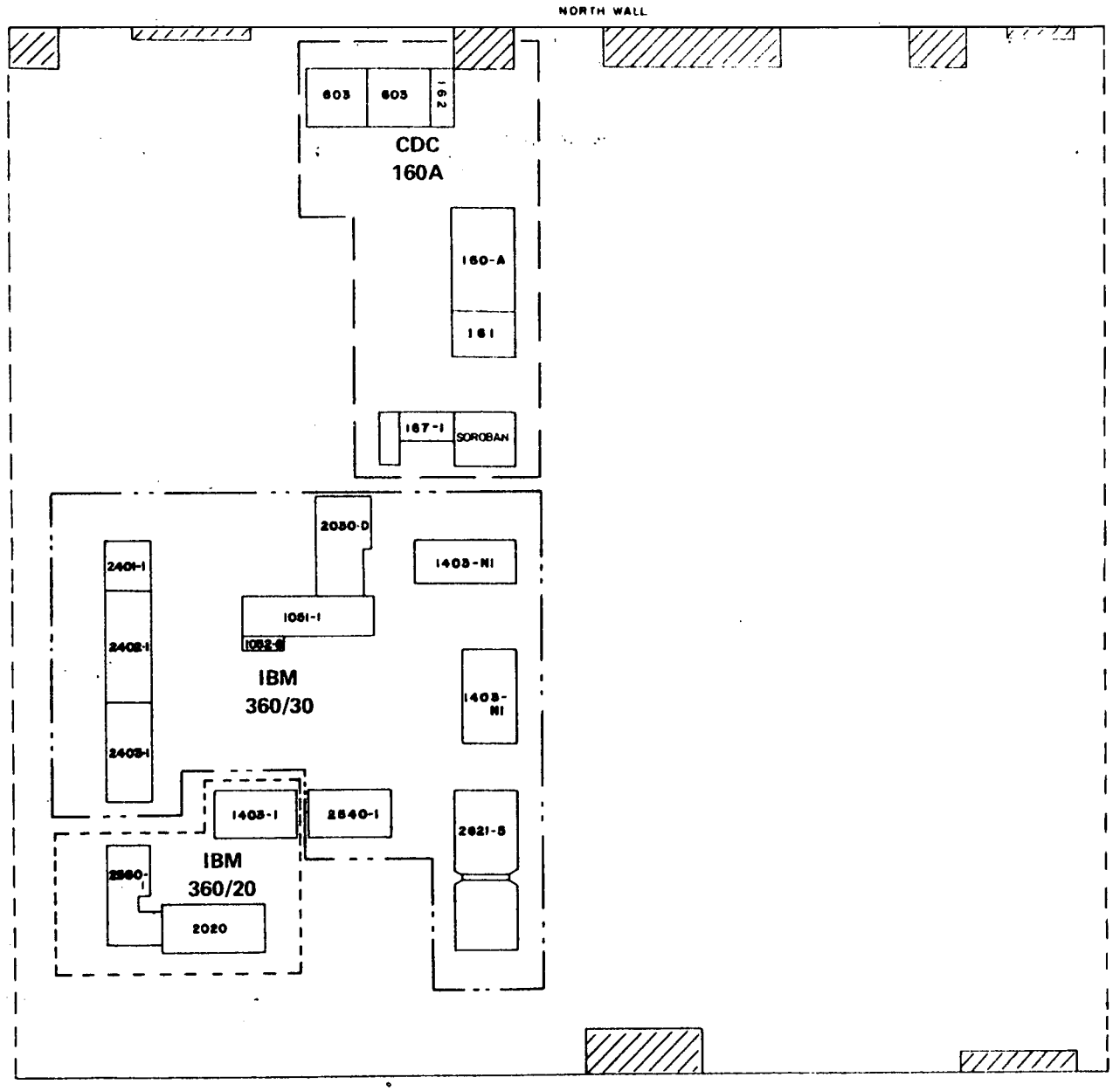


Figure 3.5-1. Floor Plan: IBM 360/30 and 360/20, and CDC 160A
September 1971 Bldg. 3, Room 153

3.6 UNIT CHARACTERISTICS

3.6.1 DIRECT-ACCESS DEVICES

A Direct-Access Storage Device (DASD) is one on which each physical record has a discrete location and a unique address. Thus records can be stored on a DASD in such a way that the location of any one record can be determined without extensive searching. Records may be accessed serially, but they may be organized so that they may be accessed directly.

The three types of DASDs are: the disk, the drum, and the data cell. Each is described in great detail in Introduction to IBM System 360 Direct-Access Storage Devices and Organization Methods, Form C20-1649. Table 3.6-1 which is extracted from the above-mentioned document, summarizes the characteristics of each type.

In the table, the capacity of a track is expressed in terms of the maximum number of data bytes. This maximum may be achieved when there is one physical data record (block) per track formatted without a key. As the track is divided into multiple data records, the additional address markers, count areas, and gaps reduce the number of bytes available for data.

Table 3.6-2 gives the capacity from the standpoint of how many physical data records of a given length will fit on a track. In some cases, the table cannot be used and the number of records per track for a given record design must be calculated, using the formulas discussed in the Introduction to IBM System 360 Direct-Access Storage Devices and Organization Methods, from which the table is extracted.

Note that the table is divided into two parts, since the capacity varies depending on whether records are formatted with or without keys. Normally at Goddard, records are formatted without keys. Examples using the table:

- Device is the 2314, records are unblocked and formatted without keys, and data length is 400 bytes. There will be 14 records per track.

- Device is the 2321, records are unblocked, and formatted with keys, data length is 100 bytes, and key length is 8 bytes. In using the right-hand side of the table, the number to look up is data length plus key length - in this example, 108. There will be 9 records per track.

- Device is the 2301, records are blocked and formatted without keys, blocking factor is 3, and logical record length is 900 bytes. The data area will be 2700 bytes, so there will be 7 blocks of 3 records each or 21 logical records per track.

Device	Storage Medium	Cylinders	Tracks per Cylinder	Bytes per			Access Motion (MS)			Rotation (ms) (Full)	Transfer Rate (KB)
				Track	Cylinder	Device (Million)	Min.	Max.	Avg.		
2314	Disk	Pack: 200 Model A1 Total: 1600 Model A2 Total: 1000	20	7294	145,880	Pack 29.17 Model A1 Total: 233,408 Model A2 Total: 145,880	25	130	60	25	312
2303	Drum	80	10	4892	48,920	3.9	0	0	0	17.5	303.8
2301	Drum	1	200	20483	4.09 (Million)	4.09	0	0	0	17.5	1200
2321	Strip of tape	Strip: 5 Array: 10,000	20	2000	40,000	400	95	600	350*	50	56

*Assuming that the previously addressed strip has already been restored. If this assumption cannot be made, average access time is 550 ms.

Table 3.6-1. Direct Access Device Characteristics

Maximum Bytes per Physical Record Formatted without Keys				Physical Records per Track	Maximum Bytes per Physical Record Formatted with Keys			
2314	2303	2301	2321		2314	2303	2301	2321
7294	4892	20483	2000	1	7249	4854	20430	1984
3520	2392	10175	935	2	3476	2354	10122	920
2298	1558	6739	592	3	2254	1520	6686	576
1693	1142	5021	422	4	1649	1104	4968	406
1332	892	3990	320	5	1288	854	3937	305
1092	725	3303	253	6	1049	687	3250	238
921	606	2812	205	7	877	568	2759	190
793	517	2444	169	8	750	479	2391	154
694	447	2157	142	9	650	409	2104	126
615	392	1928	119	10	571	354	1875	103
550	346	1741	101	11	506	308	1688	85
496	308	1585	86	12	452	270	1532	70
450	276	1452	73	13	407	238	1399	58
411	249	1339	62	14	368	211	1286	47
377	225	1241	53	15	333	187	1188	38
347	204	1155	44	16	304	166	1102	29
321	186	1079	37	17	277	148	1026	21
298	169	1012	30	18	254	131	959	15
276	155	952	24	19	233	117	899	9
258	142	897	20	20	215	104	844	
241	130	848	15	21	198	92	795	
226	119	804	10	22	183	81	751	
211	109	763	6	23	168	71	710	
199	100	726		24	156	62	673	
187	92	691		25	144	54	638	
176	84	659		26	133	46	606	
166	77	630		27	123	39	577	
157	70	603		28	114	32	550	
148	64	577		29	105	26	524	
139	58	554		30	96	20	501	

Table 3.6-2. Track Capacities

3.6.2 IBM 2400-SERIES TAPE DRIVES

IBM magnetic tape is a continuous recording medium similar to the tape used in home recorders. Data are recorded in magnetized spots or bits, are permanent, and can be retained for an indefinite period. As data are recorded, the previous information is erased, thus permitting repetitive use of the tape. IBM Form A22-6866, IBM System/360 Component Descriptions, 2400-Series Magnetic Tape Unit, contains a comprehensive presentation of the characteristics, functions, and features of the IBM 2400-Series Magnetic Tape Units. Table 3.6-3 summaries the main characteristics.

3.6.3 OTHER HARDWARE COMPONENTS

The Computation Division ADP Equipment Guide discusses other hardware components, such as EAM equipment, card readers and punches, printers, other types of magnetic tape units, display consoles, remote terminals, direct access external storage devices, control units, line attachments, and processors and storage units.

3.6.4 CHARACTER SETS AND CODES

3.6.4.1 Character Sets

IBM has 12 standard character sets which can be used on the Model 1403 printer, and which provide for printing any set of up to 240 graphics. The print train normally used on the GSFC printers is suitable for most purposes and provides the best combination of speed and readability. The more characters available for printing, the slower the printing operation will be. The full character set is available on the 1403 printer that is attached to the IBM 360/30 in the Riggs Building. Refer to the IBM System/360 Operating System Operator's Guide, Form C28-6540 and the IBM 1403 Printer, Form A24-3073 for greater detail on the available character sets.

3.6.4.2 Computer Codes

The System 360 accepts two principal, coding schemes: Extended Binary Coded Decimal Interchange Code (EBCDIC) and USA Standard Code for Information Interchange (USASCII).

EBCDIC uses eight binary positions for each character format, plus a position for parity checking. By using eight-bit positions, 256 different characters can be coded. This code permits, for instance, the coding of uppercase and lower case alphabetic characters, a wide range of special characters, and many control characters that are meaningful to certain input/output devices. At present, many bit patterns have no assigned function (control or graphic). They are reserved for future assignment. EBCDIC is one of the two principal coding schemes for System 360.

Characteristics	2401 - 2404 Tape Units						2415 Tape Units		2420 Tape Unit
	Model 1 / Model 4	Model 1	Model 2 / Model 5	Model 2	Model 3 / Model 6	Model 3	Model 1-3 / Model 4-6	Model 1-6	Model 7
Number of Tracks and Recording Method	9-Track NRZI / 9-Track PE	7-Track NRZI	9-Track NRZI / 9-Track PE	7-Track NRZI	9-Track NRZI / 9-Track PE	7-Track NRZI	9-Track NRZI / 9-Track PE	7-Track NRZI	9-Track PE
Density (BPI) Bytes per inch	800 / 1600	800 556 200	800 / 1600	800 556 200	800 / 1600	800 556 200	800 / 1600	800 556 200	1600
Data Rate (Bytes/Sec)	30,000 / 60,000	30,000 20,850 7,500	60,000 / 120,000	60,000 41,700 15,000	90,000 / 180,000	90,000 62,500 22,500	15,000 / 30,000	15,000 10,425 3,700	320,000
Tape Speed (In/Sec)	37.5 / 37.5	37.5	75.0 / 75.0	75.0	112.5 / 112.5	112.5	18.75 / 18.75	18.75	200
Interblock Gap (Inches)	0.6 / 0.6	.75	0.6 / 0.6	.75	0.6 / 0.6	.75	0.6 / 0.6	.75	0.6
Nominal IRG Time (In/MS.)	16.0 / 16.0	16.0	8.0 / 8.0	8.0	5.3 / 5.3	5.3	32.0 / 32.0	32.0	3.0
Rewind Time (In/Min)	3.0 / 3.0	3.0	1.4 / 1.4	1.4	1.0 / 1.0	1.0	4.0 / 4.0	4.0	1.0
Rewind & Unload (In/Min)	2.2 / 2.2	2.2	1.5 / 1.5	1.5	1.1 / 1.1	1.1	4.0 / 4.0	4.0	1.1

Table 3.6-3. 2400 Series Magnetic Tape Unit Characteristics

USASCII is a seven-bit code developed through the cooperation of users of equipment of communications and data processing industries, in an attempt to simplify and standardize machine-to-machine and system-to-system communication.

Because the System 360 has an eight-bit character capacity, it was necessary to expand USASCII to an eight-bit representation. This expanded representation is referred to by IBM as USASCII-8. This code may be used for internal processing and input/output purposes with System 360 in those media for which USASCII has been standardized.

The coding and character representation for EBCDIC and USASCII may be found in IBM System 360 Principles of Operation, Form A22-6821. EBCDIC is an extension of Binary Coded Decimal (BCD) interchange code, which is used extensively on second-generation equipment (e.g., IBM 1401, 1410, 7010, 7090, and 7094 Data Processing Systems). Refer to the green card (IBM System 360 Reference Data, Form X20-1703) for the relationship between EBCDIC and BCD.

3.6.4.3 Card Codes

The standard Hollerith card code uses the twelve possible punching positions of a vertical column on a card to represent a numeric, alphabetic, or special character. The twelve-hole positions are divided into two areas, numeric and zone. The first nine-hole positions from the bottom edge of the card are the numeric hole positions and have an assigned value of 9, 8, 7, 6, 5, 4, 3, 2, and 1, respectively, the remaining three positions, 0, 11, and 12, are the zone positions. (The 0 position is considered to be both a numeric and a zone position.)

The numeric characters 0 through 9 are represented by a single hole in a vertical column. For example, 0 is represented by a single hole in the 0 zone position of the column.

The alphabetic characters are represented by two holes in a single vertical column, one numeric hole and one zone hole. The alphabetic characters A through I use the twelve-zone hole and a numeric hole 1 through 9, respectively. The alphabetic characters J through R use the eleven-zone hole and a numeric hole 1 through 9, respectively. The alphabetic characters S through Z use the 0-zone hole and a numeric hole 2 through 9, respectively.

The standard special characters $, *, %, and so on, are represented by one, two, or three holes in a column of the card and consist of hole patterns not used to represent numeric or alphabetic characters.

The card punch configuration for these characters may be found on the IBM green card (IBM System/360 Reference Data, Form X20-1703). Since EBCDIC contains more characters in its set than does BCD, the 029 keypunch offers more special characters than the 026. In addition, there are different hole patterns on the two keypunches for five of the special characters. These characters and the card codes for each of the two keypunches are shown below:

Character	Keypunch Code	
	026	029
+	12	12-8-6
=	8-3	8-6
(	0-8-4	12-8-5
)	12-8-4	11-8-5
'	8-4	8-5

3.6.4.4 Paper Tape

Punched paper tape serves much the same purpose as punched cards. Developed for transmitting telegraph messages over wires, paper tape is now used for data processing communication as well. For long-distance transmission, machines convert data from cards and keyboard strokes to paper tape, send the information over telephone or telegraph wires to produce a duplicate paper tape at the other end of the wire, and reconvert the information to punched cards, for later processing.

Data are recorded as a special arrangement of punched holes, precisely arranged along the length of a paper tape. Paper tape is a continuous recording medium, as compared to cards, which are fixed in length. Thus, paper tape can be used to record data in records of any length, limited only by the capacity of the storage medium into which the data are to be placed or from which the data are received.

Data punched in paper tape are read or interpreted by a paper tape reader and recorded by a paper tape punch.

SECTION 4

SOFTWARE STATUS

4.1 GENERAL DISCUSSION

In the dynamic environment of the S/360 Operating System, the status of software items is continually changing through new system releases, development of new or improved language processors, updating of proprietary packages, and countless other modifications. Most of these changes are compatible with previous versions or require only minor modifications to existing programs. Software packages are available directly through LINKLIB; however, they may be stored in a private library and called by the use of a JOBLIB or STEPLIB card. Users desiring to use programs not available on an M&DO computer should contact the PAC for information on how to temporarily include them in the system. As changes are made to the system, they will be announced in the GSFC Newsletter, the M&DO 360 Computer Bulletin, and by notices in the Programmer Assistance Center. Users are strongly urged to use this information to maintain an up-to-date knowledge of the M&DO computer systems.

4.2 CURRENT SOFTWARE STATUS

Table 4.2-1 presents the software available on the M&DO computers. Because the system libraries are continually being updated to satisfy the requirements of changing user demands, the user should verify that his required programs are (1) still available and (2) stored in the specified library.

Release 19 of the Operating System is in use on all of the M&DO 360 computers.

4.3 CURRENT MAJOR OUTSTANDING SOFTWARE PROBLEMS

a. Number of Buffers for Unformatted Writes in FORTRAN - Under Release 19 an unformatted write with BUFNO=1 specified on the DD card, under certain conditions, produces incorrect output. IBM may not correct this error until Release 20. All users should therefore accept the default value of BUFNO=2 when doing unformatted writes under Release 19.

b. CHECKPOINT/RESTART (see section 11.5) - Step Restart works under OS Release 19. There are problems with Checkpoint within a job step. Consult the PAC before using CHECKPOINT/RESTART.

Table 4.2-1. Available Software on M & DO Computers

ITEM	SECTION	360/95	360/75	360/65
OPERATING SYSTEM RELEASE		RELEASE 19	RELEASE 19	RELEASE 19
NUMBER OF BYTES OF USER MEMORY		4800K	1800K	1500K
DEFAULT REGION SIZE		100K	80K	100K
PROCESSORS:				
ASSEMBLER (G)	19.3.1	X	J	J
ASSEMBLER (F)	6.2, 19.3.1	X	X	X
FORTRAN (G)	6.2, 19.3.1	X	X	X
FORTRAN (H)	6.2, 19.3.1	X	X	X
PL/1 (VERSION 4.3)	6.2, 19.3.1	X	J	N
PL/1 (VERSION 5.1)	6.2, 19.3.1	J	J	J
RPG	**	X	N	N
LINKAGE EDITOR (E)	**	X	N	N
LINKAGE EDITOR (F)	6.3.1, 19.3.2	X	X	X
LOADER	6.3.1, 19.3.2	X	X	X
SORT/MERGE	6.4, 19.3.3	X	X	X
OS UTILITIES	9	X	X	X
FAPCON (FORTRAN SINGLE TO DOUBLE PRECISION CONVERT)	20.2, 19.3.8	X	N	N
FORMAC (FORMULA MANIPULATION COMPILER)	7.3, 19.3.9	X	J	N
RITS (REMOTE INPUT TERMINAL SYSTEM)	14	X	N	N
CRBE (COMPUTER REMOTE BATCH ENTRY)	14	N	N	N
RJE (REMOTE JOB ENTRY)	13	X	N	N
APL (A PROGRAMMING LANGUAGE)	15	X	N	N
SIGPAC (SIGNIFICANCE PACKAGE)	21.10	J	N	N
OTHER SOFTWARE PACKAGES:				
BEEF (BUSINESS AND ENGINEERING ENRICHED FORTRAN)	*	X	N	N
BOOLE & BABBAGE (PROBLEM PROGRAM ANALYZER)	7.2, 19.3.7	X	J	X
OSSLIP (UTILITY)	9	X	J	N
GPCP (GENERAL PURPOSE CONTOUR PROGRAM)	*	N	N	N
GPSS (GENERAL PURPOSE SIMULATION SYSTEM)	7.5, 19.3.10	X	N	N
SSP (SCIENTIFIC SUBROUTINE PROGRAM)	*	X	N	J
REENTRANT ASSEMBLER	7.4	X	N	N
ECAP (ELECTRONIC CIRCUIT ANALYSIS PROGRAM)	*	N	N	N
CONVERSION AIDS:				
UNPACK (USE 7094 PACKED DATA ON S/360)	*	G	G	F
DEBLOCK/CNVRT (CONVERT 7094 TAPES TO S/360)	20.3	G	G	F
TIDY (REFORMAT FORTRAN SOURCE PROGRAMS)	20.5	N	N	N
FORTLCP (FORTRAN TO PL/1 CONVERSION)	20.5	N	N	N
DATCON (WRITE 7094 OR 1107 TAPES FROM S/360 FORTRAN)	20.4	G	G	F
DACUT9 (WRITE 7094 BINARY TAPES FROM S/360 FORTRAN)	*	G	G	F
DATASIFT (DATA STATEMENT SIFT PROGRAM)	20.1	X	N	N
GRAPHICS:				
GSP (GRAPHICS SUBROUTINE PACKAGE)	12	X	X	X
GTS (GRAPHICS TERMINAL SYSTEM)	7.6	J	J	J

Table 4.2-1. Available Software on M & DO Computers (Cont'd)

ITEM	SECTION	360/95	360/75	360/65
GRAPHICS (cont'd)				
GJP (GRAPHICS JOB PROCESSING)	12	J	X	X
SC4020 PLOT PACKAGE	12	J	J	N
SD4060 PLOT PACKAGE	12	J	J	N
GPAK (GRAPHICS SUBROUTINE LIBRARY)	*	N	J	N
TADPOLE (TEST AND DEBUG PROGRAM FOR ON LINE EXECUTION)	21.9	N	N	N
BIT MANIPULATION ROUTINES:	7.7	F	G	F
AND				
OR (INCLUSIVE OR)				
EXCLUSIVE OR				
ONES COMPLEMENT				
BITON (SET A BIT TO 1)				
BITOFF (SET A BIT TO 0)				
BITFLP (COMPLEMENT A BIT)				
SHIFT LEFT				
SHIFT RIGHT				

KEY:

X = AVAILABLE THROUGH LINKLIB

J = AVAILABLE IN ANOTHER LIBRARY; MUST USE JOBLIB, OR CATALOGED PROCEDURE HAS A STEPLIB OR SYSLIB CONCATENATION

N = NOT AVAILABLE

G = GSFCLIB (EXTENSION OF FORTLIB)

F = FORTLIB

* = NOT INCLUDED IN DOCUMENT AT THIS TIME. CONTACT MRS. PAT BARNES IN THE GSFC PROGRAM LIBRARY IN BUILDING 3, FOR AVAILABLE DOCUMENTATION

** = NOT INCLUDED IN DOCUMENT AT THIS TIME. SEE THE APPROPRIATE IBM MANUALS FOR DOCUMENTATION AND THE SYS1. PROCLIB OF THE APPROPRIATE COMPUTER FOR THE CATALOGED PROCEDURE.

4.4 LISTNEWS, RITS, AND APL NEWSFILES

A short summary of current information concerning the 360/95 may be obtained by inserting the following card after the JOB card: // EXEC LISTNEWS. If needed, LISTNEWS will be updated on a daily basis.

Pertinent information of a less current nature may be obtained by placing a // EXEC LISTOLD card after the JOB card.

Information pertaining to RITS is available by retrieving the SYSTM.NEWS file. To do this, enter:

```
/F SYSTM.NEWS
%L
```

after signing on on a RITS terminal.

The latest APL news can be obtained by entering:

```
)LOAD 1 NEWS
DESCRIBE
```

4.5 M&DO 360 COMPUTER BULLETIN

The M&DO 360 Computer Bulletin is published on an as-needed basis. It is distributed to members of the M&DO 360 User's Committee, which is responsible for passing the information contained in the bulletin along to the people it represents. Additional copies may be obtained from the Analysis and Programming Branch, Code 543, extension 6887.

4.6 GSFC COMPUTER NEWSLETTER

The GSFC Computer Newsletter is also published on an as-needed basis. Copies may be obtained from Mr. Dave Kohnhorst, Code 601, extension 2251.

SECTION 5

JOB SET-UP

5.1 GENERAL INFORMATION

5.1.1 SCOPE OF THIS SECTION

This section explains the fundamental details of setting up a deck of JCL and data cards as a job, and submitting them to be run on a T&DS computer. Items explained are the job submission slips and procedures (which are detailed in subsection 2.3) and the IBM Operating System Job Control Language or JCL. The JCL also controls input and output of data, as well as the allocation of input/output devices to the job.

5.1.2 JOB SUBMISSION SLIPS

Available at the dispatcher's desk are three-part slips which must be completed and attached securely to the job being submitted. The user fills in certain information exactly as it appears on the job card described below, including sponsor number, programmer I.D., job name, box number, and length of run. As on the job card, both CPU and I/O times should be put in the appropriate boxes on the slip. In addition, if tapes or private packs are to be used, the user must write the identifying numbers on the slip. This is important because it forewarns the computer operator that a particular volume will be required by the job and he will have it available before the job is run.

The operators separate jobs into batches depending on their use of private packs. This reduces the time required to exchange these volumes. The dispatcher also forwards any job requests requiring specific tapes via the tape library which supplies the tape with the job.

There is a space on the job submission slip for required storage. The operators of the smaller computers use this information to determine whether they can start another job. The remarks section should be used to indicate special requirements, such as an advanced or older version of the operating system or a particular machine on which the job is to be run.

5.2 JOB CONTROL LANGUAGE

5.2.1 PURPOSE

The 360 operating system was intended to be very flexible - it assumes many of the functions which are performed manually in less sophisticated systems. The OS loads and initiates programs, monitors their execution, and terminates them when necessary. In order to perform these functions, the control program must be given instructions for each job that it will monitor. In S/360 these instructions are given in the Job Control Language (JCL) which must accompany every program to be run.

5.2.2 OPERATION CONSIDERATIONS

In order to give the user a clearer understanding of the meanings of the various JCL statements and operands, the following discussion very briefly presents the sequence of events by which the operating system reads the JCL, schedules jobs, and allocates resources.

JCL is read into the operating system by a "reader-interpreter" which is described in subsection 11.8. The reader processes the "input stream" which contains JCL and input data. JCL is distinguished from the other data in the input stream by the first two characters in each card. With the exception of the delimiter or /* card which ends or delimits a section of data in the input stream, all JCL cards start with // in columns 1 and 2.

As the "reader" reads the input stream, it buffers (spools) the data to disk. The "interpreter" scans the JCL diagnosing format errors, breaks the information down, and places it on the job queue (the SYS1.SYSJOBQE data set) where it also places pointers to the input stream data which has been spooled onto the disk. If it finds a JCL error, it assigns a 14 priority (highest in the system) to the job and flushes it out of the system. An "initiator" looks at the job queue and pulls the job and data off the queue, in their proper turn, based on priority and job class. The initiator selects the job from the job queue, analyzing the I/O requirements of the job, allocating devices to fill the requirements, issuing volume mount instructions where needed, and verifying that the correct volumes were mounted. Once the job is completely executed, the terminator terminates it, de-allocates its data sets and volumes, and passes it back to the job queue. From there the writers pick up the system messages and system output data, which have been spooled out during execution of the job, and print them. The operators pick up the printed output and punched cards, if any; place them together with the job submission slip, input deck, tapes, and whatever else was submitted; and send them back to the dispatcher for delivery to the programmer.

5.2.3 GENERAL FORMAT OF JCL STATEMENTS

There are eight types of JCL statements. Four are normally used by programmers. These are the JOB statement, EXECUTE statement, DATA DEFINITION statement, and the DELIMITER or /* statement. Four other JCL cards exist: the NULL statement or // card used by the operators at Goddard to separate jobs; the COMMENT statement -- that is the //* card in which programmers may put comments as an aid in documenting their JCL; the PROC statement, which is used only in cataloged procedures; and the COMMAND statement, which may be used by operators to enter commands in the input stream rather than from the console.

The general format of JCL cards is a // in columns 1 and 2 followed by a name field, an operation field, and an operand field, followed by a comments field. The name field begins immediately after the second slash, while the name, operation, operand, and comment fields are separated from each other by one or more blank spaces. The operand and comments fields may also be continued on successive cards.

5.2.3.1 Name Field

Occasionally, the name field may be omitted, but preferably should be included, as it identifies a control card so that other JCL cards or system functions can refer to it. It can be from one to eight characters in length and can contain any alphanumeric or "national" characters. The alphanumeric characters are A to Z and 0 to 9, and the national characters are the @ sign, the $ sign, and the number or pound sign, #.

5.2.3.2 Operation Field

The operation field specifies the type of control card, JOB, EXEC, DD, PROC, or an operator's command. The other types of JCL cards (i.e., comment, delimiters, null) do not have operations.

5.2.3.3 Operand Field

The operand field contains parameters. Some of these parameters are positional. That is, they are only legal in a certain position within the operand. Others are of the keyword format: (KEYWORD=value(s)), where "keyword" is the name of the parameter and "value" is the value assigned to that parameter. Some parameters also have subparameters, which likewise may be positional or keyword. Parameters are separated by commas (blanks are not permitted). Positional parameters must be coded in the order specified before any keyword parameters. The absence of a positional parameter is indicated by the

coding of a comma in its place. If the absent positional parameter is the last parameter, or if all other positional parameters are also absent, replacing commas should not be coded. Parameters with multiple values must be coded within parentheses. That is, subparameters which are likewise separated by commas are coded surrounded by parentheses so that the scanning routines of the reader-interpreter do not interpret them as separate prime parameters. The keyword parameters may be coded in any order after the positional parameters.

If any characters are used other than the alphanumeric or national characters (and occasionally a period or a hyphen) in some subfields, the parameter must be enclosed in single quotes. If a quote or an apostrophe is used within the field, it must be doubled.

5.2.3.4 Comments

Comments may be coded on any JCL card by leaving one or more blanks between the last field and the beginning of a comment. This is true even if the operand field is to be continued. The operand field can continue no further than column 71 of the card. If the operand will not fit on one card, or if it is desired to split the parameters onto separate cards for readability or any other reason, the field must be interrupted after a complete parameter*, including the comma that follows it, at or before column 71, and a // must be coded in the next card; the interrupted card is continued in column 4 to 16. If the continuation card is begun after column 16, the continuation is treated as a comment.

Besides interrupting after a complete parameter in order to continue, certain subparameters can be interrupted. These are in the account and PARM fields on execute cards; condition parameters on job and execute cards; and the DCB, VOL=SER, SEP, and UNIT=SEP parameters on DD cards.

5.2.4 JOB SEQUENCING

On occasion, the user may need to run multiple jobs which are order-dependent, i.e., the execution of one job is dependent upon the successful completion of another job.

The introduction into the system of one job followed by another is no guarantee that the second job will be run last. Depending upon multiple considerations, including I/O and CPU times and the size of the region required, the second job might well execute first.

*Continuation cards were originally specified by coding a non-blank character in column 72. This is no longer required, but is sometimes seen in older procedures. Under this old system, a continuation card had to start in column 16. However, this is no longer true.

The only reliable means to prevent one job from being selected for processing until after another job has terminated is to code the keyword parameter TYPRUN=HOLD in the JOB card of the job which is to be executed last. The job is then held until a RELEASE command is issued by the operator or the RELEASE procedure is executed. The use of the RELEASE procedure is explained in paragraph 19.3.11. The operator must be informed (by means of the comments field on the job submission slip) about what should be done and when the job should be released. Failure to do so will cause the job to be released prematurely.

5.3 DECK SETUP

The following table presents the sequence in which JCL statements are to be placed when setting up a job.

Table 5.3-1. JCL Statement Sequence

JCL STATEMENT	FUNCTION	SECTION REFERENCE
JOB	Defines a job and supplies accounting and other information about the job to the system.	5.3.1
JOBLIB	Indicates that a private library is to be searched for the program(s) to be executed. This card is not needed for a program which is on the system library.	5.4
EXEC	Defines a "job step", naming the program to be executed (or the cataloged procedure to be referenced).	5.5
DD	Defines a data set to be referenced by the job step. One DD statement is required for each data set referenced.	5.6

A job may have multiple steps by the use of multiple sets of EXEC and DD cards. The completion of the last step in a job is marked with a JOB statement associated with the succeeding job, or a NULL statement (subsection 5.7.2).

The number of EXEC and DD statements allowed per job varies, depending upon the limit of the job queue space. Any job which has too many EXEC and DD statements for the system to handle will be ABENDed with a completion code of 422. If possible, the job should be subdivided into several different jobs and submitted separately. If it is not feasible to do so, the programmers in the PAC (see paragraph 2.3.10) should be consulted.

5.3.1 JOB CARD FORMAT

The JOB card format is as follows:

```
//useidxxx JOB (ssssscpppp,r,prgram,tttttt),box,MSGLEVEL=(x,y)
```

The JOB card is the first card in a deck of cards submitted as a S/360 computer job. The format is determined by the GSFC accounting procedures, and entries must be exactly as shown in Table 5.3-2. More detailed information about the GSFC accounting procedures may be found in subsection 2.1 of this User's Guide.

Table 5.3-2. Job Card Format

Col. 3-10	The programmer's 5-letter ID, plus three more characters that are unique to that job. The combination makes up the job name. (useidxxx)
Col. 12-14	The Word "JOB"
Col. 17-21	The sponsor number (sssss)
Col. 22	The category code (c)
Col. 23-26	The project code (pppp)
Col. 28	The run type, as follows: (r) T for test P for production run R for hardware-error run S for tape-error rerun Q for software-error rerun U for operator-error rerun
Col. 30-35	Program number (prgram)
Col. 37-39	Estimate of CPU time needed to run the job. (ttt)
Col. 40-42	Estimate of I/O time needed for the job. (Note: The operating system will assign a priority to the job, based on the greater of the two time estimates. If either the CPU or I/O estimate is exceeded by the job, the run will be ABENDed. See the discussion below of completion codes for insufficient time.) (ttt)
Col. 45-47	Box number at the computer facility. A user who does not have a box assigned should request one from the dispatcher. (box)
Col. 50-58	MSGLEVEL=(x,y): Informs the job scheduler as to which JCL and allocation/termination messages are to be printed.

Table 5.3-2. (Cont'd)

The value of x may be:

0 Only the JOB statement is to be printed (default value)

1 All JCL statements, cataloged procedures, and overrides to cataloged procedures appear in their proper sequence.

2 Only input job control statements (cataloged procedure statments will not appear).

The value of y may be:

0 No allocation/termination messages are to appear, unless the job abnormally terminates. If this occurs, these messages will appear as output.

1 All allocation/termination messages will appear (default value).

While parameters other than TYPRUN=HOLD can be inserted in the operand field on the job card, their use is not generally recommended. MSGCLASS can be used to force system output to a special printer (see SYSOUT discussion). The CLASS, PRTY, and TIME parameters are ignored. The parameters which can be coded on the EXEC card should usually be coded there, as their effect is more precise.

5.3.2 COMPLETION CODES FOR INSUFFICIENT CPU OR I/O JOB TIMES

Completion codes that will be returned to the user, if the job exceeds his CPU or I/O time estimates, are shown in Table 5.3-3.

Table 5.3-3. Completion Codes

Computer	CPU Time Completion Code	I/O Time Completion Code
S/360-95	322	F22
S/360-75	322	None
S/360-65	322	None

On printouts received, both CPU and I/O time will be given to allow refinement of original estimates. I/O time will also be broken down by device type (tape, disk, drum, etc.) so that the components of the total are available. Note that on models 65 and 75, exceeding the estimated I/O time will not result in job termination.

5.3.3 PRIORITIES

As discussed in paragraphs 2.2.7 and 18.3.1 of this User's Guide, the job stream manager utilizes the CPU and I/O time punched into the JOB card to establish priorities.

The larger of these two numbers is compared with an internal table to establish the priority for the job. The shorter the job, the higher its priority. The values used are shown in Table 18.3-3.

5.3.4 CLASSES

The job stream manager assigns a class to each job, depending on the resources (core and I/O units) required. Table 18.3-1 shows the breakdown used by the S/95. There is a separate job queue for each class. As the initiators pull jobs from specific queues and in a specific order (Table 18.3-2 shows this order for S/95), the class into which a job is placed will affect its scheduling.

5.4 JOBLIB AND STEPLIB CARDS

The execution of any program or load module not in the system library (SYS1.LINKLIB, etc.) requires that a JOBLIB or STEPLIB card be used to identify that library by data set name. For example:

```
//JOBLIB  DD   DSNAME=SYS2.AUTOFLOW,DISP=SHR
```

The JOBLIB card causes the library to be used for every step in the job (as contrasted to the STEPLIB card described below).

The JOBLIB card must always follow the JOB card.

To define a private library for a particular step, the user must provide a DD statement that contains the special ddname STEPLIB. (Note: This DD statement can appear in any position among the DD statements for that step, as contrasted to a JOBLIB card which must be placed immediately after the JOB card.)

A STEPLIB DD statement can appear in a cataloged procedure and can be referred to by, or passed to, other steps of the same job.

The JOBLIB DD statement need not appear in a job in order to use the STEPLIB DD statement.

If both JOBLIB and STEPLIB DD statements appear in a job, the JOBLIB definition is ignored for any step that contains the STEPLIB definition. If the user wants the JOBLIB definition ignored, but the step does not require use of another private library, the system library must be defined on the STEPLIB DD statement. For example, where SYS1.LINKLIB is the system library, the code is:

```
//stepname EXEC ......
//STEPLIB DD DSN=SYS1.LINKLIB,DISP=SHR
```

See subsections 8.10 and 8.11 for a comprehensive description of JOBLIB and STEPLIB.

For more detail, refer to the IBM manuals, Job Control Language User's Guide (GC28-6703) and Job Control Language Reference (GC28-6704).

5.5 EXECUTE (EXEC) CARD

5.5.1 GENERAL DISCUSSION

This card is always the first card after the JOB card (except when a JOBLIB card is used). When more than one job step is used (for example: compile, link-edit, and execute), the EXEC card is the first card in each step.

The EXEC card may or may not have other entries added to it, depending on the circumstances, as explained under individual descriptions of cataloged procedures and various programs and in subsection 5.5.3.

Although the name field of the EXEC card may be left blank, this name (chosen by the user) is so often required as a stepname that it is always good programming practice to use one. The job step name must be used if later control statements refer to the job step in any way. Further, this step name is used by the system to return CPU and I/O time to the user and as step identification with certain diagnostics. Each stepname in a job must be unique. The makeup of the stepname is the same as that for the JOB card, i.e., one to eight alphameric or national characters (@,$,#).

5.5.2 EXECUTING PROGRAMS AND CATALOGED PROCEDURES

The principal function of the EXEC statement is to identify the program to be executed or the cataloged procedure to be used. Programs to be executed can reside in three types of libraries, as follows:

1. The system library. This is a partitioned data set (PDS), named SYS1.LINKLIB, which is used to store frequently used programs. To execute a program that resides in the system library, the user should code:

   ```
   //step EXEC PGM=progname
   ```

   where progname is the name of the program.

2. Private libraries. These are partitioned data sets which store groups or programs not used frequently enough to warrant their inclusion in the system library. These private libraries must be identified on a JOBLIB or STEPLIB statement.

3. Temporary libraries. These are temporary partitioned data sets created to store a program until it is used in a later step of the same job. To execute a program from a temporary library, the user should code:

   ```
   //step EXEC PGM=*.stepname.ddname
   ```

where stepname and ddname are the names of the job step and the DD statement where the temporary library was created (usually LINK.SYSLMOD). Note that the DD statement referred to must contain the member name of the program as well as the DSNAME of the library.

To execute a program that resides in a private library, the same format is used as for a program residing in the system library.

Instead of executing a particular program, a job step may use a cataloged procedure. A cataloged procedure can contain control statements for several steps, each of which executes a particular program. Cataloged procedures are members of a library (PDS), named SYS1.PROCLIB and (model 95 only) SYS2.USERPROC. To request a cataloged procedure, the user should code:

```
//step EXEC procname
     or
//step EXEC PROC=procname
```

where procname is the member name associated with the cataloged procedure.

5.5.3 EXEC CARD PARAMETERS

Several parameters can be included on EXEC cards. Users of M&DO computers will find the PARM, COND, and REGION parameters useful or necessary. Many cataloged procedures supply them. When adding or overriding EXEC card parameters of cataloged procedures, certain precautions must be taken. Cataloged procedure steps must be overridden in order, i.e., all overrides to STEP1 must be coded on your EXEC card before any overrides of STEP2. Other precautions are mentioned in paragraphs 5.5.3.1 and 5.5.3.3.

5.5.3.1 PARM

The PARM parameter is used to pass options to a program. The program must be designed to accept the PARM information; otherwise, the field is ignored. When overriding a cataloged procedure, the overriding PARM field will completely replace the one in the cataloged procedure. Those options not stated will revert to the defaults (chosen at system generation time) which are not necessarily those stated in the cataloged procedure. In multi-step procedures overrides should be coded as "PARM.stepname" rather than just "PARM", as the latter format will only override the PARM field in the first step of the cataloged procedure.

5.5.3.2 COND

The COND field on the EXEC card is used to conditionally execute or skip a job step, depending on the completion and condition code set by a preceding step. The code returned by the preceding step is compared to the number specified in the COND statement. If the comparison is satisfied, the step is bypassed. Use of the format COND=(n,cp) tests all preceding steps. Use of the format COND=(n,cp,stepname) tests a particular step. Up to eight different tests can be made against the condition codes by coding COND= ((n,cp,STEP1),(n,cp,STEP2),(...)). There is only one condition code returned per job step.

COND=EVEN and COND=ONLY test for an abnormal termination (ABEND) of a previous step. They can be used in conjunction with the COND= tests described in the above paragraph. However, COND=EVEN and COND=ONLY cannot be used in the same job step. COND=EVEN execute this step even if one or more preceding job steps abnormally terminated. COND=ONLY execute this step only if one or more of the preceding job steps abnormally terminated. They are useful for steps which list the contents of intermediate files or restore things which might have been left in disarray by the abnormal termination. They will not be effective if a JCL error or a condition which set a completion code of n22 caused the ABEND, or if any return code tests specified in the job step have been satisfied.

5.5.3.3 REGION

The REGION parameter must be used if a step requires more core than the default provided by the reader-interpreter procedure (80K). Most cataloged procedures have the proper region specified for each step. The use of REGION in a JOB card[1] will override the specification made in an EXEC card and therefore must be the maximum used by any step. The use of REGION on a step basis results in the best use of system resources.

Lack of sufficient REGION will result in an ABEND with a code of 804 or 80A. Since some system tasks, such as access method routines and buffers, require core in a user's region, a REGION parameter must be large enough to include these tasks, as well as the specified program. 8K is usually sufficient for these system routines.

NOTE: The result of the COND parameter when specified on the JOB card differs from that when specified on the EXEC card. If a JOB card return code test is satisfied subsequent steps are bypassed and the job is terminated. COND=EVEN and COND=ONLY cannot be used on the JOB card.

1 The REGION specification may no longer be used on the JOB card on all M&DO 360 computers (see subsection 16.2).

5.6 THE DATA DEFINITION (DD) STATEMENT

5.6.1 GENERAL DISCUSSION

For general instructions on preparing the DD control cards, refer to the IBM manuals, Job Control Language User's Guide, GC28-6703 and Job Control Language Reference, Form GC28-6704. The following paragraphs present an overview of the DD card, along with the supplementary information unique to GSFC and the M&DO facilities. More detailed information on specific techniques and parameters may be found in Section 17.

5.6.2 THE DD CARDS

Data sets used by processing programs must be represented by DD statements in the job stream. The DD statements pertaining to a particular job step follow the EXEC statement associated with that step. A DD statement must contain the term DD in its operation field. Although all parameters in the DD statement's operand field are optional, a blank operand field is invalid, except when overriding DD statements defining concatenated data sets. (See paragraph 19.2.3 and "Overriding, Adding, and Nullifying Parameters on DD Statements" in Part III of the IBM Job Control Language User's Guide.)

The general form of the DD statement is:

```
//ddname DD operands
```

Like the EXEC card, the DD operand consists of positional and keyword parameters (refer to subsection 5.2.3 of this guide).

5.6.3 CONTINUATION OF DD CARDS

If the operand of a DD card will not fit on one card, it may be continued. This is explained in subsection 5.2.3 of this guide. (Note: In this User's Guide, because of space limitations on the printed page, the examples of control cards are necessarily shorter than the full 71 columns allowed on an actual card, but the examples follow all of the above rules for continuing statements.)

5.6.4 ABBREVIATIONS IN DD STATEMENTS

As explained in the IBM Job Control Language User's Guide manual, certain abbreviations are allowed in the DD statements, and are useful in saving space on a control card. (Note: These abbreviations are not allowed in utility control cards.)

In this User's Guide, both the abbreviated form and the full form are used in examples of DD cards. The two parameters with their abbreviations are:

- VOLUME or VOL
- DSNAME or DSN

For example, both of the following versions of a statement are recognized by the system:

```
//SOURCE.COMPILE    DD   DSNAME=TAPOUT,UNIT=9TRACK,VOLUME=SER=tapeid,
//                       DCB=(RECFM=F,BLKSIZE=80,DEN=2),LABEL=(,BLP),
//                       DISP=(NEW,KEEP)
```

or

```
//SOURCE.COMPILE    DD   DSN=TAPOUT,UNIT=9TRACK,VOL=SER=tapeid,
//                       DCB=(RECFM=F,BLKSIZE=80,DEN=2),
//                       LABEL=(,BLP),DISP=(NEW,KEEP)
```

5.6.5 BACKWARD REFERENCES (*.name.name)

The user may save time and avoid errors in copying information onto a DD card from an earlier DD card in the same job by writing a backward reference in the form of:

```
*.stepname.ddname
```

For example, assume that the user has several job steps, and that in the one named STEP2 there is a SYSPRINT DD card containing complete DCB information that will be used in exactly the same manner in STEP4. Therefore, in STEP4 he writes this card:

```
//SYSPRINT     DD   SYSOUT=A,DCB=*.STEP2.SYSPRINT
```

Or assume that the user has this statement in STEP1:

```
//CARD4        DD   DSN=&&WORK1
```

and he desires to refer in several later job steps to the same data set name.

He may, instead of repeating DSN=&&WORK1, write:

```
DSN=*.STEP1.CARD4
```

If the earlier DD card is in the same job step, he writes:

DSN=*.ddname

If the DD statement referred to is in a cataloged procedure within the system procedure library, and the user desires to refer to the statement from outside the procedure, he must give the step name that invoked the procedure, the name of the step within that procedure, and the name in the DD statement within that step of the procedure:

*.jobstepname.procstepname.ddname

5.6.6 PARAMETERS IN THE OPERAND FIELD OF THE DD STATEMENT

The following parameter descriptions should be used as a guide to the use of parameters in the operand field of the DD statement. They are not complete; therefore, the user is referred to the IBM Job Control Language Reference (GC28-6704) manual for a more detailed description. Section 17 of this document contains further discussion of some of these parameters as they apply to M&DO computer use. The IBM manual, Job Control Language User's Guide (GC28-6703), describes techniques of using JCL to create and retrieve data sets.

In the following descriptions, the format of the parameter is shown first, followed by a discussion of its more important subparameters and comments, if any.

The symbols used in displaying the parameter formats are as follows:

{ } = Choose one

[ ] = Optional; if more than one line is enclosed, choose one or none.

5.6.6.1 Data Control Block

Each data set that is to be read or written must have a data control block associated with it. The data control block is originally constructed in the processing program by a DCB macro instruction. This data control block can be completed when the DCB macro instruction is issued, or at execution time through the DCB parameter on the DD statement and the data set label, if one exists. The format of the DCB parameter is as follows:

```
{DCB=(list of attributes)                                          }
{DCB=(  {dsname                                    }  [,list of attributes])}
{       {*.ddname                                  }                }
{       {*.stepname.ddname                         }                }
{       {*.stepname.procstepname.ddname            }                }
```

5.6.6.1.1 Rules for Coding

a. Separate each DCB keyword subparameter with a comma.

b. If the DCB parameter value consists of only one keyword subparameter, a data set name, or a backward reference, the user need not enclose it in parentheses.

c. All DCB subparameters, except BLKSIZE and BUFNO, are mutually exclusive with the DDNAME parameter; therefore, when the DDNAME parameter is coded, the user should not code any DCB subparameters except BLKSIZE and BUFNO. The DCB subparameters BLKSIZE and BUFNO have meaning when coded with the DDNAME parameter.

5.6.6.1.2 Completing the Data Control Block

When more than one source is used to complete the data control block, a merging process takes place. First, information coded with the DCB macro instruction is placed in the data control block. Then, information coded on the DD statement is placed in unfilled sections of the data control blocks. Finally, information in the data set label, if one exists, is placed in still unfilled sections of the data control block. DCB information may also be provided by default options assumed in the OPEN macro instruction and by the user's program, either before the data set is opened (by using the DCBD macro instruction) or in the DCB exit routine. Refer to the chapter "Interface With the Operating System" in Supervisor and Data Management Services (GC28-6646) and Supervisor and Data Management Macro Instructions (GC28-6647) for details.

5.6.6.1.3 DCB Macro Instruction

The DCB macro instruction includes information about the data that is unlikely to change each time the processing program is executed. Also, it includes any information that is not related to the DCB parameter and the data set label (e.g., MACRF, DDNAME, EXLST).

5.6.6.1.4 DCB Parameter

The DCB parameter is coded on the DD statement and includes all the information that is not specified by any other source. How to specify DCB information on the DD statement is described in 5.6.6.1.6 below.

5.6.6.1.5 Data Set Label

If the data set already exists and has standard labels, certain information is contained in the label that can be used to complete the data control block. For tape, the data set label can contain the data set's record format, block size, logical record length, tape recording density, and for 7-track tape, tape recording technique. For direct access, the data set label can contain the data set's organization, record format, blocksize, logical record length, and if the data contains keys, the key length and relative key position.

5.6.6.1.6 Specifying DCB Information on the DD Statement

The DCB parameter must be coded on the DD statement unless the data control block is completed by other sources. There are several ways of specifying DCB information on the DD statement. The user can:

a. Supply all pertinent DCB keyword subparameters on the DD statement.

b. Tell the system to copy DCB information from the data set label of an existing cataloged data set.

c. Tell the system to copy DCB information from an earlier DD statement in the same job.

If the user is extending a data set which has standard label, the blocksize information in the data control block must agree with the blocksize specified in the data set label. Conflicting information may make the data set unusable by later jobs. The user should especially take care when extending sequential data sets (tape or direct access) with DISP=MOD or adding or replacing members on a partitioned data set.

5.6.6.1.7 Supplying DCB Keyword Subparameters

The DCB information required to complete the data control block can be listed as keyword subparameters in the DCB parameter; subparameters are separated by commas. If the processing program and the DCB parameter supply the same subparameter, the subparameter on the DD statement is ignored. Valid DCB keyword subparameters and the values that can be assigned to them are listed in Job Control Language Reference (GC28-6704), under "Glossary of DCB Subparameters."

5.6.6.1.8 Copying DCB Information From a Data Set Label

To save time in coding the DCB parameter, the user can tell the system to copy the DCB information from the data set label of a cataloged data set on a currently mounted direct-access volume. The data set must have standard labels. A permanently resident volume is the most likely place from which to copy such information because it is always mounted. The user should code in the DCB parameter the data set name of the cataloged data set. The name coded cannot contain special characters, except for periods used in a qualified name.

The following DCB keyword subparameters can be copied from the data set label: DSORG, RECFM, OPTCD, BLKSIZE, LRECL, KEYLEN, and RKP. The volume sequence number and expiration date of the cataloged data set are also copied unless these are specified in the DD statement. If the user codes any DCB keyword subparameters following the name of the cataloged data set, these subparameters override any of the corresponding subparameters that were copied. Valid DCB keyword subparameters and the values that can be assigned to them are listed in Job Control Language Reference (GC28-6704), under "Glossary of DCB Subparameters."

5.6.6.1.9 Copying DCB Information From an Earlier DD Statement

Another way to save time in coding the DCB parameter is to tell the system to copy the DCB information from an earlier DD statement in the same job. The earlier DD statement can be contained in the same job step, an earlier job step, or a cataloged procedure step. If the user codes any DCB keyword subparameters following the reference to the DD statement, these subparameters override any of the corresponding subparameters that were copied. If the DD statement defines an existing data set and contains the DCB parameter, the system copies those subparameters from the earlier DD statement that were not previously specified for the existing data set. Valid DCB keyword subparameters and the values that can be assigned to them are listed in Job Control Language Reference (GC28-6704), and Supervisor and Data Management Macros (GC28-6647).

5.6.6.2 DEN

It is good programming practice to specify tape density in the DCB information, rather than to take the default option, because this eliminates errors in remembering default options. Also, default density may vary from one computer to another because of system generation options, while these differences may be unknown to the user. Write:

DCB=DEN=n

where n is the desired density, as shown below.

Tape densities, used on M&DO computers, and the code for their use, are as follows:

Tape Drive	BPI if DEN=0	BPI if DEN=1	BPI if DEN=2	BPI if DEN=3
7-Track	200	556	800	Not Allowed
9-Track	Not Allowed	Not Allowed	800	1600

5.6.6.3 TRTCH

The tape recording techniques parameter is used to differentiate the various recording techniques which may be used with 7-track tapes. Allowable values are:

C - Data Conversion

When reading tape, four 6-bit characters from tape fill three 8-bit bytes in core. When writing, three 8-bit bytes from core fill four 6-bit characters. All M&DO 7-track drives support data conversion. At other centers, the user should code UNIT=2400-2 to ensure that the 7-track drive supports data conversion.

E - Even Parity

Most 1400 series tapes and formatted tapes on 7000 series computers were written in even parity.

T - Translate

BCD-to-EBCDIC translation is to be performed when reading; EBCDIC-to-BCD translation is to be performed when writing.

ET - Both Even Parity and Translation

In the absence of the TRTCH parameter, the 7-track tape is read/written in odd parity, without conversion or translation.

The result - When read from tape to core the 6 bit character will be converted to an 8 bit byte by having zeros stored in its two high order bits. When read from core to tape an 8 bit byte will be converted to a 6 bit character by having its two high order bits truncated.

5.6.6.4 Disposition

The DISP parameter describes to the system the status of a data set and indicates what is to be done with the data set after termination of the job step that processes it or at the end of the job. The user can indicate in the DISP parameter one disposition to apply if the step terminates normally after execution and another to apply if the step terminates abnormally (conditional disposition). The format of the DISP parameter is as follows:

```
DISP=( [NEW]    [,DELETE ]    [,DELETE ] )
       [OLD]    [,KEEP   ]    [,KEEP   ]
       [SHR]    [,PASS   ]    [,CATLG  ]
       [MOD]    [,CATLG  ]    [,UNCATLG]
                [,UNCATLG]
                [,       ]
```

5.6.6.4.1 Rules for Coding

a. If only the first subparameter is coded, the user need not enclose it in parentheses.

b. If the data set is new, the user can omit the subparameter NEW. However, if the user specifies a disposition or conditional disposition, he must code a comma to indicate the absence of NEW.

c. The user can omit the DISP parameter if a data set is created and deleted during a job step.

d. If the user does not want to change the automatic disposition processing performed by the system, he need not code the second subparameter. (When the second subparameter is not coded, the system automatically keeps data sets that did exist before the job and automatically deletes data sets that did not exist before the job.) If the user omits the second subparameter and codes a conditional disposition, he must code a comma to indicate the absence of the second subparameter.

e. The DISP, SYSOUT, and DDNAME parameters are mutually exclusive parameters; therefore, when SYSOUT or DDNAME is coded, do not code the DISP parameter.

5.6.6.4.2 Specifying the Data Set's Status

A data set is either a new data set or an existing data set. What the user plans to do with the data set determines which status he codes as the first subparameter of the DISP parameter. There are four different states that can be coded in this position. This subparameter allows the user to tell the system: the data set is to be created in the job step -- NEW; the data set existed before this job step -- OLD; the data set can be used by other concurrently executing jobs -- SHR; the data set is to be lengthened with additional output -- MOD.

a. Specifying NEW as the Data Set's Status

   Specifying NEW as the first subparameter of the DISP parameter tells the system that the data set is to be created in the job step and may be used by the processing program to contain output data. If the user omits the subparameter NEW, the system assumes the data set is to be created in the job step. When the status of a data set is NEW, the user must code on the DD statement all of the parameters necessary to define the data set.

b. Specifying OLD as the Data Set's Status

   Specifying OLD as the first subparameter of the DISP parameter tells the system that the data set existed before this job step. The data set will not be allocated to another JOB until this JOB terminates. Code OLD only when you want exclusive use of the data set, e.g., when you are updating the data set.

c. Specifying SHR as the Data Set's Status

   Specifying SHR as the first subparameter of the DISP parameter tells the system that the data set resides on a direct-access volume and other jobs that are executing concurrently with this job step may simultaneously use (share) the data set. When SHR is specified, any job step that uses the data set should only read the data set.

   If the user codes DISP=(SHR,DELETE) the system assumes OLD instead of SHR. Once SHR is specified for a data set, every reference to that data set within the job must be specified by SHR, or the data set can no longer be used by concurrently excuting jobs.

d. Specifying MOD as the Data Set's Status

Specifying MOD as the first subparameter of the DISP parameter tells the system that when the data set is opened for output, the read/write mechanism is to be positioned after the last record in the data set. MOD is specified when the user wants to add records to a data set with sequential, indexed sequential, or partitioned organization. MOD should not be specified for data sets with direct organization. When MOD is specified and the number of volumes required to lengthen the data set may exceed the number of units requested, a volume count should be specified in the VOLUME parameter. This ensures that the data set can be extended to new volumes.

When MOD is specified, the system first assumes the data set exists. However, if the system cannot find volume information for the data set -- on the DD statement, in the system catalog, or passed with the data set from a previous step -- the system then assumes that the data set does not exist and the data set is created for the job step. If the NEW data set is to be written to disk, space allocation must appear on the DD card. If space has been requested on the DD card and the data set is not NEW, the allocation request will be ignored. Specifying MOD for a new sequential data set causes the read/write mechanism to be positioned after the last record in the data set each time it is opened for output. MOD is often used for temporary passed data sets in those cataloged procedures which may be used repetitively within a job. For instance, a programmer may wish to compile both an ALC module and a FORTRAN module, load them together, and execute the load module. If the object module output of the compiler steps of the cataloged procedures specified NEW, the programmer would be forced to override this parameter for every procedure but the first. MOD works in either case.

If MOD is specified and volume information exists, the first volume of a multivolume data set will be mounted unless DEFER is specified in the UNIT parameter or (for tape data sets only) the VOLUME=REF parameter is used.

When a user lengthens a data set that has standard labels, DCB information in the data control block must agree with the DCB information contained in the data set label. Conflicting DCB information, specifically conflicting block sizes, may make the data set unusable by later jobs. Therefore, the DCB information contained in the data set label should not be coded on the DD statement. If this DCB

information is coded in the DCB macro instruction, it must agree with the information contained the data set label.

If a user extends a data set that has Fixed-Block spanned (FBS) records, and the last block was a truncated one, an end-of-data set condition occurs when the truncated block is encountered. If an attempt is made to read the data set backward on magnetic tape, processing is terminated immediately (with an end-of-data set condition) upon reading the truncated block.

5.6.6.4.3 Specifying a Disposition for the Data Set

The second subparameter of the DISP parameter tells the system what is to be done with the data set after normal termination of the job step. There are five dispositions that can be specified for a data set. These dispositions allow the user to:

- Delete a data set -- DELETE.
- Keep a data set -- KEEP.
- Pass a data set to a later job step -- PASS.
- Catalog a data set -- CATLG.
- Uncatalog a data set -- UNCATLG.

When the second subparameter is not coded, data sets that existed before the job continue to exist and data sets that were created in the job step are deleted. If the user creates a nontemporary data set in the job and assigns a disposition of PASS to it, the data set is deleted at termination of the job step that receives the passed data set; he does not assign a disposition to it. (The passed nontemporary data set is deleted at job termination if the data set is never received by a later job step.)

The system ignores the disposition the user has coded and automatically keeps existing data sets and deletes new data sets when the step is abnormally terminated before the step begins execution, e.g., primary direct-access space cannot be obtained.

Sometimes the system does not perform disposition processing. The system does not disposition processing of data sets when:

- The job step is bypassed because of an error that is found during interpretation of control statements, e.g., a control statement containing errors is read.
- The job step is bypassed because a return code test is satisfied.
- The job step makes a nonspecific request for a tape volume and the data set is never opened. There is one exception to this: if the data set is defined as a new generation data set, the system performs the requested disposition.
- The job step requests that the mounting of a direct-access volume be deferred and the data set is never opened.

Except for the cases mentioned above, the specified disposition is in effect for the data set if the job step terminates normally or abnormally and the user has not specified a condition disposition as the third sub-parameter of the DISP parameter.

a. Specifying DELETE as the Disposition

Specifying DELETE tells the system that the user wants the data set's space on the volume released. DELETE is the only valid disposition that can be specified for a data set assigned a temporary name or no name. If the data set resides on a tape volume, the tape is rewound and the volume becomes available for use by other data sets at the end of the job step. If the data set resides on a direct-access volume, the system removes the volume table of contents entry associated with the data set and the data set's space is available for use by other data sets at the end of the job step. However, if the direct-access data set's expiration date or retention period has not expired, the system does not delete the data set. The user can use the IEHPROGM utility program to remove the volume table of contents entry.

If the data set is cataloged, its entry in the system catalog is also removed, provided the system obtained volume information for the data set from the catalog, i.e., the VOL=SER= and UNIT=parameters were not coded on the DD statement. If the system did not obtain volume information from the catalog,

the data set is still deleted, but its entry in the catalog remains. In this case, the user may use the IEHPROGM utility program to delete the entry.

b. Specifying KEEP as the Disposition

Specifying KEEP tells the system that the user wants the data set kept intact.

c. Specifying CATLG as the Disposition

Specifying CATLG tells the system that the user wants the system to create an entry in the system catalog that points to this data set. The disposition of CATLG also implies a disposition of KEEP. Once the data set is cataloged, the user can retrieve the data set in later job steps and jobs by coding the DSNAME parameter and a status of other than NEW in the DISP parameter. The VOL=SER= and UNIT= parameters are not required.

If the data set's name is enclosed in apostrophes, the data set must not be assigned a conditional disposition of CATLG. If the data set has a qualified name, e.g., A.B.C., the user must have created all but the lowest level of the name as indexes in the catalog before asking that the system catalog the data set. This is done using the IEHPROGM utility program.

d. Specifying UNCATLG as the Disposition

Specifying UNCATLG tells the system that the user wants the data set's entry in the system catalog removed. UNCATLG does <u>not</u> tell the system to delete the data set. Later jobs that use this data set must provide on the DD statement all of the parameters necessary to define the data set.

e. Specifying PASS as the Disposition

Specifying PASS as the second subparameter of the DISP parameter tells the system that the data set is to be passed after it is used in a job step. This permits another job step in the same job to use this data set without having retrieval and disposition processing done by the system. The user continues to code PASS each time the data set is

referred to until the last time it is used in the job. At this time, the user assigns it a final disposition. If he does not assign the data set a final disposition, the system deletes the data set if it was created in the job and keeps the data set if it existed before the job.

When the data set is not in use, the volume that contains the passed data set remains mounted; therefore, the user need not code RETAIN in the VOLUME parameter of a DD statement that specifies a disposition of PASS. If the system must remove the volume that contains the passed data set, it ensures through messages to the operator that the volume is remounted before the data set is used again.

When a subsequent job step wants to use the passed data set, the user must include a DD statement for the step. On this DD statement, he must always code the DSNAME and DISP parameters, as follows:

- The DSNAME parameter identifies the data set. The user should either code the data set's name or make a backward reference to any earlier DD statement in the job that defines the data set.

- The DISP parameter specifies the data set's status and disposition. (If a later step is to use this data set, the user should specify a disposition of PASS; if this is the last job step that uses this data set, the user should specify the data set's final disposition.)

The other parameters the user may code are UNIT, LABEL, and DCB. The user should code:

- The UNIT parameter if he wants more than one device allocated to the data set.

- The LABEL parameter if the data set does not have standard labels.

- The DCB parameter if the data set does not have standard labels and the first DD statement that defines the passed data set contains the DCB parameter.

If several data sets used in the job have the same name, the user can only pass one of these data sets at a time. A job step must refer to a passed data set and assign a disposition of other than PASS to the data set before another data set with the same name can be passed.

5.6.6.4.4 Specifying a Condition Disposition for the Data Set

The third subparameter of the DISP parameter tells the system what is to be done with the data set if the step abnormally terminates. If the user does not specify a conditional disposition and the step abnormally terminates, the system uses the disposition specified as the second subparameter of the DISP parameter to determine what is to be done with the data set. (There are a few exceptions and they are noted in paragraph 5.6.6.4.3.) If a passed data set has not been received and a job step abnormally terminates, the passed data set assumes the conditional disposition specified the last time it was passed. In this case, conditional disposition processing is done at job termination, not at step termination.

There are four conditional dispositions. When a job step abnormally terminates, these conditional dispositions allow the user to:

- Delete a data set -- DELETE.
- Keep a data set -- KEEP.
- Catalog a data set -- CATLG.
- Uncatalog a data set -- UNCATLG.

Note: A scratch volume will be rewound, unloaded, and a KEEP message issued to the operator during abnormal termination of a job step when: (1) a temporary data set written on the scratch volume has been assigned a non-temporary name, and (2) a conditional disposition of KEEP has been assigned to the data set.

5.6.6.5 Data Set Name

When the user creates a data set, he should use the DSNAME parameter to assign a name to the data set. The data set name is part of the information stored with the data set on a labeled volume. Later, when another job step or job wants to use the data set, it identifies the data set name in the DSNAME parameter; the system uses the data set name to locate the data set on the volume. On an unlabeled (tape) volume, the data set sequence number in the LABEL parameter is used to locate the data set.

How the user codes the DSNAME parameter depends on the type of data set and whether the data set is nontemporary or temporary. Note that the disposition should correspond to the temporary or nontemporary status implied in the DSNAME parameter.

The format of the DSNAME parameter is as follows:

```
{DSNAME} = dsname
{DSN   }   {dsname (member name)                }
           {dsname (generation number)          }
           {dsname (area name)                  }
           {&&dsname                            }
           {&&dsname (member name)              }
           {&&dsname (area name)                }
           {*.ddname                            }
           {*.stepname.ddname                   }
           *.stepname.procstepname.ddname
```

5.6.6.5.1 Rules for Coding

a. An unqualified data set name may consist of one to eight characters. The first character must be an alphabetic or national (@,$,#) character; the remaining characters can be any alphameric or national characters, a hyphen, or an overpunched ampersand zero (12-0 punch). A temporary data set name can consist of one through eight characters, excluding the ampersands; the first character following an ampersand must be an alphabetic or national character.

b. A qualified name may consist of up to 44 characters, including periods. For each eight characters or less there must be a period, and the character following a period must be an alphabetic or national (@,$,#) character.

c. The user need not code the DSNAME parameter if the data set is created and deleted in the job, i.e., if the data set is temporary.

d. The DSNAME and DDNAME parameters are mutually exclusive parameters; therefore, when the DDNAME parameter is coded, the user should not code the DSNAME parameter.

Sometimes it may be necessary or desirable to specify a data set name that contains special characters. If the name contains special characters, the user must enclose the name in apostrophes (5-8 punch), e.g., DSNAME='DAT+5' If one of the special characters is an apostrophe, the user must identify it by coding two consecutive apostrophes (two 5-8 punches) in its place, e.g., DSNAME='DAY''SEND'. A data set name enclosed in apostrophes can consist of one through 44 characters.

There are cases when the user's data set name must contain required special characters which tell the system something about the data set (e.g., && in DSNAME=&&name are required special characters that tell the system that this is a temporary data set). In these cases, the data set name must not be enclosed in apostrophes because the system will not recognize the required special characters as having any special significance. The following data set names contain special characters that tell the system something about the data set and, therefore, cannot be enclosed in apostrophes:

- DSNAME=name (member name)
- DSNAME=name (area name)
- DSNAME=name (generation number)
- DSNAME=&&name
- DSNAME=*.stepname.ddname

The following rules should be observed:

- If the data set is to be cataloged, the data set name cannot be enclosed in apostrophes.
- If the data set name begins with a blank character, the data set is assigned a temporary data set name by the system.
- If the data set name ends with a blank character, the blank is ignored.
- If the only special character is a period or a hyphen, the user need not enclose the data set name in apostrophes.

5.6.6.5.2 Copying the Data Set Name From an Earlier DD Statement

The name of a data set that is used several times in a job, whether specified in the DSNAME parameter or assigned by the system, can be copied after its first use in the job. This allows the user to easily change data sets from job to job and eliminates assigning names to temporary data sets. To copy a data set name, the user should refer to an eariler DD statement that identifies the data set. When the earlier DD statement is contained in an earlier job step, the user should code DSNAME=*.stepname.ddname; when the earlier DD statement is contained in the same job step, he should code DSNAME=*.ddname; when the earlier DD statement is contained in a cataloged procedure step called by an earlier job step, he should code DSNAME=*.stepname.procstepname.ddname.

5.6.6.5.3 Creating or Retrieving a Nontemporary Data Set

If the data set is nontemporary, the user can identify:

- A permanent data set, by coding DSNAME=dsname.
- A member of a nontemporary partitioned data set, by coding DSNAME=dsname(member name).
- A generation of a nontemporary generation data group, by coding DSNAME=dsname(number).
- An area of a nontemporary indexed sequential data set, by coding DSNAME=dsname(area name).

5.6.6.5.4 Creating or Retrieving a Temporary Data Set

If the data set is temporary, the user can identify:

- A temporary data set, by coding DSNAME=&&dsname.
- A member of a temporary partitioned data set, by coding DSNAME=&&dsname(member name).
- An area of a temporary indexed sequential data set, by coding DSNAME=&&dsname(area name).

5.6.6.5.5 Nontemporary Data Sets

When a nontemporary data set is created, it is assigned a name in the DSNAME parameter and is assigned a disposition of KEEP or CATLG. (A data set assigned a disposition of KEEP may be assigned a disposition of CATLG by a later job step or job.) The name a user assigns to a nontemporary data set must be specified in the DSNAME parameter by all other steps and jobs that want to use the data set.

A nontemporary data set name can be either a unqualified or qualified name. An unqualified data set name consists of one through eight characters. The first character must be an alphabetic or national (@,#,$) character; the remaining characters can be any alphameric or national characters, a hyphen, or an overpunched ampersand zero (12-0 punch).

A qualified data set name consists of one through 44 characters (including periods), except when the qualified name identifies a generation data group. In this case, the data set name may consist of only one through 35 characters (including periods). For each eight characters or less there must be a period, and the first character of the name and the character following a period must be an alphabetic or national (@,#,$) character.

If the user assigns a qualified name to a data set that is to be cataloged, all but the lowest level of the name must already exist as indexes in the system catalog before he can request the system to catalog the data set. An index level is created by using the IEHPROGM utility program. Once the indexes are established, the data set can be cataloged.

When the user requests a data set that is cataloged on a control volume (e.g., DODS) other than the system catalog, the system attempts to mount this control volume if it is not already mounted. After the system obtains the pointer to this data set, the control volume may then be demounted by the system if the unit on which it was mounted is required by another volume. If the user plans to delete, uncatalog, or recatalog the data set, the volume must be mounted during disposition processing (at the end of the job step) in order for the pointer to be deleted or revised. The user can ensure that the volume remains mounted by requesting the operator to issue a MOUNT command for this volume before the job step is initiated. If the user does not use the MOUNT command to mount the volume, and if the volume is not mounted during disposition processing, then, after the job has terminated, he must use the IEHPROGM utility program to delete or revise the pointer in the control volume. (In order for the system to mount a control volume, the control volume must be logically connected to the system catalog. This is done using the CONNECT function of the IEHPROGM utility program, which is described in the IBM System/360 Operating System Utilities (GC28-6586) publication.)

5.6.6.5.6 Temporary Data Sets

Any data set that is created and deleted within the same job is a temporary data set. A DD statement that defines a temporary data set need not include the DSNAME parameter; the system generates one for the user. Temporary data set names should not be used for tape data sets, as they cause excessive printout on the operator's console. The user should use a permanent form of the DSNAME.

If the user includes the DSNAME parameter, the temporary data set name can consist of one through eight characters and is preceded by two ampersands (&&). The character following the ampersands must be an alphabetic or national (@,#,$) character; the remaining characters can be any alphameric or national characters. (A temporary data set name that is preceded by only one ampersand is treated as a temporary data set name as long as no value is assigned to it either on the EXEC statement for this job step when it calls a procedure, or on a PROC statement within the procedure.) If a value is assigned to it by one of these means, it is treated as a symbolic parameter. Symbolic parameters are discussed in Appendix A of Job Control Reference (GC28-6704).

The system generates a qualified name for the temporary data set, which begins with SYS and includes the jobname, the temporary name assigned in the DSNAME parameter, and other identifying characters.

If the user attempts to keep or catalog a temporary data set (he specifies a disposition of KEEP or CATLG in the DISP parameter), the system changes the disposition to PASS and the data set is deleted at job termination. However, this change is not made for a data set on a tape volume when the following conditions exist: (1) the data set is new; (2) the data set is not assigned a name; and (3) DEFER is specified in the UNIT parameter. The data set is deleted at job termination, but the system tells the operator to keep the volume on which the data set resided during the job.

5.6.6.5.7 Members of a Partitioned Data Set

A partitioned data set consists of independent groups of sequential records, each identified by a member name in a directory. When a user wants to add a member to a partitioned data set or retrieve a member, he must specify the partitioned data set name and follow it with the member name. The member name is enclosed in parentheses and consists of one to eight characters. The first character must be an alphabetic or national (@,$,#) character; the remaining characters can be any alphameric or national characters.

5.6.6.5.8 Generations of a Generation Data Group

A generation data group is a collection of chronologically related data sets that can be referred to by the same data set name. When a user wants to add a generation to a generation data group or retrieve a generation, he must specify the generation data group name and follow it with the generation number. The generation number is enclosed in parentheses and

the number is a zero or a signed integer. A zero represents the most current generation of the group; a negative integer represents an older generation; a positive integer represents a new generation that has not as yet been cataloged.

To retrieve all generations of a generation data group (up to 255 generations), the user should code only the group name in the DSNAME parameter and the DISP parameter.

A complete discussion of creating and retrieving generating data sets is contained in "Appendix D: Creating and Retrieving Generation Data Sets" in Job Control Language Reference (GC28-6704).

5.6.6.6 Postponing Data Set Definition

The DDNAME parameter allows the user to postpone defining a data set until later in the same job step. In the case of cataloged procedures, this parameter allows the user to postpone defining a data set in the procedure until the procedure is called by a job step.

The DDNAME parameter is most often used in cataloged procedures and in job steps that call procedures. It is used in cataloged procedures to postpone defining data in the input stream until a job step calls the procedure. (Procedures cannot contain DD statements that define data in the input stream, i.e., DD * or DD DATA statements.) It is used in job steps that call procedures to postpone defining data in the input stream on an overriding DD statement until the last overriding DD statement for a procedure step. (Overriding DD statements must appear in the same order as the corresponding DD statements in the procedure.)

The DDNAME parameter is coded as follows:

```
DDNAME=ddname
```

where ddname is the name of a following DD statement in the same job step that defines this data set.

5.6.6.6.1 Rules for Coding

a. The only parameters that can be coded with the DDNAME parameter are the DCB subparameters BLKSIZE and BUFNO.

b. The DDNAME parameter cannot appear on a DD statement named JOBLIB.

c. The user can code the DDNAME parameter up to five times in a job step or procedure step. However, each time the DDNAME parameter is coded, it must refer to a different ddname.

d. If the data set, which will be defined later in the job step, is to be concatenated with other data sets, the DD statements that define these other data sets must immediately follow the DD statement that includes the DDNAME parameter.

5.6.6.6.2 Coding the DDNAME Parameter

When the system encounters a DD statement that contains the DDNAME parameter, it saves the ddname of that statement. The system also temporarily saves the name specified in the DDNAME parameter so that it can relate

that name to the ddname of a later DD statement. Once a DD statement with that corresponding name is encountered, the name is no longer saved. For example, if the system encounters this statement:

```
//ABC      DD    DDNAME=JACK
```

the system saves ABC and, temporarily, JACK. Until the ddname JACK is encountered in the input stream, ABC defines a dummy data set.

When the system encounters a statement whose ddname has been temporarily saved, it does two things: (1) it uses the information contained on this statement to define the data set; and (2) it associates this information with the name of the statement that contained the DDNAME parameter. The value that appeared in the DDNAME parameter is no longer saved by the system. To continue the above example, if the system encounters this statement:

```
//JACK     DD    DSNAME=NIN,DISP=(NEW,KEEP),UNIT=2400
```

the system uses the data set name and the disposition and unit information to define the data set; it also associates the ddname of the statement that contained the DDNAME parameter with this information. In this example, the ddname used is ABC; the ddname JACK is no longer saved. The data set is now defined, just as it would be if the user had coded:

```
//ABC      DD    DSNAME=NIN,DISP=(NEW,KEEP),UNIT=2400
```

The system associates the ddname of the statement that contains the DDNAME parameter with the data set definition information. It does not use the ddname of the later statement that defines the data set. Therefore, any references to the data set, before or after the data set is defined, must refer to the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DD1      DD    DDNAME=LATER
               .
               .
               .
//LATER    DD    DSN=SET12,DISP=(NEW,KEEP),UNIT=2314,VOLUME=SER=G3SCR0,
//               SPACE=(TRK,(20,5))
               .
               .
               .
//DD12     DD    DSN=SET13,DISP=(NEW,KEEP),VOLUME=REF=*.DD1,
//               SPACE=(TRK,(40,5))
```

When the user wants to concatenate data sets, the unnamed DD statements must follow the DD statement that contains the DDNAME parameter, not the DD statement that defines the data set. The following sequence of control statements illustrates this:

```
//DDA      DD   DDNAME=DEFINE
//        DD   DSN=A.B.C,DISP=OLD
//        DD   DSN=SEVC,DISP=OLD,UNIT=2314,VOL=SER=G1SCR1
           .
           .
           .
//DEFINE   DD   *
           data
/*
```

The user can use the DDNAME parameter up to five times in a job step or procedure step. However, each time the DDNAME parameter is coded, it must refer to a different ddname.

5.6.6.7 Data Set Labels

Labels are used by the operating system to identify volumes and the data sets they contain, and to store data set attributes. If data set labels are present, they precede each data set on the volume. Data sets residing on direct-access volumes always have data set labels. These data set labels are contained in the volume table of contents at the beginning of the direct-access volume.

A data set label may be a standard or nonstandard label. Standard labels can be processed by the system; nonstandard labels must be processed by nonstandard label processing routines, which are not included in the M&DO systems. Data sets on direct-access volumes must have standard labels. Data sets on tape volumes should have standard labels, but can have nonstandard labels or no labels.

Tape label definitions and associated tape label processing are included in the Tape Labels (GC28-6680) publication. Direct-access label definitions and associated direct-access label processing are described in "Appendix A: Direct Access Labels" in the Supervisor and Data Management Services (GC28-6646) publication.

The LABEL parameter is coded as follows:

```
LABEL=([data set sequence number] [,SL  ]   [,,IN  ]   [,EXPDT=yyddd]  )
                                  [,NL  ]   [,,OUT ]   [,RETPD=nnnn ]
                                  [,BLP ]
                                  [,    ]
```

5.6.6.7.1 Rules for Coding

a. All the subparameters except the last subparameter in the LABEL parameter are positional subparameters. Therefore, if the user wants to omit a subparameter, he must indicate its absence with a comma.

b. If the only subparameter the user wants to specify is the data set sequence number, RETPD or EXPDT, he can omit the parentheses and commas and code LABEL=data set sequence number, LABEL=RETPD=nnnn, or LABEL=EXPDT=yyddd.

c. If the data set has standard labels, the user can omit the subparameter SL.

d. When the user is defining a data set that resides or will reside on a direct-access volume, only SL can be specified as the second subparameter.

e. The LABEL, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, if DDNAME or SYSOUT is coded, the user should not code the LABEL parameter.

5.6.6.7.2 When to Code the LABEL Parameter

The LABEL parameter must be coded if:

- The user is processing a tape data set that is not the first data set on the reel; in this case, he must indicate the data set sequence number.
- The data set labels are not standard labels; the user must indicate NL or BLP.
- The user wants to specify what type of labels a data set is to have when it is written on a scratch volume; he must indicate the label type.
- The data set is to be processed only for input or output and this conflicts with the processing method indicated in the OPEN macro instruction; the user must specify IN, for input, or OUT, for output. This option should be used when applicable for FORTRAN sequential data sets.
- The data set is to be kept for some period of time; the user must indicate a retention period (RETPD) or expiration date (EXPDT).

5.6.6.7.3 The Data Set Sequence Number Subparameter

When the user wants to place a data set on a tape volume that already contains one or more data sets, he must specify where the data set is to be placed, i.e., the data set is to be the second, third, fourth, etc., data set on the volume. The data set sequence number causes the tape to be positioned properly so that the data set can be written on the tape or retrieved.

The data set sequence number subparameter is a positional subparameter and is the first subparameter that can be coded. The data set sequence number is a 1- to 4-digit number. The system assumes 1, i.e., this is the first data set on the reel, if the user omits this subparameter or if you code 0.

When the user requests the system to bypass label processing (BLP is coded as the label type in the LABEL parameter) and the tape volume contains labels, the system treats anything between tapemarks as a data set. Therefore, in order for the tape with labels to be positioned properly, the data set sequence number must reflect all labels and data sets that precede the desired set. Section I of the Tape Labels (GC28-6680) publication illustrates where tapemarks appear.

5.6.6.7.4 The Label Type Subparameter

The label type subparameter tells the system what type of label is associated with the data set. The label type subparameter is a positional subparameter and must be coded second, after the data set sequence number subparameter. The user can omit this subparameter if the data set has standard labels.

The label type subparameter is specified as:

- SL -- if the data set has standard labels.
- NL -- if the data set has no labels.
- BLP -- if the user wants label processing bypassed.

SL is the only label type that can be specified for data sets that reside on direct-access volumes.

When SL is specified, or the label type subparameter is omitted and the data set has standard labels, the system can ensure that the correct tape or direct-access volume is mounted. When the user specifies NL or BLP, the operator must ensure that the correct tape volume is mounted. If the user specifies NL, the data set must have no standard labels.

For cataloged and passed data sets, label type information is not kept. Therefore, any time the user refers to a cataloged or passed data set that has other than standard labels, he must code the LABEL parameter and specify the label type.

BLP is not a label type, but a request to the system to bypass label processing. This specification allows the user to use a blank tape or overwrite a 7-track tape that differs from his current parity or density specifications. Bypass label processing is an option of the operation system.

Note for BLP: When the user requests the system to bypass label processing and the tape volume has labels, the system treats anything between tapemarks as a data set. Therefore, in order for a tape with labels to be positioned properly, the data set sequence number subparameter of the LABEL parameter must be coded and the subparameter must reflect all labels and data sets that precede the desired data set. Section I of the Tape Labels publication illustrates where tapemarks appear.

The label type subparameter can also be specified when the user makes a nonspecific volume request for a tape volume (i.e., no volume serial numbers are specified on the DD statement) and he wants the data set to have a certain type of label. If the volume that is mounted does not have the corresponding label type he desires, he may be able to change the label type.

When you specify NL and the operator mounts a tape volume that contains standard labels, the user may use the volume, provided: (1) the expiration date of the existing date of the existing data set on the volume has passed; and (2) the existing data set on the volume is not password protected; and (3) he makes a nonspecific volume request. All of these conditions must be met. If they are not, the system requests the operator to mount another tape volume.

When the user specifies SL and the operator mounts a tape volume that contains other than standard labels, the system requests the operator to identify the volume serial number and its new owner before the standard labels are written.

5.6.6.7.5 The IN and OUT Subparameters

The basic sequential access method (BSAM) permits a specification of INOUT or OUTIN in the OPEN macro instruction as the processing method. If the user has specified either of these processing methods in the OPEN macro instruction and wants to override it, he may be able to do so by coding either the IN or OUT subparameter. For FORTRAN users, the IN and OUT subparameters provide a means of specifying how the data set is to be processed, i.e., for input or output.

When INOUT is specified in the OPEN macro instruction and the user wants the data set processed for input only, he can specify the IN subparameter. When the IN subparameter is coded, any attempt by the processing program to process the data set for output is treated as an error. If the user does not override the INOUT specification for tape volumes, the operator must insert a ring negating the file protection feature.

When OUTIN is specified in the OPEN macro instruction and the user wants the data set processed for output only, he can specify the OUT subparameter. When the OUT subparameter is coded, any attempt by the processing program to process the data set for input is treated as an error.

The IN and OUT subparameters are positional subparameters. If either is coded, it must appear as the fourth subparameter, after the data set sequence number subparameter, the label type subparameter, and the PASSWORD subparameter, or the commas that indicate their absence.

5.6.6.7.6 The RETPD and EXPDT Subparameters

When it is necessary that a data set be kept for some period of time, the user can tell the system how long it is to be kept when he creates the data set. As long as the time period has not expired, a data set that resides on a direct-access volume cannot be deleted by or overwritten by another job step or job. (If it is necessary to delete such a data set, the user can use the IEHPROGM utility program to delete the data set. The IEHPROGM utility program is described in the IBM Utilities publication, GC28-6586.)

There are two different ways to specify a time period: (1) tell the system how many days you want the data set kept (the RETPD subparameter) or (2) tell the system the exact date after which the data set need no longer be kept (the EXPDT subparameter).

If the user codes the RETPD subparameter, he specifies a 1- to 4-digit number, which represents the number of days the data set is to be kept. If he codes the EXPDT subparameter, he specifies a 2-digit year number and a 3-digit day number (e.g., January 1 would be 001, July 1 would be 182), which represents the date after which the data set need no longer be kept. When neither the RETPD or EXPDT subparameter is specified for a new data set, the system assumes a retention period of zero days.

The RETPD or EXPDT subparameter must follow all other subparameters of the LABEL parameter. If no other subparameters are coded, the user can code LABEL=RETPD=nnnn or LABEL=EXPDT=yyddd.

5.6.6.8 Obtaining Space for Direct-Access Data Sets

The format of the SPACE parameter is as follows:

```
SPACE=( { TRK       }, (quantity  [,increment]  [,directory] )
        { CYL       }             [,          ]  [,index     ]
        { blocksize }

        [,RLSE,CONTIG]  [,ROUND]  )
        [,,MXIG      ]  [      ]
        [,,ALX       ]  [      ]
```

The SPACE parameter must be specified when creating new direct-access data sets.

Space can be requested in terms of cylinders, tracks, or blocks. For most efficient use space should be allocated in cylinders. Refer to subsection 17.2 for a further description of the SPACE parameter.

5.6.6.9 Channel Optimization

SEP=(ddnames)

AFF=ddname

A maximum of eight ddnames previously defined in the step may appear in the SEP parameter. The parentheses are unnecessary if only one ddname is coded. The AFF parameter must refer to a prior data definition statement which contains the SEP parameter. Refer to subsection 17.1.5 for a further discussion of channel optimization.

5.6.6.10 Allocating an I/O Unit

Before the data set can be used as input to a processing program or written as output by a processing program, the volume on which a data set resides or will reside must be mounted on an input/output device. The UNIT parameter provides the system with the information it needs to assign a device to the data set. The format of the UNIT parameter is as follows:

```
{UNIT=(  [unit address]      [,unit count]   [,DEFER] [,SEP=(ddname,...)] )}
{        [device type  ]     [,P         ]                                 }
{        [group name   ]     [,          ]                                 }
{                                                                          }
{UNIT=AFF=ddname                                                           }
```

5.6.6.10.1 Rules for Coding

a. If the only subparameter coded in the UNIT parameter is the first subparameter, the user need not enclose it in parentheses.

b. The user need not code the unit count subparameter if he wants only one device assigned to the data set.

c. The UNIT and DDNAME parameters are mutually exclusive parameters; therefore, if DDNAME is coded, do not code the UNIT parameter.

5.6.6.10.2 Identifying the Device

The user must identify to the system the specific device he wants or the type of device he wants. To identify a specific device, he must specify a unit address. When a unit address is coded, the system assigns the user that unit.

The user should not identify a device by its unit address unless it is absolutely necessary. Specifying a unit address limits unit assignment and may result in a delay of the job and other following jobs if the unit is being used by another job or in cancellation by the operator if the unit is not available.

5.6.6.10.3 Device Type

Device types correspond to particular set of features of input/output devices. When the user codes a device type, he allows the system to assign any available device of that device type. For example, if the device type he wants is a 2314 Disk Storage Drive, he codes UNIT=2314. The system assigns space on an available 2314. If only one device in the system is of that device type, the system assigns space on that device. If there is more than one device in the system of that device type, there is a certain degree of device independence.

The device types available on M&DO systems and their descriptions are listed below. 2400-7 and 2400-9 are device type names added to the IBM device type list by the GSFC system programmers at system generation time. (The user can code only those device types that were defined during system generation.)

TAPE

Device Type	S/95	S/75	S/65	Device
2400	X	X	X	2400 series 9-Track Magnetic Tape Drive that can be allocated to a data set written or to be written in 800 bpi when the dual-density feature is not installed on the drive or in 1600 bpi when the dual-density feature is installed on the drive.
2400-2 2400-7	X	X	X	2400 series Magnetic Tape Drive with 7-Track Compatibility and Data Conversion.
2400-3	X	X	X	2400 series 9-Track Magnetic Tape Drive that can be allocated to a data set written or to be written in 1600 bpi density.
2400-4 2400-9	X	X	X	2400 series 9-Track Magnetic Tape Drive having an 800 and 1600 bpi density capability.

DIRECT ACCESS

Device Type	S/95	S/75	S/65	Device
2301	X	-	-	2301 Drum Storage Unit
2303	-	X	X	2303 Drum Storage Unit
2314	X	X	X	2314 Storage Facility
2321	X	X	-	Any bin mounted on a 2321 data cell drive

The 2301 and 2303 Drum Storage Units are reserved for system data sets.

UNIT RECORD

Device Type	S/95	S/75	S/65	Device
1052	X	X	X	1052 Printer-Keyboard
1403	X	X	X	1403 Printer
2501	-	X	-	2501 Card Reader
2540	X	X	X	2540 Card Read Punch (read feed)
2540-2	X	X	X	2540 Card Read Punch (punch feed)

GRAPHIC

Device Type	S/95	S/75	S/65	Device
2250-1	X	X	-	2250 Display Unit, Model 1
2250-3	-	-	X	2250 Display Unit, Model 3
2260-1	X	-	X	2260 Model 1 Display Station (Local Attachment)

5.6.6.10.4 Group Name

A group name is one through eight alphameric characters and identifies a device or a group of devices. The group of devices can consist of devices of the same type or different direct access and tape device types. Group names are established during system generation.

When a user codes a group name, he allows the system to assign any available device type that is included in the group. (If a group consists of only one device type, as is true of all the GSFC group names, the system assigns that device.) For example, if all 2314 Disk Storage Units are included in the group named DISK and the user codes UNIT=DISK, the system assigns an available 2316 disk pack on a 2314 device.

Subsection 19.2 contains a list of the GSFC standard group names.

5.6.6.10.5 Unit Count

The unit count subparameter indicates how many devices the user wants assigned to a data set. If he does not code this subparameter, or codes 0, the system assigns one device. (If he receives a passed data set or refers the system to a cataloged data set or earlier DD statement for volume and unit information (VOLUME=REF=reference), the system assigns one device, even if more devices were requested in an earlier DD statement.) Only in one case may the system assign more than one device: when two DD statements in a step request use of the same volume. If either of these two DD statements requests any other volume(s), the system assigns an additional device.

For operating efficiency, the user can request multiple devices for a multivolume data set or for a data set that may require additional volumes. When each required volume is mounted on a separate device, time is not lost during execution of the job step while the operator demounts and mounts volumes. The maximum number of devices that can be requested per DD statement is 59.

In the following cases, the user should always code the unit count subparameter when the data set may be extended to a new volume:

- If the data set resides on a permanently resident or reserved volume. In these two cases, the volume cannot be demounted in order to mount another volume.

- If the data set is assigned space through suballocation. Code the unit count subparameter on the DD statement that requests the space to be suballocated.

The unit count subparameter is a positional subparameter, and it shares the same position as the subparameter P. If neither of these subparameters is coded and the DEFER or SEP subparameter follows, code a comma to indicate the absence of the unit count subparameter and the subparameter P.

5.6.6.10.6 Parallel Mounting

Requesting parallel mounting has the same effect as specifying a unit count, i.e., more than one device is assigned to the data set. When parallel mounting is requested, the system counts the number of volume serial numbers specified in the VOL=SER=parameter on the DD statement and assigns to the data set as many devices as there are serial numbers. (For cataloged data sets, the system counts the number of volume serial numbers contained in the catalog.) The user can request parallel mounting by coding the letter P in place of the unit count subparameter.

The subparameter P is a positional subparameter, and it shares the same position as the unit count subparameter. If neither of these subparameters is coded and the DEFER or SEP subparameter follows, code a comma to indicate the absence of the subparameter P and the unit count subparameter.

5.6.6.10.7 Deferred Mounting

The DEFER subparameter requests the system to assign the required units to a data set and to defer the mounting of the volume(s) on which the data set resides until the processing program attempts to open the data set. The DEFER subparameter should only be coded on DD statements that define data sets residing on removable volumes. The DEFER subparameter cannot be coded on a DD statement that defines an indexed sequential data set or that defines a new data set that is to be written on a direct-access volume, because space cannot be allocated to the data set.

If the user requests deferred mounting of a volume and the data set on that volume is never opened by the processing program, the volume is not mounted during the execution of the job step. If a later job step refers to that data set, the system may assign a different device to the data set than was originally assigned to it.

If the user requests deferred mounting of a private volume (a private disk pack or data cell or any specific tape) and has not filled in the serial number on his computer request form, the following sequence of events may take place: (1) the user's job will be read in and be executed; (2) the deferred data set will be opened; (3) the requested volume will not be available; (4) the operator will cancel the user's job; and (5) the user will be charged for all the time used.

5.6.6.10.8 UNIT Separation and Affinity

These topics are discussed in paragraph 17.1.5.2.

5.6.6.10.9 When Not to Code the UNIT Parameter

Except in a few cases, the UNIT parameter is always coded on a DD statement that defines a data set that requires one or more devices. In the following cases, the system obtains the required unit information from other sources. Therefore, the user need not code the UNIT parameter:

- When the data set is cataloged. For cataloged data sets, the system obtains unit and volume information from the catalog. However, if VOLUME=SER=serial number is coded on a DD statement that defines a cataloged data set, the system does not look in the catalog. In this case, the user must code the UNIT parameter. If the VOLUME parameter is not coded but the user requests a device in the UNIT parameter, the request is ignored.

- When the data set is passed from a previous job step. For passed data sets, the system obtains unit and volume information from an internal table. However, if VOLUME=SER=serial number is coded on a DD statement that defines a passed data set, the system does not look in the internal table. In this case, the user must code the UNIT parameter. If the VOLUME parameter is not coded but the user requests a device in the UNIT parameter, the request is ignored.

- When the data set is to use the same volumes assigned to an earlier data set, i.e., VOLUME=REF=reference is coded. In this case, the system obtains unit and volume information from the earlier DD statement that specified the volume serial number or from the catalog. If the user requests a device in the UNIT parameter, the request is ignored.

- When the data set is to share space or cylinders with an earlier data set, i.e., SUBALLOC or SPLIT is coded. In this case, the system obtains unit and volume information from the earlier DD statement that specifies the total amount of space required for all the data sets. If the VOLUME parameter is coded, it is ignored. If the user requests a device in the UNIT parameter, the request is ignored.

In all of these cases, the user can code the UNIT parameter when he wants more devices assigned.

5.6.6.11 Defining a Volume

A volume can be a tape reel, a disk pack, a data cell, or a drum. The VOLUME parameter provides information about the volume or volumes on which an input data set resides or on which an output data set will reside.

Before a data set can be read or written, the volume on which the data set resides or will reside must be mounted. For an existing data set, the user must identify the volume or volumes on which the data set resides by making a specific volume request. For a new data set, the user can make a specific volume request or let the system select a volume for him by making a nonspecific volume request. The VOLUME parameter is specified as follows:

```
{VOLUME}  =  ([PRIVATE]  [,RETAIN]  [,volume sequence number]  [,volume count]
{VOL   }     (           [,      ]  [,                     ]

      [,] [ SER=(serial number,...)                          ]
          [ REF=dsname                                       ]
          [ REF=*ddname                                      ]  )
          [ REF=*.stepname.ddname                            ]
          [ REF=*.stepname.procstepname.ddname               ]
```

5.6.6.11.1 Rules for Coding

a. The volume sequence number subparameter can be one to four digits.

b. The volume count subparameter is a number from one through 255.

c. If the only subparameter the user is coding is PRIVATE, he need not close it in parentheses.

d. If the only subparameter the user is coding is SER or REF, he codes VOLUME=SER=(serial number,...) or VOLUME=REF=reference.

e. If the list of volume serial numbers consists of only one serial number, the user need not enclose the serial number in parentheses.

f. The VOLUME, DDNAME, and SYSOUT parameters are mutually exclusive parameters; therefore, if DDNAME or SYSOUT is coded, do not code the VOLUME parameter.

g. The VOLUME parameter should not be used to retrieve a data set which is cataloged or passed.

5.6.6.11.2 Specific Volume Request

A specific volume request informs the system of the volume's serial number. Any of the following implies a specific volume request:

1. The data set is passed from an earlier step or is cataloged.
2. VOLUME=SER=serial number is coded on the DD statement.
3. VOLUME=REF=reference is coded on the DD statement, referring to an earlier specific volume request.

When the user makes a specific volume request, he can code the PRIVATE subparameter or the PRIVATE and RETAIN subparameters in the VOLUME parameter. For passed data sets, he can also code the volume count subparameter. For cataloged data sets, he can also code the sequence number and volume count subparameters.

5.6.6.11.3 Nonspecific Volume Request

A nonspecific volume request can be made only if the user is defining a new data set. When he makes a nonspecific volume request, the system may assign his data set to a volume that is already mounted, or may cause a volume to be mounted. What the system does depends on the volume state of the volumes that are already mounted. The volume states that mounted volumes can assume and how they affect volume selection are described under "Volume States" in Section 17.

When you make a nonspecific volume request, you can code the PRIVATE subparameter, or the PRIVATE and RETAIN subparameters, and the volume count subparameter in the VOLUME parameter.

5.6.7 DEFINING DATA IN THE INPUT STREAM (DD * or DD DATA)

The input stream can be on a card reader, a magnetic tape, or a direct-access device.

Data in the input stream are written onto a direct-access device to allow for high-speed retrieval when the data are required. The reader procedure assigns two buffers to the data control block plus a blocking factor (3200 bytes per block) to be used to block the data in the input stream when they are placed on the direct-access device. The user can assign a smaller blocking factor by including the DCB subparameter BLKSIZE on the DD * or DD DATA statement, e.g., DCB=BLKSIZE=80. He can also assign the number of buffers by including the DCB subparameter BUFNO, e.g., DCB=(BLKSIZE=3200,BUFNO=2).

If the processing program does not read all the data in an input stream, the remaining data are flushed without causing abnormal termination of the job.

5.6.7.1 Rules for Coding

a. In MVT, there may be more than one DD * and/or DD DATA statement per job step.

b. In MVT, when the user calls a cataloged procedure, he may add more than one DD * and/or DD DATA statement to a procedure step.

c. In MVT, if the data are preceded by a DD * statement, a delimiter statement (/*) following the data is optional.

d. If the data are preceded by a DD DATA statement, a delimiter statement (/*) following the data is required.

e. The data cannot contain the characters /* in columns 1 and 2. PL/I comments begin with /*, but most PL/I coders begin in column 2, which is the default starting position.

f. In MVT, the DCB subparameters BLKSIZE and BUFNO have meaning when coded on a DD * or a DD DATA statement. Any other parameters coded on a DD * or DD DATA statement are not used but are checked for syntax.

g. A cataloged procedure cannot contain either a DD * or a DD DATA statement.

h. Code the DATA parameter instead of the * parameter when the data contains job control statements.

The user can include several distinct groups of data in the input stream for a job step or procedure step. The system can recognize each group of data if the user precedes each group with a DD * or DD DATA statement, or follows each group with a delimiter statement (/*), or both. (If he leaves out the DD DATA or DD * statement for a group of data, the system provides a DD * statement having SYSIN as its ddname.)

The following rules apply when data are entered through an input stream:

- The input stream can be on any device supported by QSAM.
- The characters in the records must be coded in BCD or EBCDIC.

5.6.7.2 The DCB Subparameters BLKSIZE and BUFNO

BLKSIZE and BUFNO may be coded on a DD statement that contains the DDNAME parameter, which refers to another DD statement. If, in turn, the referenced DD statement defines data in the input stream, these DCB subparameters are used to block the data. However, if the referenced DD statement contains its own DCB subparameters BLKSIZE and BUFNO, these values override those on the DD statement that contains the DDNAME parameter.

5.6.8 BYPASSING I/O OPERATIONS ON THE DATA SET (DUMMY)

The DUMMY parameter, a DD statement positional parameter, allows the user to bypass input/output operations, device and space allocation, and disposition of data sets referred to by the basic sequential or queued sequential access method. This facility can be used to suppress the writing of certain output data sets, such as assembler listings, and to update new master files with a dummy detail file. Bypassing operations on noncritical data sets also results in a saving of time when a program is being tested. To use this facility, DUMMY is coded as the first parameter in the operand field.

DUMMY specifies that no devices or external storage space is to be allocated to the data set, no disposition processing is to be performed on the data set, and, for BSAM and QSAM, specifies that no input or output operations are to be performed on the data set.

5.6.8.1 Rules for Coding

1. The user can code the DUMMY parameter by itself or follow it with all the parameters necessary to define a data set.

2. If the DUMMY parameter is coded and an access method other than the basic sequential access method (BSAM) or queued sequential access method (QSAM) is requested to read or write the data set, a programming error occurs.

5.6.8.2 The Function of the Dummy Parameter

When the user uses either the basic sequential or queued sequential access method, the DUMMY parameter allows his processing program to execute without performing input or output operations on a data set. When the processing program asks to write a dummy data set, the write request is recognized, but no data are transmitted. When the processing program asks to read a dummy data set, an end-of-data-set exit is taken immediately.

Besides bypassing input or output operations on a data set, the DUMMY parameter causes the UNIT, VOLUME, SPACE, and DISP parameters, when coded on the DD DUMMY statement, to be ignored (if coded, these parameters are checked for syntax). Therefore, no devices or external storage space is allocated to the data set and no disposition processing is performed on the data set.

If the user knows that certain parts of a program "work" and need not be processed each time the job is submitted for testing, the DUMMY parameter can help save time. The DUMMY parameter can also be used to suppress the writing of data sets, such as output listings, that the user does not need.

5.6.9 DEFINING THE SYSTEM OUTPUT STREAM

```
SYSOUT=(classname [,program name  [,form number])]
                  [,
```

5.6.9.1 Rules for Coding

a. The classname can be any alphameric character (A-Z, 0-9). See paragraph 5.6.9.4.

b. The form number is one to four alphameric and national (@,$,#) characters.

c. If a program name and form number are omitted, the user need not enclose the classname in parentheses.

d. The UNIT, SPACE, OUTLIM (Release 19 and later only), and DCB parameters can be coded with the SYSOUT parameter. Besides the mutually exclusive parameters listed below, other parameters codes with the SYSOUT parameter are ignored.

e. The DISP, DDNAME, AFF, SEP, VOLUME, LABEL, SPLIT, and SUBALLOC parameters and the SYSOUT parameter are mutually exclusive parameters; therefore, if any of these parameters are coded, do not code the SYSOUT parameter. To override a SYSOUT parameter in a cataloged procedure, code the DISP parameter.

5.6.9.2 Advantages or Coding the SYSOUT Parameter

When a user wants a data set printed on an output listing or in the form of punched cards, he can code the UNIT parameter and identify the unit record device he wants, or code the SYSOUT parameter and specify the class that corresponds to the type of unit record device he wants. There are advantages to coding the SYSOUT parameter:

a. During execution, the output data set is written to a direct-access device, and a system output writer writes the data set to a unit record device at a later time. This allows greater flexibility in scheduling print and punch operations, and improves operating system efficiency. The user can also write his output data set directly to a unit record or magnetic tape device.

b. The output data set and system messages resulting from the job can be assigned to the same type of unit record device. This is accomplished by specifying the same classname in the SYSOUT and MSGCLASS parameters. (The MSGCLASS parameter is coded on the JOB statement.)

c. When a user wants the output data set printed or punched on a special output form, he can specify the form number in the SYSOUT parameter and let the system inform the operator at the time the data set is to be written what form is to be used. The use of this parameter is not encouraged. When multiple copies of a printout are desired they may be obtained by submitting the original to the 360/95 dispatcher to be duplicated on the xerox 2400. The xerox 2400 will reduce 11 x 14 7/8 inch pages to 8 1/2 x 11 inch pages. It can sort and collate up to 29 sets in one pass of the original listing.

d. The reader-interpreter procedure provides a default space allocation. This may be overridden. If the cataloged procedure or program do not provide DCB information, it must be provided in the DD card. A SPACE allocation of (CYL,1) is sometimes used on SYSUDUMP or SYSABEND DD cards to print the first pages of a dump.

5.6.9.3 Coding Other Parameters with the SYSOUT Parameter

The UNIT, SPACE, OUTLIM and DCB parameters can be coded with the SYSOUT parameter. The DDNAME, DISP, AFF, SEP, VOLUME, LABEL, SPLIT, and SUBALLOC parameters are mutually exclusive with the SYSOUT parameter; any other parameters that the user codes with the SYSOUT parameter are ignored.

The user can write output data sets destined for unit records devices to a direct-access device instead of immediately writing the data set to the desired unit record device. Later, a system output writer writes the data set to the desired unit record device. In the UNIT parameter, he can request what type of direct-access device he wants for writing the output data set, how many devices he wants (up to a maximum of five), and unit separation from other data sets defined in the job step. In the SPACE parameter, he can specify how much space should be allocated to the data set and that unused space is to be released. If he omits the UNIT parameter, the system assigns a device; if he omits the SPACE parameter, the system assigns the amount of space to be allocated. These values are part of the PARM parameter field in the input reader procedure used to read the input stream.

The user can also write an output data set directly to the desired unit record or magnetic tape device. When direct system output is desired, the operator selects a unit record or magnetic tape device for a class by issuing a START DSO (Direct System Output) command. In addition to the SYSOUT parameter, the DCB and UCS parameters can be coded. If the SYSOUT subparameters other than classname are coded, the specified information is ignored. The UNIT and SPACE parameters are also ignored if direct system output processing is used. Since the type of processing to be used may not always be known, it is advisable to code these parameters in case an intermediate direct-access device is used.

The DCB parameter can be coded with the SYSOUT parameter to complete the data control block associated with the output data set. The information contained in this data control block is used when the data set is written to the direct-access device and read by the system output writer. However, the output writer's own DCB attributes are used when the data set is written to the desired unit record device.

The OUTLIM parameter allows the user to specify a limit for the number of logical records he wants included in the output data set being routed through the output stream. The OUTLIM parameter has meaning only in systems with the System Management Facilities (FMS) option with system, job, and step data collection. Unless the SYSOUT parameter is coded in the operand field of the same DD statement, the OUTLIM parameter is ignored.

5.6.9.4 The Classname

When the user codes the SYSOUT parameter, he indicates a classname. A classname is an alphameric character (A-Z, 0-9) that indicates the output class desired. Each installation specifies what classnames correspond to what output classes. Therefore, when the user specifies a classname, the operator knows what type of output device the user wants, and he ensures that a system output writer is available to write the output data set to the desired output device.

The system determines where system messages resulting from a job are to be written based on what is coded in the MSGCLASS parameter on the JOB statement. If the MSGCLASS parameter is not coded, system messages associated with the user's job are routed to the default output class specified in the PARM field of the input reader procedure. The default for the MSGCLASS parameter is A unless changed by the user's installation. Class A corresponds to a printer. If the user wants his output data set and the system messages resulting from the job written to the same unit record device, he simply codes the same classname in both the MSGCLASS and SYSOUT parameters, or omits the MSGCLASS parameter and codes his installation's default output class in the SYSOUT parameter.

a. SYSOUT=A - This is the regular printout from the on-line printer. It is on regular paper, 6 lines per inch.

b. SYSOUT=B - This is the card punch, for punched decks.

c. SYSOUT=C - This is the same as class A output, but printed with a lower priority.

d. SYSOUT=R - This is used only for RITS.

e. SYSOUT=Z - Any jobs left in the system after RJE is shut down, and which have specified SYSOUT=Z, will be printed on the system printer in the computer center.

5.6.9.5 Job Separators

The user's output data are preceded by a job separator - a series of three listing pages or three punched cards that separate the output data sets of different jobs. The output data sets from these jobs were written to the same unit. Each page or card contains the name of the job whose data follows, and identifies the output class and box number. Job separators make it easier for the operator to separate the data produced by the user's job from the data of other jobs.

On the 360/95, jobs submitted via RITS or RJE will have an X printed as the last character of the box number, making those jobs readily identifiable to the operator.

5.7 DELIMITER AND NULL CONTROL CARDS

5.7.1 DELIMITER STATEMENT

The DELIMITER statement marks the end of a data set in the input stream and is used to separate data in the input stream from the job control statements that follow the data. The DELIMITER is coded with the characters /* in columns 1 and 2, with the other columns blank.

It should be noted that the /* card is not needed, except after data introduced by a DD DATA statement. Following data introduced by a DD * statement, the /* card may be used or omitted at the user's discretion. In the JCL examples shown in this User's Guide, note that the /* card sometimes is used for clarity, but is otherwise omitted. Refer to paragraph 5.6.7 for a discussion of DD * and DD DATA.

5.7.2 NULL STATEMENT

The NULL statement is used to mark the end of a job's control statements and data. A NULL statement causes the scheduler to look for the next JOB statement. If there are any cards between the NULL statement and the next JOB statement, these cards are flushed from the input stream. The NULL statement is coded with the identifying characters //, in columns 1 and 2, with all other columns blank.

SECTION 6

STANDARD (IBM-SUPPLIED) PROCESSORS

6.1 GENERAL

Computer manufacturers customarily furnish software to facilitate the use of their computers. This software is comprised of an operating system, processors used in program preparation, and utility programs for performing certain standard functions. This section describes the IBM-supplied processors which are available on the M&DO computers, including the language processors -- FORTRAN, PL/I, RPG, and Assembler (F); and the large utilities -- Linkage Editor, Loader, and Sort/Merge. Note that COBOL is not supported on M&DO computers.

The language processors translate symbolic statements into machine instructions, producing object modules. The object modules must undergo two additional steps - linkage editing and loading - before they become executable programs. This is done so that a number of separately compiled programs, including library subroutines, may be combined into a single load module. The linkage editor combines and edits modules to produce a single load module that can be brought into main storage for execution by program fetch. The linkage editor provides several processing facilities that are performed either automatically or in response to control statements prepared by the programmer.

The loader combines the basic editing and loading functions of the linkage editor and program fetch in one job step. It is designed for high performance loading of modules that do not require the special processing facilities of the linkage editor and fetch, such as overlay. The loader does not produce load modules for program libraries.

Some of the processors are available in several "design levels" (e.g., FORTRAN G and FORTRAN H). Originally, the letter designating the design level corresponded to the amount of memory required in the host computer in order to execute the processor. For example, a computer with G level memory (128k) or larger would be needed to use FORTRAN G. However, because of the MVT environment and variations in the choice of operating system configurations on a given computer, only the size of the region in which the compiler operates is of concern. Further, some of the processors, notably the Linkage Editor, are subdivided into several design levels within these stages. In general, the higher design levels require increasingly more memory and sometimes offer additional capabilities.

6.2 LANGUAGE PROCESSORS

The most widely used language at GSFC is FORTRAN IV for use in mathematical and scientific applications. Paragraph 6.2.1 briefly describes the evolutionary development of FORTRAN IV and points out the major differences between the FORTRAN IV language as supported by the FORTRAN G and H compilers, and the FORTRAN compilers on the IBM 7094 and Univac 1108. ANSI FORTRAN IV specifications are also given. Differences between the FORTRAN G and H compilers are presented in paragraph 6.2.1.1. PL/I is a more recently developed language than FORTRAN and is more comprehensive, suitable for commercial applications as well as scientific applications. RPG is a language designed for report generation.

6.2.1 FORTRAN IV

The FORTRAN language has undergone an evolutionary development, beginning with the original FORTRAN compiler for the IBM 704 in 1956. This processor was modified in 1958 to accept an augmented language known as FORTRAN II and, subsequently, processors for FORTRAN II became available on a variety of computers. The language was further extended in 1962 with the advent of FORTRAN IV. While FORTRAN IV is a more general language, it is defined so that it is in some ways incompatible with FORTRAN II. To facilitate the transition, some FORTRAN IV processors were designed to accept certain FORTRAN II features which had been eliminated from FORTRAN IV. Among these are the PRINT and PUNCH statements, and the READ statement with an implied unit number. As FORTRAN processors proliferated, a number of language differences arose, since there was no existing body charged with controlling the development of the language. Finally, in 1966, the American Standards Association (ASA), currently known as the American National Standards Institute (ANSI), established a standard definition of FORTRAN based upon FORTRAN IV. This standard did not attempt to extend the language, but to define those features which were in current use and which were considered valuable. It did not contain any of the extended features supported by some processors which tended to depend upon the capabilities of a particular computer or operating system. Extension of this type, therefore, continues to appear in later processors, while ANSI FORTRAN was used as the base.

The IBM S/360 FORTRAN IV language embraces the complete ANSI FORTRAN and includes some additional features which give the user greater control over S/360 facilities and which offer increased generality. Among the most significant extensions are direct-access input/output, the IMPLICIT statement, mixed-mode expressions, and the length specifications on arithmetic variables. The table in paragraph 6.2.1.1 presents the language features supported by the IBM FORTRAN IV (G and H) compilers which are not part of ANSI FORTRAN. This table also provides a comparison of ANSI FORTRAN with 7094 FORTRAN IV and 1108 FORTRAN V.

Additional information on the use of these features may be obtained from the IBM FORTRAN IV Language, Form GC28-6515, and IBM System/360 Operating System FORTRAN IV (G and H) Programmer's Guide, Form GC28-6817. For detailed information on ANSI FORTRAN, the reader is referred to ASA FORTRAN (ANSI, X3.9 - 1966).

6.2.1.1 Major Language Differences

The major differences between IBM S/360 FORTRAN IV (G and H compilers) and the FORTRAN supported by the IBM 7094 and Univac 1108 computers are presented here.

In relating S/360 FORTRAN to that of the 7094 and 1108, it is important to note that the IBM 7094 FORTRAN IV compiler predates ANSI FORTRAN, and thereby was a direct influence on development of ANSI FORTRAN. This compiler supports some features which are a carry over from FORTRAN II, such as the PRINT and PUNCH statements, and the READ statement with implicit unit designation. These were eliminated from ANSI FORTRAN in the interest of generality, but were carried into the S/360 FORTRAN IV to facilitate conversion of 7094 programs to the S/360. Other than these considerations, the 7094 FORTRAN IV is essentially the same as ANSI FORTRAN.

The Univac 1108 FORTRAN V compiler is a more recent development than ANSI FORTRAN and extends upon it, as does the S/360 FORTRAN IV. The 1108 extensions, however, differ considerably from those of the S/360. While it is not within the scope of this document to describe the 1108 FORTRAN V language, those features of S/360 FORTRAN IV which are supported by the 1108 FORTRAN V will be noted.

Any discussion which compares a programming language as supported by different computers must also deal with certain considerations imposed by the hardware. Word size, for instance, affects the precision of numeric values. It also influences the way in which character data is manipulated by the program. On the 7094 and 1108, for instance, six characters can be contained in a word, while on the S/360, only four characters are possible. The bit configurations for characters may also vary from one machine to another.

Table 6.2-1 presents a comparison of S/360 FORTRAN IV, ANSI FORTRAN, 7094 FORTRAN IV, and 1108 FORTRAN V. Along with linguistic differences, differences imposed by hardware are also shown. This table is not intended to be all inclusive; it uses the S/360 FORTRAN as a base, i.e., those features of 1108 FORTRAN which are not part of S/360 FORTRAN are not represented in the table.

The above discussion relates to considerations which affect the way that a FORTRAN program is written. There are many other considerations, such as the use of the compiler and interfaces with the operating system, which are completely dependent on a specific implementation and the host environment. There are differences, for instance, in the degree and types of optimization, debugging facilities, and limitations of each compiler. For details of these considerations as related to the S/360 G and H compilers, refer to the IBM System/360 Operating System FORTRAN IV (G and H) Programmers Guide, Form GC28-6817.

Table 6.2-1. Comparison of S/360 FORTRAN IV, ANSI FORTRAN, 7094 FORTRAN IV, and 1108 FORTRAN V

S/360 FORTRAN Features	ANSI	7094	1108	Notes
Direct-access I/O statements				Allows the user to specify the location (relative record number) within a data set of the record to be accessed.
DEFINE FILE	No	No	No	Specifies the data set characteristics (such as record size, number of records)
FIND	No	No	No	Provides an overlap of record accessing and processing.
READ, WRITE with a'r parameter	No	No	No	Specifies the data set reference number (unit number) and the relative position of the record within a data set.
END, ERR parameters in READ statement	No	No	Yes	Specifies the recovery points in case of an error condition or end of data.
NAMELIST statement	No	Yes	Yes	Allows the names of variables to be input/output to be specified separately from the READ/WRITE statement.
PRINT, PUNCH statements	No	Yes	Yes	Provides compatibility with other FORTRAN IV compilers (which predate the ANSI standard).
READ with implicit unit number	No	Yes	Yes	

Table 6.2-1. (Cont'd)

S/360 FORTRAN Features	ANSI	7094	1108	Notes
T Format Code	No	No	Yes	Specifies the actual character position within a record (as opposed to the relative position, as with X format).
Z Format Code	No	No	No	Specifies hexadecimal data--replaces O format for S/360.
ENTRY statement	No	Yes	Yes	Provides for multiple entry points into a subprogram.
Nonstandard returns from subroutines (statement label parameters in CALL statement, RETURN:)	No	Yes	Yes	There are syntax differences in statement label parameters between various implementations. Check the appropriate programming manual for specifics.
Length specification of variables as part of type specification	No	No	No	Gives the user greater control over the amount of storage occupied by variables, and over the resulting precision.
IMPLICIT statement	No	No	Yes	Gives the user the means to establish his own default type attributes for variables not explicitly declared.
Initial data values in type specification	No	No	Yes	Accomplishes the same result as the DATA statement, but is more convenient to use.

Table 6.2-1. (Cont'd)

S/360 FORTRAN Features	ANSI	7094	1108	Notes
Hexadecimal constant	No	No	No	May be used as data initialization value.
Literal enclosed in apostrophes	No	No	No	Eliminates the necessity of counting characters, as with hollerith constants.
PAUSE 'message'	No	No	No	Allows for an alphanumeric message to be sent to the operator (1108 FORTRAN allows for six alphanumeric characters, without the apostrophes).
Mixed mode expressions	No	No	Yes	The type of result depends on the combination of operands.
Generalized subscripts	No	No	Yes	The result is converted to integer, if necessary.
Maximum number of dimensions in an array = 7	3	7	7	
Adjustable dimensions	Yes	Yes	Yes	For the 7094, the values of the arguments that represent the array dimensions must agree with the dimensions of the actual array. For other versions, they may be less than the actual dimensions.

Table 6.2-1. (Cont'd)

S/360 FORTRAN Features	ANSI	7094	1108	Notes
Integer, maximum magnitude = $2^{31}-1$	--	$2^{35}-1$	$2^{35}-1$	These are hardware considerations.
Integer, maximum decimal digits = 10	--	11	11	
Real, maximum magnitude = 10^{75}	--	10^{38}	10^{38}	
Real, maximum decimal digits = 16	--	17	17	

6.2.1.2 FORTRAN IV Compilers

The FORTRAN IV compilers accept programs written in the FORTRAN IV language (as defined in the IBM System/360: FORTRAN IV publication), as input, and produce, as output, machine language object modules which in turn may be used as input to the Loader or Linkage Editor for execution as problem programs. Two different FORTRAN compilers are available on the M&DO computers -- FORTRAN level G, and FORTRAN level H. The major difference between these two levels is in the internal compiler processing, which results in differences in generated object code and differences in the compiler printed output. However, both compilers operate on an identical syntactic set of FORTRAN source statements; therefore, a FORTRAN source program may be used as input to either compiler. In general, the level H compiler has an extended range of options which provide the user with increased flexibility in specifying compiler operations and compiler output.

The name of the FORTRAN G compiler is IEYFORT; the name of the FORTRAN H compiler is IEKAA00. Most of the following discussion refers to the GSFC FORTRAN procedures as available on the M&DO computers. Users desiring to write their own compile procedures, rather than use the ones available in the system procedure library, must use an execute statement with a compiler name (e.g., // EXEC PGM=IEYFORT or PGM=IEKAA00) and must supply the necessary DD statements for the compiler.

6.2.1.2.1 GSFC FORTRAN Procedures and Compiler Data Sets

The FORTRAN compiler is normally invoked by executing the appropriate GSFC procedure. The G compiler is invoked by an execute statement of the form:

```
// EXEC FORTRANG,PARM=..... [1]
```

and the H compiler is invoked by the statement:

```
// EXEC FORTRANH,PARM=..... [1]
```

Several standard data sets are used by the compiler during its processing. Each data set has a specific functional use and specific device requirements. Standard assumptions are made for the DCB parameter of the data sets used by the compilers. Table 6.2-2 contains data set definition DCB parameters for the G compiler. Table 6.2-3 contains the DCB values for the H compiler data sets. Of the DCB values in these two tables, only the values for block-size can be overridden with a DD statement. The user may also specify the number of buffers to be used for compiler data sets. If the buffer number is not supplied, the QSAM default is used. The buffer defaults are three buffers for the card read punch (IBM 2540) and two buffers for all other devices. The compilers' use of these data sets is mentioned in conjuction with the discussion of compiler options which follows.

[1] See paragraph 6.2.1.2.2

Table 6.2-2. FORTRAN G Data Sets - DCB Parameters

DDNAME	RECFM	LRECL	BLKSIZE
SYSLIN	FB	80	3200
SYSPRINT	FBA	120	7200
SYSPUNCH	FB	80	7280
SYSIN[2]	FB	80	3200

Table 6.2-3. FORTRAN H Data Sets - DCB Parameters

DDNAME	RECFM	LRECL	BLKSIZE
SYSLIN	FB	80	3200
SYSPRINT	VBA	137	7265
SYSPUNCH	FB	80	7280
SYSUT1	FB	105	3465
SYSUT2	FB	1024 4096[1]	4096
SYSIN[2]	FB	80	3200

1
The value is within this range. The actual value is calculated during execution.

2
The SYSIN data set is not defined in the GSFC procedure. If a SYSIN DD card is not added by the user, the following card - //SYSIN DD * is automatically generated by the system.

From the tables, it can be seen that the H compiler uses two more data sets than does the G compiler. These two data sets are used for temporary work space to process the additional options available with the H compiler.

6.2.1.2.2 Compiler Options

Compiler options are a series of keywords that direct the processing in terms of the type of output which the compiler generates. The compiler options for the G and H compilers are identical with the exception that the H compiler has three additional options not available in the G compiler.

The compiler options are set during system generation. The option values set at SYSGEN are called default options. A list of these default options is presented in Table 6.2-4. The user may request options other than the standard defaults by explicitly coding the desired options in the PARM field of the EXEC statement. The available options are explained in detail in the FORTRAN G & H Programmer's Guide. Figure 6.2-1 describes the relationship between the specified options and compiler data sets.

All compiler listings of source code, object code, storage maps, error messages, structured source listing, and cross references are output to SYSPRINT. Object modules generated by the compiler are output to SYSLIN, which can be used as input to the Loader or Linkage Editor. Requested punched object decks are output to SYSPUNCH. The H compiler uses its two additional data sets (SYSUT1 and SYSUT2) for temporary work space while creating the structured source listing (available only in conjunction with OPT=2) and the cross reference listing.

6.2.1.2.3 Multiple Compilations

Both the G and H compilers are designed to facilitate multiple compilations; the compiler control program design is such that reloading of the compiler is unnecessary to accomplish compilation of multiple source modules. Therefore, SYSIN input of source statements to the compiler may contain several FORTRAN source modules. In compiling multiple FORTRAN source modules, the NAME option can be used to specify the NAME assigned to the main FORTRAN program. If the NAME option is not specified, the compiler assumes the name MAIN for the main program. The name of a FORTRAN subprogram is the name specified in the SUBROUTINE or FUNCTION statement.

6.2.1.2.4 FORTRAN G Compiler

The G compiler consists of a control program and five processing phases. The compiler operates in a minimum of 80k bytes of main storage. This includes space for compiler code, data access routines, and work space for compiler tables. The region size available for the compiler is directly related to

Table 6.2-4. Default Options for FORTRAN G and FORTRAN H Compilers

H Compiler Default Options		G Compiler Default Options	
S/360 Model 95	S/360 Model 75	S/360 Model 95	S/360 Model 75
SOURCE	SOURCE	SOURCE	SOURCE
NOLIST	NOLIST	NOLIST	NOLIST
EBCDIC	EBCDIC	EBCDIC	EBCDIC
NODECK	NODECK	NODECK	NODECK
LOAD	LOAD	LOAD	LOAD
MAP[1]	MAP	MAP[1]	MAP
LINECNT=58	LINECNT=58	LINECNT=58	LINECNT=58
NAME=MAIN	NAME=MAIN	NAME=MAIN	NAME=MAIN
ID[1]	ID	ID[1]	ID
OPT=0	OPT=0		
XREF[1]	NOXREF		
NOEDIT	NOEDIT		
SIZE=250K	SIZE =250K		

[1] These options differ from those defaults in the IBM FORTRAN procedures.

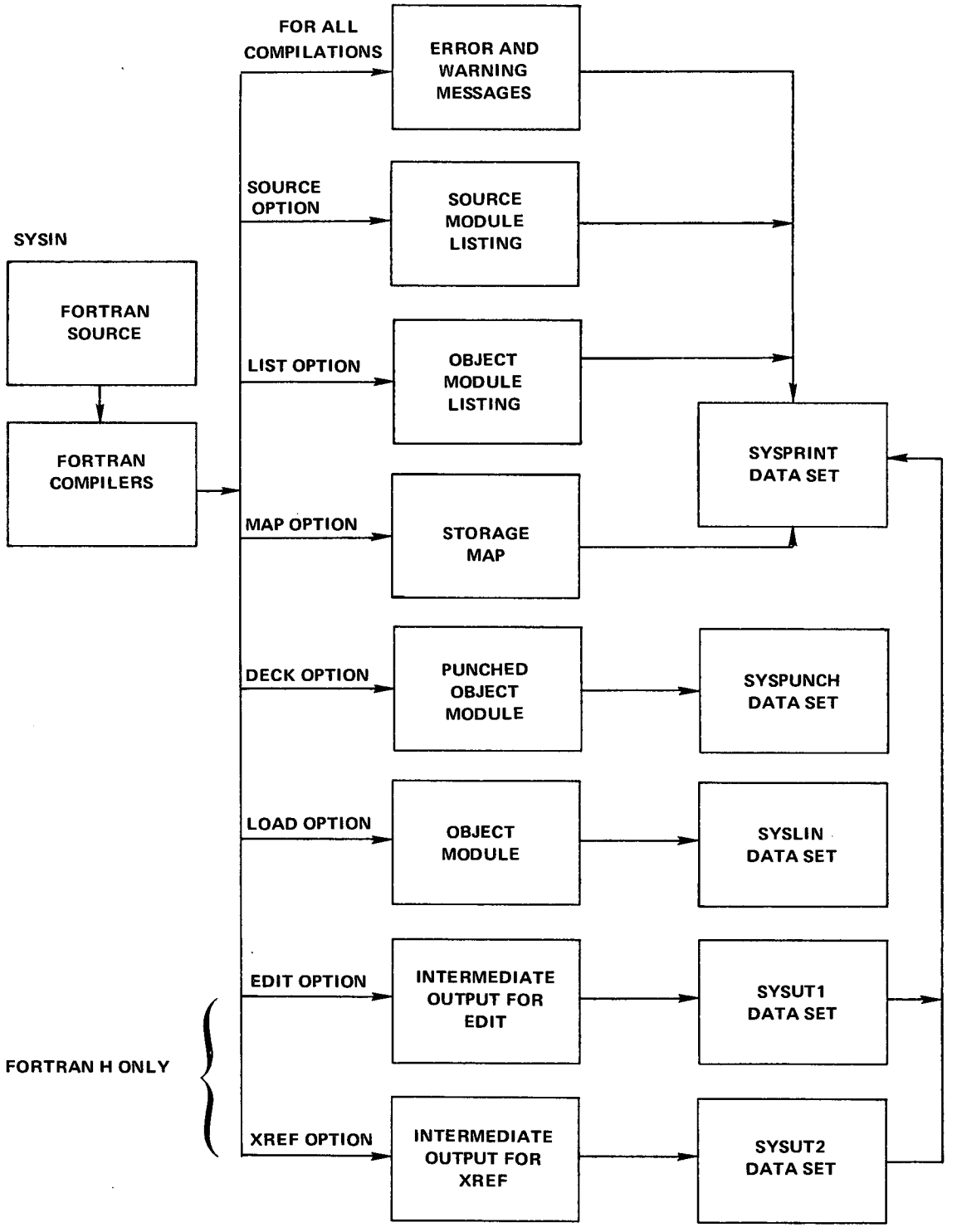


Figure 6.2-1. Usage of Compiler Data Sets

the maximum number of source statements which can be compiled. Therefore, if compilation is deleted because the main program or a subroutine is too large, an increase of the region specified in the EXEC statement will normally permit successful compilation. A region size of 100k is estimated to be adequate to compile 400 source statements. To adjust the region size to accommodate smaller or larger programs, IBM suggests allowing 75 bytes per source statement. A region size of 200k is specified in GSFC FORTRAN G procedure. A description of the G compiler storage map and G compiler optimization follows:

a. G Compiler Storage Map -- The FORTRAN G storage map, produced if the MAP option is specified, consists of a list of variables classified by their type (such as scalar variables and array variables) followed by a relative address. The map which is produced is not as useful as the H compiler storage map (which contains additional information useful in debugging). The options for structured source listing and cross reference listing are not available with the G compiler.

b. G Compiler Optimization -- The G compiler performs less extensive optimization than the H compiler. The optimization procedure operates over DO loops, and an attempt is made to optimize subscripting operations within the loop. During the optimization phase, decisions are made on the basis of frequency of use, as to which subscript expressions are to be kept in general registers and which are to be maintained in storage. The result of this optimization is that only the portion of each subscript which depends on the DO loop variable is computed on each pass through the loop.

6.2.1.2.5 FORTRAN H Compiler

The level H compiler consists of a control program and five processing phases. The compiler is an overlay structure that operates in a minimum of 89k bytes of main storage. A region size of 300k bytes is specified in the GSFC FORTRANH procedure. There is no known method of determining the number of FORTRAN source statements which can be compiled under the H level. Under release 19, the programmer may request the amount of storage used by the compiler by specifying the SIZE parameter on the EXEC card. Detailed information on use of this SIZE parameter is contained in the FORTRAN G & H Programmer's Guide (release 19). NOTE: Some discrepancies have been noted in the use of the SIZE parameter. Before making use of this parameter please contact the PAC in building 3, room 133A, extension 6768.

The H compiler storage map and H compiler optimization are described below:

a. H Compiler Storage Map -- The H compiler storage map which is generated by specifying the MAP option, lists the variable name, its type and length, and relative storage locations. In addition,

the variable names are followed by single-letter codes which describe how the variable is used within the program. These codes are often useful in debugging a program because they indicate whether the variable was used on the left and/or right of an expression, whether the variable is in common or passed as a subroutine argument, etc.

In addition, if the XREF option is specified, a cross reference listing is produced by the compiler. This listing consists of names of variables followed by a list of internal statement numbers in which the variables are used.

If the programmer is using the second level of optimization (OPT=2), he may also request a structured source listing by coding the EDIT option. This listing indicates the loop structure of the program. Each loop is assigned a three-digit number, and entrance and exits from the loop are marked.

b. H Compiler Optimization -- Three levels of optimization are available in the H compiler. The desired optimization level should be coded in the parameter field of the EXEC statement:

1. OPT=0 -- The OPT=0 level causes no optimization of the object code produced. Only a basic register assignment is made; that is, the S/360 is treated as if it only had three available registers -- a single branch register, a base register, and an accumulator. Therefore, the code produced is less efficient in terms of execution speed, and it is recommended that this level only be used for testing purposes, since the code produced generally runs slower than the G compiler.

2. OPT=1 -- The first level of optimization takes advantage of all available S/360 registers and performs full register assignment. The entire program is treated as a loop and subdivided into text blocks. Frequently used variables and constants are maintained in registers to eliminate excessive register loading and value storing. In addition, branching optimization is performed by the generation of RX instructions, when possible. This eliminates a register load and reduces the number of necessary address constants.

3. OPT=2 -- The second level of optimization uses the optimization techniques of OPT=1, with some additional optimization methods. Like OPT=1, full register assignment and branching optimization is used; however, this is performed on a loop-by-loop basis and is therefore more effective. The compiler analyzes the loop structure of the program.

In addition, the H compiler recognizes and replaces redundant computations. Common expression elimination is performed (i.e., unnecessary recomputation of identical expressions is eliminated). When logically possible, the compiler moves computations which need not be calculated within a loop outside the range of the loop. The compilation time using OPT=1 and OPT=2 is longer than OPT=0 or G compile time; however, the object code produced is more concise and efficient. Thus, execution time of programs compiled with optimization is shorter. Comparisons of large programs using OPT=2 generally indicate run times that are a minimum of two times faster than are available with FORTRAN G or OPT=0. Instances of a factor of 14 to 15 times greater speed have been noted with the use of OPT=2.

There are some cases in which the object code generated by OPT=2 is in error. However, the number of these cases is small. Because of the great saving of execution time, it is recommended that programmers use the optimization feature, but with a careful examination of output results. One approach is to check test case results by comparing a run made with OPT=0 and OPT=2 to determine if discrepancies exist.

The debug package available with FORTRAN G may not be used with FORTRAN H. However, the extended error handling feature is provided. This allows the user to monitor certain error codes, and to take appropriate action.

6.2.1.2.6 FORTRAN Programming Considerations

The following programming considerations should be noted:

a. Boundary Alignment -- The programmer must insure that all variables defined in the FORTRAN COMMON and EQUIVALENCE statements have proper boundary alignment. Full words must begin on full word boundaries. Boundary alignment is a system generation option which is not available for FORTRAN on M&DO computers. Corrective boundary alignment will not be made. Boundary violations will result in specification errors (completion code 0C6) when the variables are referenced in the program.

b. Sequential Data Sets -- FORTRAN sequential data sets which are read or written without format control must have a record format of variable span or variable block span specified (VS or VBS). Such records may not be described as fixed length even if the actual records are fixed length. If the RECFM parameter is not explicitly coded in the DCB, a default RECFM of VS is used. If some other RECFM is used to describe data sets without format control, ABEND occurs with a completion code of 0C0 or 0C5.

c. Equivalence Statements -- Array names appearing in an equivalence statement must have a subscript explicitly coded. If a subscript is not coded, a compiler diagnostic results. This is a language violation which was not previously detected.

d. Block Data -- Each labeled common area used in a block data subprogram must be dimensioned to the actual size of that common block. If this is not done, the Linkage Editor generates an error code (IEW0552). The Loader does not produce a diagnostic for this error; however, incorrect results may be generated.

e. Rewind -- Rewind should not be issued for SYSOUT data sets. If it is used, no output is produced.

f. Backspace -- The backspace statement backspaces one logical record rather than one physical record. Care should be taken when reading, backspacing, and writing the same data set, since a read operation followed by a write operation may give unpredictable results. This is a known error in the FORTRAN I/O routines which will be corrected with release 20.

g. Real Element Assignment -- The assignment of a real element to a complex variable may cause problems; compilation may be deleted or an incorrect object code program check may be made during execution.

6.2.2 PL/I

The PL/I language originated from efforts of a joint SHARE and IBM committee. PL/I combines many of the functional capabilities of FORTRAN IV, COBOL, ALGOL, and list processing languages. In addition, some features were incorporated which are not found in any of the existing high-level programming languages. IBM adopted PL/I as a major programming language for the S/360 computers and fostered the development of the language. Some of the significant features of PL/I are:

- The capability to specify actions to be taken in the case of hardware interrupts, errors not related to hardware such as data conversion, and programmer-specified conditions
- Multi-tasking facilities
- Bit and byte string manipulation, including substring, concatenation and boolean operations
- Extensive debugging facilities, providing for monitoring the setting of variables, checking for subscripts going out of range of an array, and tracing the flow through specified areas of a program

Details on these and other facilities may be found in the PL/I Primer, Form GC28-6808, and the PL/I Reference Manual, Form GC28-8201. PL/I for FORTRAN USERS, Form GC20-1637, is also useful to FORTRAN programmers.

6.2.2.2 PL/I Compiler

The PL/I processor consists of a compiler which accepts statements in the PL/I language and calls subroutines from the PL/I library.

The program name of the PL/I compiler is IEMAA. This name is used in both the IBM-supplied procedures (PL1LFC, PL1LFCL, PL1LFCLG, PL1LFLG) and the GSFC PL1 procedure.

The compiler is comprised of a control module that remains in main storage throughout compilation, and a series of subroutines (phases) that are loaded and executed in turn by the control module. One phase is the preprocessor (compile-time processor) which can modify source statements or insert additional source statements before compilation commences. Because PL/I may use either a 48-character set or 60-character set, a preprocessor phase is needed to convert the 48-character set to the 60-character set.

6.2.2.2.1 Data Sets

The compiler requires several optional data sets; the exact number of data sets depends on the optional facilities requested. These data sets and their characteristics are shown in Tables 6.2-5 and 6.2-6.

Additional information may be found in the PL/1 (F) Programmer's Guide (GC28-6594).

6.2.2.2.2 Options

PL/I has a wide range of options which may be specified at compile time. These options are shown in Table 6.2-7. Because the PARM=field is limited to 100 characters, abbreviated names were developed and are listed with the standard default values (GSFC default values are also shown).

The GSFC default options for PL/I are shown below. Those which differ from the IBM standard default options are flagged with an asterisk.

```
SIZE=200000 * (225280 on Model 65)      LOAD
OPT=1                                   NODECK
STMT *                                  LINECNT=58 *
--                                      OPLIST
M91 *                                   SOURCE2
NOEXTDIC                                SOURCE
NOMACRO                                 NONEST
COMP                                    ATR *
NOMACDCK                                XREF *
CHARG60                                 EXTREF *
EBCDIC                                  NOLIST
SORMGIN=(2,72)                          FLAGW (FLAGE on Model 95)
```

Table 6.2-5. PL/I Compiler Optional Data Sets

ddname	Purpose	Associated Compiler Option
SYSIN	Primary input (PL/I source statements)	
SYSPUNCH	Punched card output	DECK, MACDCK
SYSLIN	Load Module output	LOAD
SYSUT1	To contain overflow from main storage	
SYSUT3	Storage for:	
	1. Converted source module when 48-character set is used	CHAR48
	2. Source statements generated by preprocessor	MACRO, COMP
SYSPRINT	Listing	
SYSLIB	Library containing source statements for insertion by preprocessor	MACRO

Table 6.2-6. PL/I Compiler Optional Data Sets Characteristics

ddname	Possible Device Classes	Record Format	Reserved Buffer Area (in bytes)	No. of Buffers	Record Size (in bytes)	Default Block Size (in bytes)
SYSIN	SYSDA or input job stream (specified by DD *)	F,FB,U	1000	2	100 (max)	-
SYSPUNCH	SYSDA,SYSOUT=B	F,FB	400	1	80	7280
SYSLIN	SYSDA	F,FB	400	1	80	7280
SYSUT1	SYSDA	F	-		1024	-
SYSUT3	SYSDA	F,FB,U	160		80	-
SYSPRINT	SYSDA or SYSOUT=A	V,VB	258	2	125	7254
SYSLIB	SYSDA	F,FB,U	-		100 (max)	

Table 6.2-7. Compiler Options, Abbreviations, and Standard Defaults

	Compiler Options	Abbreviated Names	Standard Defaults
	SIZE=yyyyyyK 999999 MAX	SIZE	999999
	OPT=n	O	O=1
Control	STMT|NOSTMT	ST|NST	NOSTMT
Options	OBJNM=aaaaaaaa	N	-
	M91|NOM91	M91|NOM91	NOM91
	EXTDIC|NOEXTDIC	ED|NED	NOEXTDIC
Preprocessor	MACRO|NOMACRO	M|NM	NOMACRO
Options	COMP|NOCOMP	C|NC	COMP
	MACDCK|NOMACDCK	MD|NMD	NOMACDCK
Input	CHAR60|CHAR48	C60|C48	CHAR60
Options	BCD|EBCDIC	B|EB	EBCDIC
	SORMGIN=(mmm,nnn [,ccc])	SM	SM=(2,72)
Output	LOAD|NOLOAD	LD|NLD	LOAD
Options	DECK|NODECK	D|ND	NODECK
	LINECNT=xxx	LC	LC=50
	OPLIST|NOOPLIST	OL|NOL	OPLIST
	SOURCE2|NOSOURCE2	S2|NS2	SOURCE2
	SOURCE|NOSOURCE	S|NS	SOURCE
Listing	NEST|NONEST	NT|NNT	NONEST
Options	ATR|NOATR	A|NA	NOATR
	XREF|NOXREF	X|NX	NOXREF
	EXTREF|NOEXTREF	E|NE	NOEXTREF
	LIST|NOLIST	L|NL	NOLIST
	FLAGW|FLAGE|FLAGS	FW|FE|FS	FLAGW

6.2.3 ASSEMBLER (F)

The OS/360 Assembler F is a processor which accepts programs written in a symbolic language (ALC). This is a non-specialized language providing a mnemonic for each machine instruction and a set of pseudo operations used for defining data areas, boundary alignment, base register usage, etc. In addition, it has a powerful macro capability, allowing the user to define and use macro instructions of his own design, and to use any of the standard system macros. The system macros offer a convenient way for the ALC programmer to request supervisor and I/O services.

The program name for the ASSEMBLER F processor is IEUASM. The assembler is invoked by executing the cataloged procedure, // EXEC ASSEMBLY. A minimum region of 100K bytes is required for this program.

6.2.3.1 Data Sets

Table 6.2-8 lists the assembler data set requirements and characteristics based on a minimum core size of 44k bytes. Because the region specified in the GSFC procedure library is 100K bytes, block sizes and buffer numbers may be increased. The SYSLIB data set contains the macro library. Additional macro and/or source module libraries may be concatenated to SYSLIB.

6.2.3.2 Options

The assembler options for the model 95 are the standard IBM default options as shown below:

```
PARM='NOLOAD,DECK,LIST,NOTEST,XREF,LINECNT=55,ALGN,OS,NORENT'
```

The prefix NO is added or deleted accordingly to request the opposite of the default value. The exceptions to this are LINECNT where a new value between 01-99 must be specified, and the operating system name which will be OS or DOS.

TABLE 6.2-8. ASSEMBLER F DATA SET CHARACTERISTICS

	SYSIN	SYSLIB	SYSPRINT	SYSPUNCH	SYSGO	SYSUT1 SYSUT2 SYSUT3
LRECL	Fixed at 80	Fixed at 80	Fixed at 121	Fixed at 80	Fixed at 80	N/A
RECFM ①	User must specify in LABEL or DD card F, FS, FBS, FB, FBST, FBT	User must specify in LABEL or DD card F, FS, FBS, FB, FBST, FBT	F and M set by assembler, user may specify B and/or T in label or DD card FM, FMB, FMT, FMBT	F set by assembler, user may specify B and/or T in label or DD card F, FB, FT, FBT	F set by assembler user may specify B and/or T in label or DD card F, FB, FT, FBT	Fixed for U
BLKSIZE ②	User must specify in LABEL or DD card, must be a multiple of LRECL	User must specify in LABEL or DD card, must be a multiple of LRECL	Optional, but must be a multiple of LRECL; if omitted BLKSIZE = LRECL	Optional, but must be a multiple of LRECL; if omitted BLKSIZE = LRECL	Optional, but must be a multiple of LRECL; if omitted BLKSIZE = LRECL	User can not specify maximum of 3624 minimum of 1739
BUFNO	Optional; if omitted 2 is used	Set by assembler to 1	Optional; if omitted 2 is used	Optional; if omitted 3 is used for unit record and 1 for other devices	Optional; if omitted 3 is used for unit record and 1 for other devices	User can not specify either 1 or 2
For 44K availability	BLKSIZE times BUFNO can not be greater than 3600	BLKSIZE can not be greater than 3600 ④	BLKSIZE times BUFNO can not be greater than 1210	BLKSIZE times BUFNO can not be greater than 400	BLKSIZE times BUFNO can not be greater than 400	
For calculating core requirements	L1 = BLKSIZE times BUFNO	L2 = BLKSIZE	L3 = BLKSIZE times BUFNO	L4 = BLKSIZE times BUFNO	L5 = BLKSIZE times BUFNO	

③ Minimum core required for the assembler is the largest of the following: (1) 45056
(2) $L_1 + L_2 + 41000$
(3) $L_3 + L_4 + L_5 + 41000$

③ Maximum core that the assembler can effectively use = $L_4 + L_5 + 535{,}000$

① U = undefined, F = fixed length records, B = blocked records, S = standard blocks, T = track overflow, M = machine code carriage control.

② Blocking is not allowed on unit record devices. Blocking on other direct access can not be greater than the track size unless T is specified on RECFM.

③ For MVT environment add 5,000 for core required.

④ A smaller blocksize may have to be specified for SYSLIB if global or local dictionaries overflow. See item 4 under "Correction of Dictionary Overflow."

6.2.4 RPG

The S/360 Report Program Generator (RPG) language is used to generate reports from one or more input files and to print the reports in a user-defined format. It is efficient and easy to use but is less sophisticated than other high-level languages. RPG is a problem-oriented language in which each program is designed to print a specific report from a specific file or files.

The program name for the RPG processor is IESRPG. The EXEC statement is in the form // EXEC PGM=IESRPG which must be coded by the programmer unless the cataloged procedure is used, in which case it would be // EXEC RPGEC.

6.2.4.1 Data Sets

The RPG compiler can use seven data sets (five are required). Each data set has a specific ddname, function, and device requirements. All but the SYSIN data set may be included in a cataloged procedure. These data sets are listed in Table 6.2-9.

The GSFC cataloged procedure has a slightly modified version of the IBM compile procedure for RPG. BLKSIZE is set at BLKSIZE=400, and the space parameter is different.

6.2.4.2 Options

The RPG processor has a limited number of options which are passed to the compiler through the PARM field in the EXEC statement:

```
                DECK       ,LOAD      ,LIST
PARM = ' NODECK     ,NOLOAD    ,NOLIST  '
```

The programmer specifies the options which are defined as:

DECK -- The object module is placed on the device specified in the SYSPUNCH DD statement, (usually the card punch).

LOAD -- The object module is placed on the device specified in the SYSGO DD statement, (usually intermediate storage).

LIST -- An output listing is written on the device specified in the SYSPRINT DD statement.

The underlined PARM options are the default values that will be assumed if the PARM option is not specified.

Table 6.2-9. RPG DD Names Required

ddname	FUNCTION	DEVICE REQUIREMENTS
SYSIN	reading the source program	• card reader • intermediate storage
SYSPRINT	writing the storage map, linking, and messages	• printer • intermediate storage
SYSPUNCH	output data set for the object module deck	• card punch • intermediate storage
SYSUT1	work data set needed by the compiler during compilation	• direct-access • magnetic tape
SYSUT2	work data set needed by the compiler during compilation	• direct-access • magnetic tape
SYSUT3	work data set needed by the compiler during compilation	• direct-access • magnetic tape
SYSGO	output data set for the object module used as input to the Linkage Editor	• direct-access • magnetic tape

6.3 LARGE UTILITIES

The system and data set utilities are described in Section 9 of this User's Guide. The Linkage Editor, Loader, and Sort/Merge programs perform utility functions; since they are larger processors, their descriptions are included in this section. The Linkage Editor or Loader is required to prepare the output of any of the language processors for execution. Sort/Merge is a generalized program used for sorting data contained in one or more data sets.

6.3.1 LINKAGE EDITOR

The Linkage Editor prepares the output of the language processors for execution. Primary inputs to the Linkage Editor are object modules and Linkage Editor control cards. Additional inputs can be either object modules and control statements, or load modules. The Linkage Editor processing facilities are provided either automatically or in response to control statements prepared by the programmer. These facilities combine and edit modules to produce an executable load module.

The primary output of the Linkage Editor is a load module which is placed in a library (a partitioned data set) as a named member. This library may be either permanent or temporary. In contrast, the Loader does not produce a load module that can be saved. The secondary output of the Linkage Editor is diagnostic output.

The level F version of the Linkage Editor is in use on the M&DO computers.

The statement most commonly used to invoke the Linkage Editor is // EXEC PGM=IEWL, which invokes the largest Linkage Editor design available on the system. On the M&DO computers, this statement invokes the 128k design of the level F Linkage Editor. A particular design level may be invoked by using its program name in the EXEC statement if that level is available (see the footnotes to Table 6.3-1). The cataloged procedures, LINK and LINKGO, are the recommended means for users to invoke the Linkage Editor. A listing of these procedures and details in their use are presented in paragraphs 19.3.2.1 and 19.3.2.3, respectively.

The user should refer to the IBM Manual, Linkage Editor and Loader (GC28-6538) for a more complete description of the Linkage Editor.

6.3.1.1 Options

As with the language processors, the Linkage Editor has several options which increase its versatility. These may be divided into several categories as follows:

Module Attributes	Meaning
DC	Downwards compatible
HIAR*	Hierarchy
NE	Not editable
OL	Only loadable
OVLY	Overlay
REUS	Re-usable
RENT	Re-enterable
REFR**	Refreshable
SCTR	Scatter format
TEST	Test (use of TESTRAN)

Space Allocation	Meaning
SIZE	The amount of main storage to be used by the level F Linkage Editor
DCBS	Specifies blocksize for the SYSLMOD data set

Output Options	Meaning
LIST	List Control Statements
MAP	Request a MODULE MAP
XREF	Request cross reference table

Special Processing Options	Meaning
XCAL	Exclusive Call
LET	Let execution continue
NCAL	No call (do not try to resolve external references)

*Not supported on the M&DO computers

**Supported only on the Model 65. Used primarily by systems programmers.

Each option desired must be explicitly stated in the PARM parameter of the EXEC card. Some options are stated in the cataloged procedures, while others must be coded by the programmer. The user should refer to Section 19 of this User's Guide, or to a listing of the Procedure Library (PROCLIB) to determine if the options included meet his requirements. Note that, when overriding the PARM parameter in a cataloged procedure, all options desired must be explicitly stated in the override statement; otherwise, those options not stated will default to their system-generated value.

6.3.1.2 Data Sets

The Linkage Editor uses five data sets (four are required). The DD statements for these data sets must use the preassigned ddnames given in Table 6.3-2.

The DCB characteristics of these data sets are given in Table 6.3-3.

Table 6.3-1. Linkage Editor Design Levels

PROGRAM NAME	MINIMUM CORE[4] (in bytes)	USED ON IBM MODEL 95	75	65
IEWLF440	44k		X[1]	
IEWLF880	88k		X[1]	
IEWLF128	128k	X	X[1]	X
IEWL	--	X[2]	X[2]	X[2]
LINKEDIT	--	X[3]	X[3]	X[3]

1
On the model 75, all F level Linkage Editor names invoke the 128k design level.

2
Using PGM=IEWL on the M&DO computers invokes the largest Linkage Editor design available, which is the 128K design level.

3
Invokes the 128k design level of the Linkage Editor.

4
Add 8K for system overhead.

Table 6.3-2. Linkage Editor ddnames

Data Set	ddname	Required
Primary input data set	SYSLIN	Yes
Automatic call library	SYSLIB	Only if the automatic library call mechanism is used
Intermediate data set	SYSUT1	Yes
Diagnostic output data set	SYSPRINT	Yes
Output module library	SYSLMOD	Yes

Table 6.3-3. DCB Requirements

		LRECL	BLKSIZE	RECFM
Primary input	SYSLIN	80	80 400,800,3200	F,FS FB,FBS
Secondary Input	Object modules and/ or control statements	80	80 400,800,3200	F,FS FB,FBS
	Load modules • SYSLIB • Included modules	Maximum for device, or one-half of value$_2$ of SIZE option whichever is smaller	equal to LRECL	U
Output	SYSPRINT	121 121	121 605,1210,4840	FM FBM
	SYSLMOD	Maximum track size for device or ½ of value$_2$ of SIZE option, whichever is smaller	equal to LRECL	U

6.3.2 LOADER

The Loader combines in one job step the basic editing functions of the Linkage Editor and the loading functions of program fetch. It is designed for high-performance loading of modules that do not require the special processing facilities of the Linkage Editor, as does overlay. The Loader does not produce load modules for program libraries.

The Loader can be referred to by its program name, IEWLDRGO, or its alias, LOADER. It can be invoked through the EXEC statement // EXEC PGM=LOADER or through the LOAD, ATTACH, LINK, or XCTL macro instructions. See paragraph 19.3.2.3 for a description of the LOADER cataloged procedure and its use.

6.3.2.1 Data Sets

The loader uses three DD statements -- SYSLIN, SYSLIB, and SYSLOUT. (These ddnames can be changed during system generation with the LOADER macro instruction.) The SYSLIN DD statement must be used in every loader job. The other two statements are optional.

The following considerations apply to the DCB parameter of SYSLIN, SYSLIB, and SYSLOUT:

- For better performance, BLKSIZE and BUFNO can be specified.
- If BUFNO is omitted, BUFNO=2 is assumed.
- Any value given to BUFNO is assumed for NCP (number of channel programs).
- If RECFM=U is specified, BUFNO=2 is assumed, and BLKSIZE and LRECL are ignored.
- RECFM=V is not accepted.
- RECFM=FBSA is always assumed for SYSLOUT.
- If RECFM is omitted, RECFM=F is assumed for SYSLIN and SYSLIB.
- If BLKSIZE is omitted, the value given to LRECL is assumed.
- LRECL=121 is always assumed for SYSLOUT.
- If LRECL is omitted, LRECL=80 is assumed for SYSLIN and SYSLIB.

Table 6.3-4 illustrates the basic format of the Loader input deck. Table 6.3-5 is a load-and-go procedure using the SYSLIN data set as the only input. Table 6.3-6 represents a Loader program using the SYSLIB and SYSLOUT data sets and having program data in the input stream.

6.3.2.2 Options

Because of the load-and-go function of the Loader, the PARM operand of the EXEC statement is used to specify options for the Loader and the loaded program. The PARM field has the following format:

```
PARM=(loaderoptions/programoptions)
```

Those options before the slash apply to the Loader. Those following the slash, if any, are passed to the loaded program.

Table 6.3-4. Input Deck for the Loader (Basic Format)

```
//name          JOB       parameters
//name          EXEC      PGM=LOADER,PARM=(parameters)
//SYSLIN        DD        parameters
//SYSLIB        DD        parameters                          (optional)
//SYSLOUT       DD        parameters                          (optional)
//       (optional DD statements and data required for loaded program)
```

Table 6.3-5. Input Deck for a Load Job

```
//LOAD          JOB       MSGLEVEL=1
//LDR           EXEC      PGM=LOADER
//SYSLIN        DD        DSNAME=MASTER,DISP=OLD
//              (DD statements and data required for execution of MASTER)
```

Table 6.3-6. Loader and Loaded Program Data

```
//LOAD          JOB       MSGLEVEL=1
//LDR           EXEC      PGM=LOADER,PARM=MAP
//SYSLIB        DD        DSNAME=SYS1.FORTLIB,DISP=SHR
//SYSLOUT       DD        SYSOUT=A
//FT06F001      DD        SYSOUT=A
//SYSLIN        DD        *
                (Loader data)
//FT05F001      DD        *
                (Loaded program data)
```

6.3.3 SORT/MERGE

The OS/360 Sort/Merge package provides the capability to re-order records within files:

- Sorting and merging operations may be performed on from 1 to 64 fields which may be in any combination of the following formats -- character, decimal (packed or zoned), arithmetic (fixed or floating), or binary strings.
- The sorting and merging keys may be from 1 to 256 bytes long, and different keys may be ordered by different rules (ascending, descending, or user-specified) within a single run.
- The user may specify the type of sort used (polyphase, oscillating), and the type of intermediate storage (tape or disk).
- The user may bypass the first "n" records on the input.
- The user may request checkpoint/restart dumps during the course of lengthy jobs.
- Input data may be sequential data sets of fixed or variable length, and may be blocked or unblocked.

The name of the Sort program is IERRCOOO. It requires a minimum of 15k bytes main storage; however, Sort/Merge performance improves as the amount of main storage available to the program increases. Approximately 44k bytes of main storage are required for efficient operation. At least one selector channel or one multiplexor channel is required. The amount of intermediate storage required for sorting operations depends on the size of the input data set. At least three work units (data sets) are required. More may be used. They may be on the same physical device, but it is more efficient to separate them. Refer to IBM Form #GC28-6662 for Sort/Merge timing estimates. See paragraph 19.3.3 for a description of the Sort cataloged procedure and its use.

6.3.3.1 Data Sets

A variety of DD statements are required depending on the use of a cataloged procedure, sort-only operations, merge-only operations, use of the checkpoint facility, and user-written modification routines. Table 6.3-7 provides a JCL summary for the Sort/Merge program.

6.3.3.2 Options

The Sort/Merge program must know what to do with the input data. This information is provided by five Sort/Merge control statements:

Table 6.3-7. Summary of Job Control Language Statements for Sort/Merge (Sheet 1 of 2)

Statement		Purpose	When Required
//jobname	Job	Introduces the job.	At all times.
//stepname	EXEC	Introduces the step.	At all times.
//SYSPRINT[1]	DD	Used by Linkage Editor.	When you do not use a cataloged procedure and have modification routines that require link editing.
//SYSLMOD[1]	DD	Defines Linkage Editor output data set.	Same as for SYSPRINT.
//SYSUT1[1]	DD	Defines work area for Linkage Editor.	Same as for SYSPRINT.
//SYSLIN[1]	DD	Defines input data set for Linkage Editor.	Same as for SYSPRINT.
//SORTLIB[1]	DD	Defines data set that contains Sort/Merge program modules.	When you do not use cataloged procedures SORT or SORTD.
//SYSOUT[1]	DD	Defines system output data set.	Same as SORTLIB.
//SORTIN	DD	Defines input data set for a Sort.	For a sort, at all times unless LINK, ATTACH, or XCTL is used to invoke sort and the input data set is inserted by your routine at Sort/Merge exit E15. Not used for a merge.

Table 6.3-7. Summary of Job Control Language Statements for Sort/Merge (Sheet 2 of 2)

Statement		Purpose	When Required
//SORTIN01-16	DD	Define input data sets for a merge.	For a merge, at all times. Not used for a sort.
//SORTWK01-32	DD	Define intermediate storage data sets for a sort.	For a sort, at all times. Not used for a merge.
//SORTOUT	DD	Defines Sort/Merge output data set.	At all times, unless LINK, ATTACH, or XCTL is used to invoke sort and your routine disposes of output via Sort/ Merge exit E35.
//SORTMODS	DD	Defines a temporary data set for your modification routines in SYSIN.	When you supply modification routines through the system input stream.
//SORTCKPT	DD	Defines data set for checkpoint records.	When you use the checkpoint facility.
//SYSIN	DD *	Indicates that data set containing Sort/Merge control statements follows in input stream.	At all times.

1
These data sets are provided by the GSFC-cataloged SORT procedure.

a. SORT Statement -- This statement provides information about control fields and data set size. The statement is used with a sort job; it is not used for a merge-only job.

b. MERGE Statement -- This statement provides the same information as a SORT statement, and is used with a merge job. This statement is not used for a sort operation.

c. RECORD Statement -- This statement provides record length and type information, and is required only when modification routines change record lengths during Sort/Merge execution.

d. MODS Statement -- This statement associates modification routines with particular Sort/Merge program exits and is required only when modification routines to be executed at Sort/Merge exits are supplied. (Section 3, Program Modification, describes these exits and the requirements for routines that use them.)

e. END Statement -- This statement signifies the end of a related group of Sort/Merge control statements and is not required.

Further information on these options can be found in the IBM manual Sort/Merge (GC28-6543).

SECTION 7

ADDED PROCESSORS

7.1 GENERAL DISCUSSION

In the wide range of data processing operations and data manipulations there are many areas which are not covered by the IBM-supplied processors. These requirements are usually filled by proprietary packages, such as the Boole and Babbage problem program analyzer, and user-written routines such as the data manipulation routines. Many of these routines, such as FORMAC, GTS, and Re-entrant Assembler, are written to extend the capabilities of existing processors.

7.2 BOOLE AND BABBAGE

The Problem Program Analyzer, a program that analyzes and measures Problem Program Efficiency (PPE), is part of the Boole and Babbage System Measurement Software (SMS/360) for S/360 computers. The programmer can use the Problem Program Analyzer to locate areas in his program where large amounts of time are consumed, and thus determine the parts of the program where efficiency might be increased.

The PPE analyzer is recommended for those users who have programs to be used for a large number of hours, and who are willing to spend the needed time to re-work those sections of their programs which require large amounts of time. This is usually confined to a very few small blocks of code and does not normally involve extensive rewriting of programs.

PPE is particularly helpful when used during the debugging stages of programs under development. It can be used to test alternative coding techniques and to indicate potential sources of wait time.

The PPE program consists of two parts. The first part is the Extractor, which samples the execution of the user's program (at intervals which can be specified by the user) and gathers data on what is happening. The second part is the Analyzer, which reports these data.

The Extractor is initiated within the same step as the problem program, which runs as a subtask under the Extractor. For best results, analysis of a production program should be done during an actual production run. The user can specify various levels of detail for the Code Activity Report developed by the Analyzer program. This report may be prepared to cover all or selected portions of the data from one or more Extractor data sets.

The user must furnish his problem program in object form on cards, tape, or disk, or as a load module on disk. In any case, when he compiles it, he must include the list parameter in his EXEC card, so that he will receive a listing of his object program for later comparison with the Analyzer report. (The parameter is "LIST" for FORTRAN, PL/I, and assembly language.)

If the test is not being done during an actual production run, the user may also supply test data to be processed by the problem program.

The user may also input the time sampling interval and any identifying information, such as programmer name, machine ID, and a unique run number, that he wishes printed on the Code Activity Report.

The Code Activity Report furnishes the user with the following items of information which may be used along with his object program listing to help determine where he may improve his program:

1. Identifying information such as programmer's name and location.

2. For each data set provided by the Extractor program to the Analyzer program, such items as DS name, JOB name, step name, extraction date and time, region bounds, sample bounds (relative to the region), sample interval in milliseconds, and related data.

3. Percentage of activity (excluding I/O wait), outside of and within the sample boundaries, giving a measure of overhead for the program, such as the overlay supervisor.

4. Percentage of time spent in I/O wait state for each data set used by the program (SYSIN, SYSPRINT, SYSUT1, SYSABEND, etc.), thus identifying potential I/O improvements through blocking and buffering.

5. Module map of load modules encountered in the program being tested, showing for each load module its address relative to the region, percent of run time, whether it contained overlays, and whether a report on it was included in the Analyzer report.

6. Study report for each specified load module, or in the case of overlay programs for each requested segment, including:

7. Study report for a selected segment of the load module, including:

    a. Bounds for the study report. (These can be specified to be less than the entire module to facilitate examination of small areas of code.)

    b. Percentage of executed instructions (excluding I/O wait) outside of and within the study boundaries.

    c. Percentage of time spent in I/O wait state for each data set.

    d. Percentage of time spent in a wait state (excluding (I/O wait), with addresses where each wait state occurred.

    e. Percentage of time spent executing SVCs.

f. A histogram which breaks the load module (or study bounds) into small intervals. The percent of run time spent in each interval is listed along with a cumulative count. The default interval size is 32 bytes, but this can be varied by the analyst. By manipulating the study bounds and histogram interval size and working back through Linkage Editor and compiler object listings, it is possible to identify those source statements which cause the most execution time.

Programmers desiring to use the PPE analyzer should contact the Boole and Babbage representative, Mr. David Morgan, in Building 3, Room 133-B, extension 6796. Mr. Morgan will provide assistance in setting up the PPE run and in interpreting the information supplied in the Code Activity Report. On the basis of this information, he will determine what portions of the program, if any, may be improved.

Note: Boole and Babbage should not be used in programs where the system library routine REMTIM is used, unless REMTIM is not entered during the time Boole and Babbage is running.

The cataloged procedure BB is used for executing the Boole and Babbage Problem Program Analyzer. The three steps in this procedure are a LINK step, a GO step, and an analysis step (BSTEP2).

The input to the LINK step is the same as that for the LINK procedure. The program to be analyzed should first be compiled using the LIST option to get an assembler language listing of the source program. This list will be used later as an aid in the analysis of the program. The output of the LINK step is a load module, called FORT1, which is placed in the partitioned data set &&BOOLIB.

The GO step executes the program PPDEXT1 which, in turn, executes the problem program named FORT1. While the problem program is executing, the Extractor program takes readings, at specified intervals, of the amount of time spent in the wait state and in executing sets of instructions. These data are placed in the data set &&PPDEXT, which is passed to the analyzer program, PPANAL, in the third step, BSTEP2.

The third step executes the analyzer program, PPANAL, which processes the data in &&PPDEXT and prints the report.

In the following example, the BB procedure is executed following a compile step:

```
//stepname          EXEC      BB
//GO.INPUT1         DD        *
FORT1,2,DSOW
//GO.DATA5          DD        *
     (problem program input data)
//BSTEP2.INPUT2     DD        *
ANAME=programmer
```

The input control card to PPDEXT1 is:

```
FORT1,n,DSOW
```

where:

FORT1 is the procedure assigned name of the problem program

n is an integer, where n multiplied by 16.6 milliseconds is the timing interval between samples

DSOW is Data Set Oriented Wait

The optional input control card to PPANAL is:

```
ANAME=programmer
```

where:

programmer is any combination up to 24 characters, including leading blanks and punctuation (other than commas, which are not allowed).

Additional DD statements for the LINK and GO steps may be included as required, but must conform to the standard rules for JCL and cataloged procedures.

Users requiring assistance with the BB procedure should contact Mr. David Morgan in Building 3, extension 6796. User's Guides, giving details of various parameters and options, plus instructions for analysis, are also available from this source.

A listing of this procedure follows:

```
MEMBER NAME     BB
ALIASES    BOOLE
//DEFAULT PROC       REG1=200K,REG2=125K,LTRK=20,BTRK=20
//                   D1='SYS2.DUMMY',V=G1SYS1,U=DISK
//LINK     EXEC      PGM=IEWL,PARM=(MAP,LIST),COND=(5,LT),REGION=300K
//LOADLIB        DD  DSN=SYS2.LOADLIB,DISP=SHR
//NEWLIN         DD  DUMMY
//SYSLIB         DD  DSN=SYS2.DUMMY,DISP=SHR
//               DD  DSN=SYS2.DUMMY,DISP=SHR
//               DD  DSN=SYS1.FORTLIB,DISP=SHR
//               DD  DSN=SYS2.GSFCLIB,DISP=SHR
//               DD  DSN=SYS1.PL1LIB,DISP=SHR
//               DD  DSN=SYS1.TELCMLIB,DISP=SHR
//               DD  DSN=SYS2.LOADLIB,DISP=SHR
//               DD  DSN=SYS1.SSPAK,DISP=SHR
//SYSLMOD        DD  DSN=&&BOOLIB(FORT1),DISP=(NEW,PASS),UNIT=DISK,
//                   SPACE=(TRK,(&LTRK,10,1))
//SYSPRINT       DD  SYSOUT=A,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=1210),
//                   SPACE=(TRK,(3,3))
//SYSUT1         DD  UNIT=DISK,SPACE=(TRK,(&LTRK,20)),SEP=SYSLMOD
//TAPELIB        DD  DUMMY,VOL=SER=TAPEIN,UNIT=(9TRACK,,DEFER),LABEL=(,BLP),
//                   DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSLIN         DD  DSN=&&OBJMOD,DISP=(OLD,DELETE),DCB=RECFM=FB
//               DD  DDNAME=OBJECT
//GO       EXEC      PGM=PPDEXT1,COND=(5,LT),REGION=&REG1
//STEPLIB        DD  DSN=SYS2.SMSLIB,DISP=SHR
//               DD  DSN=&&BOOLIB(FORT1),DISP=(OLD,PASS)
//               DD  DSN=&D1,VOL=SER=&V,UNIT=&U,DISP=(SHR,PASS)
//FT05F001       DD  DDNAME=DATA5
//FT06F001       DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                   SPACE=(CYL,(1,1))
//FT07F001       DD  DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),
//                   SPACE=(TRK,(1,20))
//* INSERT           //GO.FT07F001 DD DSN=&&DECK,SYSOUT=B  FOR PUNCHED OUTPUT
//SYSPRINT       DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629),
//                   SPACE=(TRK,(1,20))
//PPE2EOU2       DD  DSN=&&PPDEXT,DISP=(NEW,PASS),DCB=(RECFM=F,BLKSIZE=512),
//                   SPACE=(TRK,(&BTRK,10)),UNIT=DISK
//PPE2EERR       DD  SYSOUT=A,SPACE=(TRK,(1,5))
//PPE2EINN1      DD  DDNAME=INPUT1
//SYSUDUMP       DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//BSTEP2   EXEC      PGM=PPANAL,COND=(5,LT),REGION=&REG2
//STEPLIB        DD  DSN=SYS2.SMSLIB,DISP=SHR
//FT04F001       DD  DSN=&&PPDEXT,DISP=(OLD,DELETE)
//FT05F001       DD  DDNAME=INPUT2
//FT06F001       DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                   SPACE=(CYL,(1,1))
//SYSUDUMP       DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                   SPACE=(CYL,(1,1))
```

7.3 FORMAC

FORMAC is an extension of the PL/I (F) compiler. It provides the capability to perform formal algebraic manipulation of variables and expressions within a PL/I numeric evaluation of formally derived expressions, using any desired data. Thus, algebraic formulas of great complexity may be derived and printed as formal expressions, together with numeric results obtained from evaluation of the expressions using specified data.

The PL/I-FORMAC Interpreter allows the user to set a FORMAC variable equal to a symbolic algebraic expression. This variable may, in turn, be included in another symbolic expression, and in this fashion extensive algebraic formulas may be constructed.

Various subroutines perform editing and formal manipulation of expressions. Included are such capabilities as analytic differentiation, formal function evaluation, expansion of integer powers of expressions, true rational arithmetic, and symbolic complex arithmetic.

Editing capabilities allow formal factorization, combination of terms, and substitution of complicated expressions.

Formally derived expressions may be numerically evaluated and used as PL/I variables in numeric algorithms.

The FORMAC interpreter uses the full PL/I capability; hence, all legal PL/I subroutine names may be used as FORMAC variables.

A limited knowledge of PL/I is sufficient to allow use of FORMAC. For instance, familiarity with PL/I input and output capabilities, BEGIN-END brackets, and the PROCEDURE statement would provide a FORTRAN programmer with a knowledge base sufficient to use FORMAC.

Extensive formal manipulation of algebraic expressions has direct application in analysis, celestial mechanics, and optimization theory, although it is not limited to these fields. The following is intended as an indicative, but not exhaustive, list of suggestions:

1. FORMAC may be used to derive high order Runge-Kutta type formulas for integration, analytic expressions for terms in function approximation, and Taylor-series approximations of arbitrary functions.

2. FORMAC may be used to generate and combine formal power series to obtain analytic expressions for solutions of orbital elements. These expressions may then be formally differentiated to provide error-analysis capabilities.

3. Solutions to linear programming problems, trajectory optimization, and control-type problems often use methods based on gradient techniques requiring expressions for first- and second-order derivatives of complicated expressions in several variables. FORMAC may be used to generate their numeric evaluation.

The FORMAC cataloged procedure consists of a preprocessor, compile, LINK, and GO steps. The PL/I source deck is input to the FORMAC program in the preprocessor step. The output of the FORMAC program is a temporary data set called &SRCE which is passed to the PL/I compiler. The changed version is compiled, linked, and executed.

To execute the FORMAC procedure, the following is coded:

```
//stepname          EXEC      FORMAC
//STEP1.SYSIN       DD        *
     (source deck)
//STEP4.SYSIN       DD        *
     (data)
```

If a punched object deck is required, the following is coded:

```
//stepname          EXEC      FORMAC,PARM=DECK
//STEP1.SYSIN       DD        *
     (source deck)
//STEP2.SYSPUNCH    DD        DSN=DECK,SYSOUT=B
//STEP4.SYSIN       DD        *
     (input data)
```

A listing of this procedure follows:

```
MEMBER NAME     FORMAC
//STEP1   EXEC      PGM=MINIMAC,REGION=200K
//STEPLIB       DD  DSN=SYS2.MINIMAC,DISP=SHR
//SYSPRINT      DD  SYSOUT=A,SPACE=(CYL,(3,1))
//SYSUT3        DD  UNIT=DISK,DCB=(RECFM=FB,LRECL=80,BLKSIZE=80),
//                  SPACE=(80,(4000,2000)),DSNAME=&SRCE,DISP=(,PASS)
//STEP2   EXEC      PGM=IEMAA,PARM='S,ST,NT,SM=(2,80)',COND=(16,EQ,STEP1)
//SYSPRINT      DD  SYSOUT=A,SPACE=(CYL,(5,1))
//SYSLIN        DD  DSNAME=&LOADSET,DISP=(MOD,PASS),UNIT=DISK,
//                  SPACE=(80,(250,100))
//SYSUT3        DD  UNIT=DISK,SPACE=(80,(250,250)),SEP=SYSPRINT
//SYSUT1        DD  UNIT=DISK,SPACE=(1024,(60,60)),,CONTIG),
//                  SEP=(SYSUT3,SYSLIN,SYSPRINT)
//SYSIN         DD  DISP=(OLD,DELETE),DSNAME=&SRCE
//STEP3   EXEC      PGM=IEWL,PARM='XREF,LIST,LET,DCBS',COND=((16,EQ,STEP1),
//                  (9,LT,STEP2)),REGION=300K
//SYSLIB        DD  DSN=SYS2.FORMAC,DISP=SHR
//               DD  DSNAME=SYS1.PL1LIB,DISP=SHR
//SYSLMOD       DD  DSN=&GOSET(30),DISP=(MOD,PASS),UNIT=DISK,
//                  SPACE=(3702,(50,20,1))
//SYSPRINT      DD  SYSOUT=A,SPACE=(CYL,(2,1))
//SYSUT1        DD  UNIT=DISK,SEP=(SYSLMOD,SYSLIB),SPACE=(1024,(200,20))
//SYSLIN        DD  DSN=&LOADSET,DISP=(OLD,DELETE)
//               DD  DDNAME=SYSIN
//STEP4   EXEC      PGM=*.STEP3.SYSLMOD,
//                  COND=((9,LT,STEP3),(9,LT,STEP2),(16,EQ,STEP1))
//SYSPRINT      DD  SYSOUT=A,SPACE=(CYL,(10,2))
//SYSUT1        DD  UNIT=DISK,DCB=(RECFM=F,BLKSIZE=829),
//                  SPACE=(829,1000)
//SYSABEND      DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                  SPACE=(CYL,(10,1))
```

The input to the compile step (STEP2) is the modified version (&SRCE) of the source deck. The output of the compile step is the data set &LOADSET which is passed to the LINK-EDIT step (STEP3). Additional input decks may be input to STEP3 through the SYSIN DD statement:

```
//STEP3.SYSIN       DD        *
     (object deck(s))
```

The output of STEP3 is an executable load module FORMAC stored in the PDS &GOSET. This program is executed by STEP4 of the FORMAC procedure.

7.3.2 REFERENCES

For further documentation on FORMAC, contact Mrs. Pat Barnes, extension 6796, in the GSFC Program Library in Building 3.

7.4 RE-ENTRANT ASSEMBLER

The 360 Re-entrant On-Line Assembler (ROLA 2260) is desiged to perform, in a re-entrant environment, a majority of the functions that the IBM 360 assembler now performs. It is basically compatible with OS Assembly Language. It also has the capability of functioning in three modes of operation: non-conversational from a batch stream; non-conversational from an IBM 2260 terminal using Graphic Terminal Services (GTS); and conversational from the IBM 2260 using GTS.

The 360 ROLA 2260 consists of a control program and three processing programs. The control program determines which processing program is to be executed and in what order each is to be executed. The processing programs translate source and macro code into object code to be executed by the system. This assembler differs from most assemblers in that the time-sharing environment in which it operates tends to curtail excessive I/O activity. One of the effects of this is that no overflow onto external storage of the tabled information (i.e., the symbol table) is permitted in the assembler. Intermediate storage is restricted to the necessary expanded source deck and certain coded information which may be contained in the 80-byte source card images. In this way, the external storage which originally contained the source code is used for all intermediate external requirements if sufficient expansion room is provided.

A special format for input is used. A "side-by-side" format for the source and object code is used. The source statement appears in the left 54 columns of an 80 column card, and the corresponding hexadecimal object code generated by the assembler appears in columns 55-80. Because of this use of the original external storage (80-byte source statements) for intermediate requirements, the flexibility of format allowed in the OS Assembler is not provided by the ROLA 2260. All rules governing the contents of the fields are unchanged.

Provision has been made for the Re-entrant Assembler to accept input from a user library. This library is called the COPY library. It must be organized in the same manner as the SYS1.MACLIB. It must be a partitioned data set, each member containing statements in Re-entrant Assembler format.

A COPY statement in the source program is used to include the statements of the member it references after the COPY statement. The COPY statement usage is the same as defined for the S/360 assembler, except that programmer-defined macros may also be included in the library.

If the source program contains COPY statements, a DD control card must be included in the job control cards for the Re-entrant Assembler to reference

the COPY library. This DD control card follows the SYSLIB DD card as shown below:

```
//SYSLIB      DD   DSNAME=SYS1.MACLIB,DISP=(OLD,PASS)
//            DD   DSNAME=COPLIB,UNIT=2314,VOLUME=SER=xxxxxx,
//                 DISP=(OLD,PASS)
```

where xxxxxx is the serial number of the volume which contains COPLIB.

7.4.1 MODES OF OPERATION

The ROLA 2260 can operate in three modes: batch process, non-conversational on an IBM 2260 terminal, and conversational on an IBM 2260 terminal. The batch process mode allows the programmer to assemble his program in a non-conversational mode by invoking the Re-entrant Assembler through job stream control cards. The Assembler accepts as input a SYSIN source deck and outputs a SYSOUT listing and/or binary object deck.

The two modes initiated from the 2260 terminals are both executed in the same manner. The conversational mode will additionally allow hands-on capability in correcting assembly errors as the program is being processed.

In order to operate the assembler in a non-conversational mode utilizing the IBM 2260, several initial processes must be completed. The source program to be assembled must reside on a direct-access device as either a sequential data set or a member of a partitioned data set. Also, sequential or partitioned data set space must be available for the output unless the user wants to use the input data set space. If this second option is used, care must be taken to insure that non-processed input is not destroyed.

It should be understood that any input to the succeeding phase is destroyed. In this regard, if all three phases are to be executed in the overlapping manner, the hexadecimal listing of the source program is lost and only the object deck code remains for the user.

A deck must be available either in the user's RITS (Remote Input Terminal Service) file or as an input stream deck to initiate the Graphic Terminal Service (GTS) under which the 2260s operate.

The 2260 terminal requested in the GTS deck must be turned on. There is a knob on the right side of the terminal that should be pulled out. Once the GTS deck has been executed, the terminal is ready for operation.

Each mode allows one phase, two phases, or all phases of the assembler to be executed, depending on what the user wants done to his program, macro expansion, assembly, or object code generation. According to the phase or phases to be executed and the location of input and output of each phase, a parameter list is to be constructed by the user for the particular mode in which he is processing.

7.4.2 RESTRICTIONS

1. Since no table overflow is provided in the assembler, the amount of storage provided by the user program must be adequate for all symbol tables, literal pools, the External Symbol Dictionary, and other internal storage requirements. Failure to provide sufficient storage will result in an aborted assembly.

2. The number of references to a symbol which will appear in the Cross Reference Listing is limited to 255. Additional references may be made to an equated symbol.

7.4.3 REFERENCES

For more complete documentation, see program G00310 (REA360) in the GSFC Program Library. A user's guide and program documentation are available.

7.5 GPSS V

GPSS V is a simulation program applicable where the simulation model deals with discrete items, events, and timing, such as in scheduling.

GPSS is useful in the solution of problems involving a network of discrete units where operations are being performed on them, and where the user can describe quantities, timing, and operations in a block diagram.

GPSS requires no previous programming knowledge. The user prepares the block diagram, using certain standard flowchart symbols to represent some step in the action of his system.

GPSS uses "transactions" to represent different things, depending on the system being simulated. For example, a "transaction" could be messages in a communication system, electrical pulses in a digital circuit, or records in a data processing system.

The system that the user wishes to simulate by means of GPSS must be described as a block diagram with blocks representing activities. The sequence in which activities are to be executed is indicated by the lines joining the blocks. Where a choice of activities is desired, the user draws more than one line leaving a block. The condition determining the branching is stated at the block.

GPSS block diagrams (unlike most diagrams in which the form of the diagram depends on the ideas of the user) use blocks that have precise meanings, so that the programming language may be based on them. There are 37 specific block types, each of which represents a characteristic action of systems. In GPSS, a set of subroutines is associated with each type of standard block. Therefore, the programmer must draw his block diagram for GPSS, using only these block types.

Moving through the system being simulated are entities dependent on the nature of the system. For instance, communication systems are involved with movement of messages and data processing systems with records. In simulation, the movement of these entities (transactions) from block to block in simulated time reflects the sequence of events in real time.

The user may create entities and track them through the network, tabulating their paths, waiting times, queue lengths, and other pertinent statistics. The results may be displayed in tabular form or in histogram plots on the printer.

7.5.1 GPSS V APPLICATIONS

Several scheduling applications might be suitable for solution by GPSS. One might be the simulation of a computer system in order to uncover bottlenecks. Another might measure the response of the CRBE/RITS service and the effect of different types of terminals or numbers of dial-up stations.

Another application might be in the simulation of new scientific satellites in which the various experiments encode their data digitally and must compete for the PCM telemetry transmission equipment. A GPSS simulation would measure the data lost, demands on the transmission equipment, and whether such a concept were feasible.

A listing of this procedure follows:

```
MEMBER NAME    GPSS
//SOURCE       EXEC PGM=DAG01V,PARM=B,REGION=300K
//STEPLIB      DD   DSN=SYS2.GPSS,DISP=SHR
//DINTERO      DD   UNIT=DISK,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7144
//DINTWORK     DD   UNIT=DISK,SEP=DINTERO,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7236
//DOUTPUT      DD   SYSOUT=A,DCB=BLKSIZE=7182
//DREPTGEN     DD   UNIT=DISK,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7280
//DSYMTAB      DD   UNIT=DISK,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7112
```

To execute the procedure, the following is coded:

```
//stepname      EXEC      GPSS
//SOURCE.DINPUT1    DD   *
     (formatted input deck)
/*
```

GPSS is available on the M&DO model 95 computer.

7.5.2 REFERENCES

Contact Mrs. Pat Barnes, extension 6796, in the GSFC Program Library, Building 3, for further documentation on GPSS. Refer to the IBM manuals, General Purpose Simulation System V Introductory User's Manual (SH20-0866), and the General Purpose Simulation System V User's Manual (SH20-0851) for more detail on GPSS. The above IBM manuals are available only through purchase from IBM.

7.6 GRAPHICS TERMINAL SERVICE (GTS)

Graphics Terminal Services (GTS) provide support services for the 2250 and 2260 display terminals. GTS is functionally identical for both devices, but they have different operational characteristics because of hardware differences.

The five major function groups are:

1. Log-on/Log-off - This function provides the means to initiate and terminate the terminal.

2. Data set editing - This function allows the user to create, display, and modify card-image data sets on direct-access devices.

3. Job scheduling - This function allows the user to display and modify cataloged procedures for job submission, enter all JCL directly through the terminal, or display and modify the JCL from the preceding job for use in the current job.

4. Job output processing - This function allows the user to visually examine job output and to have any or all output printed, as desired.

5. Job status - This function allows the user to examine all jobs that he has submitted through the terminal.

For more detailed information on GTS, see paragraph 12.1.3.

7.7 BIT-MANIPULATION ROUTINES

The FORTRAN language does not have the bit manipulation capabilities of the 360 Assembler Language. To increase the data processing capabilities of S/360 FORTRAN, several routines have been added to the GSFC Library for the logical manipulation of 32-bit data words. All operate as FORTRAN functions, and the user must be careful of fix/float conversions. The routines were written by Jack Balakirsky, Code 543.

1.	A=AND(B,C)	ANDs real arguments B and C bit by bit
2.	I=LAND(J,K)	AND for integer and logical* expressions
3.	A=OR(B,C)	ORs real arguments B and C bit by bit
4.	I=LOR(J,K)	OR for integer and logical* expressions
5.	A=XOR(B,C)	ORs (exclusive) real arguments B and C bit by bit
6.	I=LXOR(J,K)	Exclusive OR for integer and logical* expressions
7.	A=COMPL(B)	Stores the one's complement of B in A
8.	I=LCOMPL(J)	Ones complement for integer and logical* expressions
9.	SHFTL	Shifts the bit configuration of J to the left, n positions. All 32 bits participate in the shift. High-order bits are shifted out and lost. Zeros are inserted in the low-order vacated positions.

```
INTEGER SHFTL                LOGICAL SHFTL,A,B
.                            .
.                            .
I=SHFTL(J,n)                 A=SHFTL(B,n)
.                            .
.                            .
```

*The function and its arguments must be defined in a LOGICAL statement, i.e., LOGICAL LAND,J,K.

10. SHFTR — Shifts the bit configuration of J to the right, n positions. All 32 bits participate in the shift. Low-order bits are shifted out and lost. Zeros are inserted in the high-order vacated positions.

```
INTEGER SHFTR                LOGICAL SHFTR,A,B
.                            .
.                            .
I=SHFTR(J,n)                 A=SHFTR(B,n)
.                            .
.                            .
```

11. FSHFTL — Shift left for real variables

```
A=FSHFTL(B,n)
```

12. FSHFTR — Shift right for real variables

```
A=FSHFTR(B,n)
```

13. BITON** — Sets to 1 the n^{th} bit position (n=0-31) of B and stores the result in A. B remains unchanged.

```
INTEGER BITON,A,B            LOGICAL BITON,A,B
.                            .
.                            .
A=BITON(B,n)                 A=BITON(B,n)
.                            .
.                            .
```

14. BITOFF** — Sets to zero the n^{th} bit position (n=0-31) of B and stores the result in A. B remains unchanged.

```
INTEGER BITOFF,A,B           LOGICAL BITOFF,A,B
.                            .
.                            .
A=BITOFF(B,n)                A=BITOFF(B,n)
.                            .
.                            .
```

15. BITFLP** — Complements the n^{th} bit position (n=0-31) of B and stores the result in A. B remains unchanged.

```
INTEGER BITFLP,A,B           LOGICAL BITFLP,A,B
.                            .
.                            .
A=BITFLP(B,n)                A=BITFLP(B,n)
.                            .
.                            .
```

**Routines BITON, BITOFF, and BITFLP operate on only one bit of a 32-bit word.

SECTION 8

SYSTEM, PROCESSOR, AND USER LIBRARIES

8.1 GENERAL DISCUSSION

A "library" is a partitioned data set (PDS) that resides on a direct-access volume and has a directory which identifies the members by name and location. Libraries contain routines widely used at an installation and are therefore made easily accessible. OS/360 libraries fall into the following categories:

a. System Libraries -- These libraries are considered part of the operating system and usually reside on the system residence volume, although they may reside on other direct-access volumes. Examples of libraries in this category are SYS1.LINKLIB and SYS1.PROCLIB. Libraries concatenated to the automatic call library (SYSLIB) in the cataloged procedure may be considered to be system libraries. See subsection 8.12 for a description of SYSLIB.

b. Libraries required when using processors available under OS/360 -- The libraries in this category include the IBM-supplied libraries SYS1.FORTLIB (its extension, SYS2.GSFCLIB), SYS1.MACLIB, and SYS1.PLILIB. The routines may be referenced explicitly by the programmer (such as the CALL in a FORTRAN program) or used as required by the system (e.g., the input/output routines used in FORTRAN).

c. User libraries and other libraries that are created or maintained by the user or systems programmer to best accommodate installation requirements -- These include libraries developed by users of such systems as Definitive Orbit Determination System (DODS) and Attitude Determination, as well as programs designed to give the individual capabilities not incorporated into the existing system, such as the Stromberg Datagraphics 4060 package or GPSS360. Private libraries are also defined by this third category.

   A library is a data set and may therefore be created during any job step by defining the library in a DD statement. The library may be given a simple or qualified name and may be passed, kept, deleted, or cataloged. A library may be called a source, object, or load library, depending on the type of modules it contains. In the use of libraries, one may encounter the terms JOBLIB, STEPLIB, and SYSLIB. These are discussed in subsections 8.10, and 8.11, and 8.12.

8.1.1 REFERENCES

IBM System/360 Operating System manuals.

- System Generation (GC28-6554)
- INTRODUCTION (GC28-6534)
- Supervisor and Data Management Services (GC28-6646)
- Supervisor and Data Management Macro Instructions (GC28-6647)

8.2 LINKLIB

The link library (SYS1.LINKLIB) is designated as the System Library. It contains the most frequently used programs, such as the non-resident system routines, language processors, the Linkage Editor and Loader, utilities, and other IBM-supplied programs. It also contains frequently used user-written programs. Any program in this library can be executed by coding PGM=program name in the EXEC statement of a job step. LINKLIB is always available to all steps of all jobs. The control program provides the necessary data control block and establishes the logical relationship between the user's program and the library.

When the system is IPLed, two or more libraries may be concatenated to the link library; for example, on the model 95 LINKLIB consists of SYS1.LINKLIB, SYS2.LINKLIB, and SYS2.GSFCLINK.

On the model 75 LINKLIB consists of SYS1.LINKLIB and SYS2.LINKLIB. Another library, SYS3.LINKLIB, contains Operating System modules, but a JOBLIB or STEPLIB card must be used to concatenate it with LINKLIB. This library is similar to SYS2.GSFCLINK on the 360/95 and contains less frequently used programs, and modules from a different release or compiler version, such as the PL/I Version 4.3 compiler.

8.2.1 REFERENCES

IBM System/360 Operating System manuals.

- System Programmer's Guide (GC28-6550)
- Supervisor and Data Management Services (GC28-6646)

8.3 PROCLIB

A procedure library (PROCLIB) is a partitioned data set containing job control language statements for standard, frequently run jobs. A particular set of such JCL statements in a PROCLIB is referred to as a cataloged procedure.

Cataloged procedures may be referenced on an EXEC card in the job stream to cause the inclusion of the designated set of JCL, thus reducing the number of JCL statements to be supplied by the user. This not only reduces the burden on the user, but also reduces error probabilities in preparation of JCL. Details on the usage of cataloged procedures are presented in Section 5.

There is a procedure library named SYS1.PROCLIB on each of the M&DO computers, containing cataloged JCL procedures for all standard processors (such as FORTRAN, PL/I, and Linkage Editor). A partial list of cataloged procedures in SYS1.PROCLIB for each of the M&DO computers is found in subsection 19.3. The Model 95 has a second procedure library, named SYS2.USERPROC, which is concatenated with SYS1.PROCLIB. This library contains cataloged procedures for frequently used user-written programs. Before a procedure can be placed on the SYS2.USERPROC, the following conditions must be met:

a. The procedure must be checked out - no "JCL errors" are allowed.

b. The procedure must have at least 15 cards or must be used at least five times per day.

A procedure which meets the above conditions may be placed in the SYS2.USERPROC by submitting a written request, accompanied by a listing of the procedure and the deck necessary to update SYS2.USERPROC, to the computer manager, Mr. Harry G. Bitting, Code 543. Subsequent updates must follow the same rules. Please note that SYS2.USERPROC is restricted to procedures for execution of user programs; procedures for compilations, assemblies, or linkage edits will not be approved.

8.3.1 REFERENCES

See subsection 19.3 for a description of the GSFC standard cataloged procedures.

IBM System/360 Operating System manual.

- Job Control User's Guide (GC28-6703) and Job Control Language Reference (GC28-6704).

8.4 SVCLIB

The members of the Supervisor Call (SVC) library (SYS1.SVCLIB) are non-resident SVC routines, the data management access methods, and the system's standard error recovery routines (SER). These members are in load module form.

This library is primarily of interest to the systems programmer.

8.4.1 REFERENCES

IBM System/360 Operating System manuals.

- System Programmer's Guide (GC28-6550)
- System Generation (GC28-6554)

8.5 MACLIB

The macro library, SYS1.MACLIB, is a collection of macro definitions that can be used in assembler language programs at GSFC. Once a macro definition has been placed in the macro library, the definition may be used by writing its corresponding macro instruction in a source program.

A macro definition included in a source deck is called a programmer macro definition, and a macro definition residing in the macro library is called a system macro instruction. There is no difference in function and they will be expanded in the same way. However, because syntax errors are handled differently, a macro definition should be thoroughly debugged as a programmer macro before being entered in the macro library.

8.5.1 REFERENCES

IBM System/360 Operating System manual.

- Assembler Language (GC28-6514)

8.6 FORTLIB

The FORTRAN library, SYS1.FORTLIB, is a PDS which contains a group of FORTRAN subprograms. These programs are provided by IBM to perform specific functions such as mathematical functions and input/output processing. New modules may be added and other modules deleted to meet the needs of a specific installation

SYS1.FORTLIB is one of the libraries concatenated to SYSLIB, the automatic call library. At GSFC, other FORTRAN modules are stored in SYS1.SSP, SYS2.GSFCLIB, SYS2.SC4060, and other libraries.

8.6.1 REFERENCES

IBM System/360 Operating System manuals.

- FORTRAN IV (G and H) Programmer's Guide (GC28-6817)
- FORTRAN IV Library Subprograms (GC28-6596)

8.7 PLILIB

The PL/I subroutine, SYS1.PLILIB, is a system library that houses a set of load modules that, during execution of a PL/I program, supplement the machine instructions generated by the compiler. These modules can be divided into two groups:

a. Modules that serve as an interface between compiled code and the facilities of the operating system -- These modules are concerned primarily with input and output, storage management, and error and interrupt handling.

b. Modules that perform data processing operations during program execution -- These modules handle, for example, input/output editing, data conversion, and many of the PL/I built-in functions.

The model 95 data set which contains routines for Version 4.3 PL/I is SYS2.V43.PL1. The model 75 data set which contains routines for Version 4.3 PL/I is SYS2.PLILIB. When Version 4.3 PL/I is used the appropriate data set must be concatenated to SYSLIB to provide the required routines.

8.7.1 REFERENCES

IBM System/360 Operating System manuals.

- PL/I (F) Programmer's Guide (GC28-6594)
- PL/I (F) Subroutine Library (GC28-6590)

8.8 LOADLIB

SYS2.LOADLIB is a GSFC load module library. On the models 95 and 75, the load module library is concatenated with the automatic call library in the LINK and LINKGO procedures. Programs to be entered in LOADLIB must meet certain conditions of size and usage before being accepted. These conditions are listed below, as stated in the M&DO 360 Computer Bulletin #3:

a. Programs or subroutines must be checked out.

b. They must be used not less than once a day.

c. Their size must not exceed 250k bytes of memory.

d. Each member name must be of the following format -- USRIDXXX (e.g., GAFGR001).

e. A written request accompanied by a copy of the Linkage Editor map must be submitted to the computer manager, Mr. Harry G. Bitting, Code 543, for approval.

f. After approval is granted, procedure SAVEPROG must be used to enter the member into the library. Refer to paragraph 19.3.6 for a description of the SAVEPROG procedure.

8.8.1 REFERENCES

IBM System/360 Operating System manual.

- Linkage Editor and Loader (GC28-6538). See automatic call library.

8.9 TELCMLIB

The members of the telecommunications library (SYS1.TELCMLIB) are load modules which support the optional telecommunications access methods specified at system generation. The access methods supported by SYS1.TELCMLIB are the Basic Telecommunications Access Method (BTAM) and Queued Telecommunications Access Method (QTAM). SYS1.TELCMLIB must be specified at system generation if either or both of these access methods are to be generated. TELCMLIB is concatenated to SYSLIB in the LINKGO procedure on the model 95.

8.9.1 REFERENCES

IBM System/360 Operating System manuals.

- System Generation (GC28-6554)
- Introduction to Teleprocessing (GC30-2007)
- Linkage Editor and Loader (GC28-6538)

8.10 JOBLIB

The major function of the JOBLIB statement is to make programs which reside in a private library available to the operating system. When the JOBLIB statement is encountered, the operating system concatenates the private library with the system library (SYS1.LINKLIB). When a request is made for the program, the operating system searches first in the private library and then in the system library. This is especially useful when working with more than one version of the same program. The alternate version in the private library may be JOBLIBed when its use is required. Removing the JOBLIB card makes available the version in the system library.

The JOBLIB statement defines a private library for the duration of the job. If the private library is cataloged, the operand field requires only the DSNAME and DISP parameters. If the library is not cataloged, volume and unit information must also be provided.

The JOBLIB statement must immediately follow the JOB card. If only the first operand of the DISP parameter is coded, the second operand will default to PASS. The first operand may be NEW, OLD, or SHR. SHR is the recommended disposition for libraries which may be required by other users. If a private library is created (space would have to be allocated on the JOBLIB card) and used within a single job step, it is deleted at the end of the job step unless DISP=(NEW,KEEP) is coded.

Reference back to the JOBLIB is the same as for other DD statements. When a library is JOBLIBed, the user must insure that subroutine libraries having supporting modules are added to the SYSLIB DD statement in the LINK and LINKGO procedures. One example of this (on the M&DO 360/75) is the PL/I Version 4.3 processor which is in SYS3.LINKLIB and the PL/I Version 4.3 subroutines which are in SYS2.PL1LIB. The user must code both:

```
//JOBLIB        DD        DSN=SYS3.LINKLIB,DISP=SHR
```

and

```
//LINK.SYSLIB   DD        DSN=SYS2.PL1LIB,DISP=SHR
```

The libraries are searched in the order in which the DD statements appear, with the system library searched last.

When concatenating private libraries, as when concatenating any data sets, the ddname must be omitted from all the DD statements defining private libraries, except the first DD statement. The first statement must specify a ddname of JOBLIB or STEPLIB. If JOBLIB is specified, the entire group must appear immediately after the JOB statement. If STEPLIB is specified, the entire group would appear as one of the DD statements for a particular step.

8.10.1 REFERENCES

IBM System/360 Operating System manuals.

- Job Control User's Guide (GC28-6703) and Job Control Language Reference (GC28-6704).

8.11 STEPLIB

The major function of the STEPLIB DD card is to define a private library for the duration of a job step. This DD statement can appear in any position among the DD statements for that step. Those parameters required to retrieve a data set are coded as in the JOBLIB card. A STEPLIB DD statement can appear in a cataloged procedure and can be referred to by or passed to other steps of the same job.

The JOBLIB DD statement need not appear in a job in order to use the STEPLIB DD statement. If both JOBLIB and STEPLIB DD statements appear in a job, the JOBLIB definition is ignored for any step that contains the STEPLIB definition. If the user wants the JOBLIB definition ignored but the step does not require use of another private library, the system library must be defined on the STEPLIB DD statement as shown in the following statement:

```
//STEPLIB      DD       DSN=SYS1.LINKLIB,DISP=SHR
```

As with ordinary DD statements, a sequence of DD statements may be concatenated with the STEPLIB card so that they are effectively read as one.

8.11.1 REFERENCES

See References, paragraph 8.10.1.

8.12 SYSLIB

The DD name, SYSLIB, defines the automatic call library of the Linkage Editor. SYSLIB contains libraries which are comprised of modules required by language processors and other processing systems. The SYSLIB libraries in the LINK and LINKGO procedures on the M&DO computers are shown in the following chart:

Model 95		Model 75	Model 65
LINK	LINKGO	LINK/LINKGO	LINK/LINKGO
SYS2.DUMMY	SYS2.DUMMY	SYS2.DUMMY	SYS2.DUMMY
SYS1.FORTLIB	SYS2.DUMMY	SYS2.GSFCLIB	SYS2.DUMMY
SYS2.GSFCLIB	SYS1.FORTLIB	SYS1.FORTLIB	SYS1.FORTLIB
SYS2.LOADLIB	SYS2.GSFCLIB	SYS1.PL1LIB	SYS1.PL1LIB
SYS1.PL1LIB	SYS1.PL1LIB		
SYS1.SSPAK	SYS1.TELCMLIB		
	SYS2.LOADLIB		
	SYS1.SSPAK		

The SYSLIB statement in these procedures concatenates the libraries which provide the routines required by a majority of users. Users requiring routines in a private library can concatenate that library to SYSLIB by coding:

```
//LINK.SYSLIB      DD       DSN=pvtlib,DISP=SHR,
//                          UNIT=2314,VOL=SER=xxxxxx
```

If the library is cataloged, the UNIT and VOL parameters are not required.

Note that SYSLIB in the procedures is coded with a DSN=SYS2.DUMMY to simplify the concatenation of a private library.

An example of the addition of a private library is the user who requests the SC4060 plot package. The routines for this package are stored in the library named SYS2.SC4060. This library is concatenated to SYSLIB by coding:

```
//LINK.SYSLIB       DD        DSN=SYS2.SC4060,DISP=SHR
```

8.12.1 REFERENCES

IBM System/360 Operating System manual.

- Linkage Editor and Loader (GC28-6538)

SECTION 9

UTILITIES

9.1 GENERAL

9.1.1 NATURE OF UTILITIES

Utility programs are written to perform common functions associated with the creation and maintenance of S/360 data sets. They are invoked by the EXEC card:

```
//stepname EXEC PGM=utilityname
```

and are told what to do through the use of control cards called utility control statements. These control cards can be used individually or in combination to perform a variety of operations such as the copying, moving, printing, punching, reblocking, updating, deleting, and cataloging of data sets, and the dumping, restoring, mapping and analyzing of direct-access storage devices. In some cases, the desired operation may be accomplished by more than one utility.

9.1.2 HOW TO CHOOSE A UTILITY

The IBM utility programs offer a wide range of functions which process data from the volume level to the record level. Those system utilities whose name begins with IEH can operate at the volume or data set level. Those data set utilities whose name begins with IEB process data at the data set level or below. This overlap may sometimes cause confusion in choosing a utility for a particular purpose.

When an IEH and an IEB utility can both perform a desired function, the user should use the IEB utility.

Three major factors in choosing a utility are the operation to be performed, the level of data to be processed, and the data set organization.

Table 9.1-1 may be used in selecting a utility by application. This table allows the user to find the proper utility by using the terms with which he is most familiar.

Table 9.1-1. How to Select a Utility

APPLICATION		9.2.1 – IEHMOVE	9.2.2 – IEHLIST	9.2.3 – IEHINITT	9.2.4 – IEHDASDR	9.2.6 – IEHPROGM	9.2.5 – IEFBR14	9.3.1 – IEBCOPY	9.3.2 – IEBGENER	9.3.3 – IEBPTPCH	9.3.4 – IEBUPDTE	9.3.5 – IEBDG	9.4.1 – MAPDISK	9.4.2 – PATRICK	9.4.3 – IEBFGR	9.4.4 – OSSLIP	9.4.5 – NEWMAN	9.4.6 – LMODMAP
LIST ENTRIES	IN A CATALOG		X															
	IN A VTOC		X									X						
	IN A DIRECTORY OF A PDS		X															
PRINT PRINT/PUNCH	CONTENTS OF A DIRECT-ACCESS VOLUME				X													
	CONTENTS OF A PDS OR MEMBER OF A PDS							X	X									
	OF A SEQUENTIAL DATA SET, MEMBER OF A PDS, INPUT CARD DECK							X	X				X					
CREATE A BACKUP COPY	OF A VOLUME				X													
	OF A SEQUENTIAL DATA SET	X						X					X					
	OF A PDS OR A MEMBER OF A PDS	X					X	X		X								
UPDATE	A PDS (AND/OR ALLOCATE SPACE)	X				X	X	X	X		X			X	X			
	A MEMBER OF A PDS WITH 80 BYTE LOGICAL RECORDS										X							
	A SEQUENTIAL DATA SET					X									X			
COMPRESS	A PDS	X					X											
	A SEQUENTIAL DATA SET							X										

Table 9.1-1. (Cont'd)

APPLICATION / SUBSECTION REFERENCE – UTILITY		9.2.1 – IEHMOVE	9.2.2 – IEHLIST	9.2.3 – IEHINITT	9.2.4 – IEHDASDR	9.2.6 – IEHPROGM	9.2.5 – IEFBR14	9.3.1 – IEBCOPY	9.3.2 – IEBGENER	9.3.3 – IEBPTPCH	9.3.4 – IEBUPDTE	9.3.5 – IEBDG	9.4.1 – MAPDISK	9.4.2 – PATRICK	9.4.3 – IEBFGR	9.4.4 – OSSLIP	9.4.5 – NEWMAN	9.4.6 – LMODMAP
EXPAND	A PDS	X						X	X							X		
	A SEQUENTIAL DATA SET								X		X					X		
CONVERT DATA SET ORGANI-ZATION	FROM SEQUENTIAL TO PARTITIONED								X		X					X		
	FROM PARTITIONED TO SEQUENTIAL								X		X							
CREATE A PDS	FROM SEQUENTIAL INPUT								X		X							
	BY MERGING TWO OR MORE PDS'S	X						X										
CREATE FILES	OF TEST DATA									X								
	BY EDITING EXISTING FILES									X								
TAPE TESTING													X					
LABEL	A 7-TRACK OR 9-TRACK MAGNETIC TAPE			X														
INITIALIZE	A DASD				X													
MISCELLA-NEOUS	BUILD OR DELETE AN INDEX OR INDEX ALIAS					X												
	BUILD AND MAINTAIN A GENERATION DATA GROUP INDEX					X												
	CONNECT OR RELEASE TWO VOLUMES					X												
MAP A LOAD MODULE																		X

9.1.3 UTILITY CATEGORIES

Utility programs fall into two broad categories:

a. IBM-supplied utilities

- System utilities (IEHMOVE, IEHLIST, IEHINITT, IEHDASDR, IEHPROGM)
- Data Set utilities (IEBGENER, IEBCOPY, IEBPTPCH, IEBUPDTE, IEBDG)

b. Other utilities (PATRICK, MAPDISK, IEFBR14, IEBFGR, OSSLIP)

The IBM utilities contain system utilities, which are used to maintain volumes of data at an organizational level, and data set utilities, which are used to process data at the data set or record level. The other utilities include four user-written programs and a module of the S/360 Operating System (IEFBR14). IEFBR14 is not generally classified as a utility, but is used to perform data set allocation and disposition utility functions.

The IBM-supplied utilities are all documented in IBM System/360 Operating System Utilities (Form GC28-6586). Documentation is generally available for user-written utilities and may be obtained by calling Mrs. Pat Barnes (extension 6796) in the GSFC Program Library, Building 3, Room 133.

9.1.4 UTILITY CONTROL STATEMENTS

The four fields of utility control statements are: Name, Operation, Operand, and Comment. The Name field begins in column 1 (except IEBUPDTE) and is followed by a space. Each of the other fields must be preceded and followed by a space, as shown in the following general format:

```
Name      Operation      Operand(s)      Comments
```

The IEBUPDTE control statements must start with a ./ in columns 1-2 and the Name (if any) begins in column 3. The remainder of the statement is the same as the described format.

Depending upon the utility used, the control statements identify the function to be performed, the specific volume or data set to be processed, and any parameters which modify the operation.

The name field begins in column 1 (column 3 for IEBUPDTE) and conforms to the usual rules for S/360 names, i.e., it can be from one to eight alphameric characters, the first of which must be alphabetic. When the name field is not used, column 1 of the control card (column 3 for IEBUPDTE) must be blank. The name field is required only in the IEHINITT utility. It is generally not used in the other utilities, but may be used if desired.

The operation field identifies the type of control statement. It may specify an operation to be performed, or may provide information which further defines the extent of the operation. This field begins before column 17 and must be preceded and followed by at least one blank. If the name field is not used, the operation field may begin in column 2. It is helpful to either right align or left align all operation fields to improve readability. Indenting an operation field may be used to indicate a reference to the preceding operation.

The operand field consists of keyword parameters separated by commas. The keywords are defined in the utility program and contain information such as volume and data set identification.

The comments field must be preceded by a blank and may contain any information.

Utility control statements are most often supplied in punched card form in the input stream, but may also be supplied as a sequential data set or as a member of a PDS.

The rules for continuation of utility control statements are as follows:

a. Interrupt the field after any comma before column 72.

b. Punch a non-blank character in column 72.

c. Continue in column 16 of the continuation card.

9.1.5 UTILITY PECULIARITIES

a. IEH utilities are unique in that the DD cards specify volume identification but do not reference the data set name, i.e., they do not contain the DSNAME parameter.

b. Derived names for devices should not be used in utility control statements.

For DISK use:	VOL=2314=serial number
but not:	VOL=DISK=serial number
For TAPE use:	VOL=2400=serial number
but not:	VOL=9TRACK=serial number

Derived names may be used in utility DD statements.

c. In the IEH utilities, space allocation for moved data sets is provided by the utility program and does not need to be coded in the JCL. Space may also be preallocated by the utility IEHPROGM or IEFBR14, or on a DD card in the program being executed.

9.1.6 NOTES ON EXAMPLES

In the examples in Section 9, the JOB card is not shown, but the user must supply one when executing his program. In some examples, additional steps are shown to illustrate the relationship between the utility and other steps within the job.

The use of the delimiter /* is optional when the input stream is defined by:

```
//SYSIN  DD  *
```

It does serve the purpose of visually identifying the end of an input data set.

The delimiter is required when the input stream is defined by:

```
//SYSIN  DD  DATA
```

The input data must not include a /* card because this card signifies the last card in the input stream.

9.1.7 RETURN CODES

The term RETURN CODE is the name of the condition codes returned by S/360 OS utilities. The return code indicates the level of success when executing an OS utility and may be tested using the COND execution parameter of the following step. The return codes vary in increments of 4 from 00 for successful completion to 16 for the most severe errors. The exact meaning for each value varies for each utility.

9.2 SYSTEM UTILITIES

9.2.1 IEHMOVE

The IEHMOVE system utility moves or copies logical groups (volumes, data sets, catalogs) of S/360 data.

Data to be moved may be:

a. Sequential or partitioned data sets.

b. Resident on one or more volumes (up to 5).

c. Cataloged or uncataloged.

d. A catalog or portions of a catalog.

e. A BDAM data set containing variable length spanned (VS or VBS) records.

Members of moved or copied data sets may be merged, renamed, replaced, and selectively included or excluded.

9.2.1.1 MOVE versus COPY

A MOVE operation differs basically from a copy operation in that upon successful completion, MOVE scratches the source data set from direct-access volumes, whereas COPY leaves the source data intact. Also, MOVE updates the catalog entry for cataloged data sets to point to the MOVED data set, whereas COPY does not change the catalog entry.

If for some reason the MOVE operation cannot be successfully completed, the source data set is not scratched. Also, if space has been allocated by the IEHMOVE program, all data being moved or copied is scratched from the receiving volume.

If space has been previously allocated for the new data set (by this or a previous job), no data that has been MOVED or COPIED is scratched. Utility messages tell the user which data sets or members have been moved and the type of error which occurred.

If an ABEND occurs during execution of IEHMOVE, IEHMOVE is not able to complete the aforementioned housekeeping functions. The partially built new data set remains on the receiving volume. Before resubmitting the IEHMOVE step, one must scratch the partially built new data set from the receiving volume; otherwise, IEHMOVE will treat the operation as a merge of previously allocated data sets. This is not desirable, since it is possible that some of the alias names will not be updated.

For example, if the IEHMOVE program has moved 104 members of a 105-member partitioned data set, and on moving the 105th member, an I/O error is encountered, then:

- If space was allocated by the IEHMOVE program, the entire partitioned data set is scratched from the receiving volume.
- If space was previously allocated, no data is scratched from the receiving volume. In this case, after determining the nature of the error, the user need move only the 105th member into the receiving partitioned data set.

The following table illustrates catalog maintenance by MOVE and COPY.

FOR CATALOGED DATA SETS		
OPERANDS	MOVE	COPY
FROM not used UNCATLG not used	Catalog updated	Catalog not changed
FROM not used UNCATLG used	Catalog entry deleted	Catalog entry deleted
FROM used UNCATLG not used	Catalog not changed	Catalog not changed
CATLG used	(Not applicable)	Catalog updated

FOR UNCATALOGED DATA SETS		
OPERANDS	MOVE	COPY
FROM used CATLG not used	Catalog not changed	Catalog not changed
CATLG used	(Not applicable)	Catalog updated

9.2.1.2 Example - MOVE

An example of moving a PDS from disk to disk with space allocated by the IEHMOVE program is shown below:

```
//                EXEC        PGM=IEHMOVE
//SYSPRINT        DD          SYSOUT=A
//SYSUT1          DD          UNIT=2314,VOL=SER=G1SCR1,DISP=SHR
//DISKNEW         DD          UNIT=2314,VOL=SER=xxxxxx,DISP=SHR
//DISKOLD         DD          UNIT=2314,VOL=SER=yyyyyy,DISP=SHR
//SYSIN           DD          *
   MOVE                       PDS=sourcpds,TO=2314=xxxxxx,FROM=2314=yyyyyy
```

9.2.1.3 Example - COPY (Sequential Data)

An example of copying a sequential data set from labeled tape to disk, with space allocation provided by the IEHMOVE program, is shown below:

```
//STEP1           EXEC        PGM=IEHMOVE
//SYSPRINT        DD          SYSOUT=A
//SYSUT1          DD          UNIT=2314,DISP=SHR,VOL=SER=G1SCR1
//TAPEIN          DD          UNIT=2400,DISP=OLD,VOL=SER=xxxxxx
//DISKOUT         DD          UNIT=2314,DISP=SHR,VOL=SER=G1SCR2
//SYSIN           DD          *
   COPY                       DSNAME=seqset,TO=2314=G1SCR2,FROM=2400=xxxxxx
```

Where xxxxxx would, in this case, be replaced by the user's 9-track tape serial number. G1SCR1 and G1SCR2 are 'scratch packs' on the 360/95.

9.2.1.4 Example - COPY (Multi-volume Sequential Data)

An example of copying a multi-volume sequential data set from three labeled 9-track tapes to one disk is shown below:

```
//MULTI           EXEC        PGM=IEHMOVE
//SYSPRINT        DD          SYSOUT=A
//SYSUT1          DD          UNIT=2314,VOL=SER=G1SCR1,DISP=SHR
//TAPE            DD          DISP=(OLD,KEEP),UNIT=(2400-9,,DEFER),
//                            VOL=SER=(xxxxxx,yyyyyy,zzzzzz)
//DISK            DD          UNIT=2314,DISP=SHR,VOL=SER=G1SCR3
   COPY                       DSNAME=seqdata,TO=2314=G1SCR3,          X
                              FROM=2400=(xxxxxx,3,yyyyyy,1,zzzzzz,1)
```

The format on the FROM operand is:

```
FROM=(tapeno,seqno,...)
```

where the tape number and file sequence number are listed for each tape on which the file resides. In this example, the data set is the third file on tape xxxxxx, and the first file on tapes yyyyyy and zzzzzz.

Notes:

a. Since the LABEL parameter is omitted on the TAPE DD card, the system default LABEL=(1,SL) is assumed.

b. When using unlabeled tapes, the FROMDD keyword would have to be included on the COPY DSNAME statement, i.e., (FROMDD=TAPE), to make DCB and LABEL information available. The DCB parameter is mandatory only when dealing with UNLOADED data sets (see paragraph 9.2.1.6) and must be of the form (RECFM=FB, LRECL=80, BLKSIZE=800).

9.2.1.5 Example - COPY (Multiple Data Sets)

An example of copying multiple data sets from one tape to one or more disk volumes is shown below:

```
//MULTDAT       EXEC    PGM=IEHMOVE
//SYSPRINT      DD      SYSOUT=A
//TAPEIN        DD      UNIT=2400,VOL=SER=xxxxxx,DISP=OLD,
//                      LABEL=(1,BLP)
//DISKOUT       DD      UNIT=2314,VOL=SER=yyyyyy,DISP=OLD
//SYSUT1        DD      UNIT=2314,VOL=SER=zzzzzz,DISP=OLD
//SYSIN         DD      *
   COPY                 DSNAME=seqset1,TO=2314=yyyyyy,          X
                        FROM=(xxxxxx,3),FROMDD=TAPEIN
   COPY                 DSNAME=seqset2,TO=2314=yyyyyy,          X
                        FROM=(xxxxxx,4),FROMDD=TAPEIN
   COPY                 DSNAME=seqset3,TO=2314=yyyyyy,          X
                        FROM=(xxxxxx,7),FROMDD=TAPEIN
```

In this example, three sequential data sets are copied from an unlabeled 9-track tape to a 2314 disk volume. One copy statement is required for each data set to be copied. The format of the FROM operand is:

FROM=(tapeno,seqno)

Note that since the data sets are not unloaded, no DCB information need appear on the TAPEIN DD card.

See paragraph 9.1.4.1 for the rules for continuation of utility control statements.

9.2.1.6 Unload/Load

Because magnetic tape does not support partitioned data sets in their true form, the program reorganizes the data and puts them on the specified device as an 'unloaded' data set. This is a sequential data set consisting of 80-byte blocked records that contain the source data and control information for subsequently reconstructing the source data in their true form. When an unloaded data set is moved by IEHMOVE to a device that will support the data in their true form, the data are automatically reconstructed.

Unloaded data sets are a convenient way to maintain a backup or transportable copy of partitioned data sets, especially in the GSFC environment where disk space is at a premium. Any PDS may be unloaded.

The following example (using the output from a previous compile step) illustrates (1) the creation of a PDS containing a load module and (2) the unloading of the load module, using IEHMOVE. The load module is a member of the data set, LODMOD; its name is GSFC. Because IEHMOVE requires a specific volume and dsname for the input data set, LODMOD is placed on an arbitrarily chosen scratch pack, G1SCR3.

```
//CREATE          EXEC        LINK
//LINK.SYSLMOD    DD          VOL=SER=G1SCR3,DSN=LODMOD(GSFC)
//UNLOAD          EXEC        PGM=IEHMOVE
//SYSPRINT        DD          SYSOUT=A
//DISKIN          DD          UNIT=2314,VOL=SER=G1SCR3,DISP=OLD
//TAPEOUT         DD          UNIT=2400,VOL=SER=xxxxxx,DISP=(NEW,KEEP),
//                            DCB=(DEN=3,RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSUT1          DD          UNIT=2314,VOL=SER=G1SCR1,DISP=OLD
//SYSIN           DD          *
   COPY                       PDS=LODMOD,TO=2400=xxxxxx,FROM=2314=G1SCR3
```

The omission of the label parameter requires that a previously labeled tape be used for the output data set. It is always wise to use a labeled tape.

At a later time, the unloaded data set could be reloaded by IEHMOVE and then executed as shown in the following example. Note that the disposition of the DISKOUT data set is DISP=OLD, because the space will be allocated by the IEHMOVE program.

```
//RELOAD        EXEC    PGM=IEHMOVE
//SYSPRINT      DD      SYSOUT=A
//TAPEIN        DD      UNIT=2400,VOL=SER=xxxxxx,DISP=(OLD,KEEP)
//DISKOUT       DD      UNIT=2314,VOL=SER=G1SCR7,DISP=OLD
//SYSUT1        DD      UNIT=2314,VOL=SER=G1SCR6,DISP=OLD
//SYSIN         DD      *
   COPY                 PDS=LODMOD,TO=2314=G1SCR7,FROM=2400=xxxxxx
//GO            EXEC    PGM=GSFC,REGION=xxxx  (as required)
//STEPLIB       DD      DSN=LODMOD,UNIT=2314,VOL=SER=G1SCR7,
//                      DISP=(OLD,DELETE,KEEP)
//SYSPRINT      DD      SYSOUT=A
//SYSUDUMP      DD      SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,
//                      BLKSIZE=7265),SPACE=(CYL,1)
//FT06F001      DD      SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,
//                      BLKSIZE=7625),SPACE=(CYL,(1,1))
//FT05001       DD      *
```

(Data for program being executed)

The preceding examples illustrate creating, unloading, reloading, and executing a load module. Other partitioned data sets, such as source libraries and object module libraries, could be unloaded and then reloaded when needed. Once reloaded, the PDS could be updated, compiled, changed, or processed, just as any other PDS.

A disposition DISP=(OLD,DELETE,KEEP) was specified on the STEPLIB DD card so that if the step executed successfully, the data set LODMOD would be DELETED. If the step did not execute successfully, the PDS LODMOD would remain intact on the volume. The error condition could be corrected and the step rerun. This would save the time involved in unloading the PDS from tape to disk again. One should note, however, that the scratch packs are "cleaned" at the start of every day and one cannot depend upon the unauthorized data sets remaining intact from day to day.

9.2.2 IEHLIST

IEHLIST can be used to list the names of entries in a catalog, the names in a directory of a partitioned data set, or a volume table of contents (VTOC). This is particularly useful in listing the system catalog, the member names of the procedure library, or in determining what data sets are on a particular disk volume.

9.2.2.1 System Catalog Listing

The example below may be followed when listing the system catalog on any one of the M&DO computers. The listing includes the fully qualified name of each applicable cataloged data set and the serial number of the volume on which it resides.

```
//LISTCLG          EXEC          PGM=IEHLIST
//SYSPRINT         DD            SYSOUT=A
//NUCLEUS          DD            VOL=REF=SYS1.NUCLEUS,DISP=SHR
//SYSIN            DD            *
   LISTCTLG
```

Note: The volume serial number could be stated explicitly as VOL=SER=GlSYS1; this would restrict the example to the model 95 only.

By omitting the VOL=device=serial field on the LISTCTLG card, the catalog is assumed to reside on the system residence.

To list a catalog other than the system catalog, the DD card must refer to the volume on which the catalog is located; the volume would be specified in the utility control card as shown below for a disk.

```
//LIST             EXEC          PGM=IEHLIST
//SYSPRINT         DD            SYSOUT=A
//CATALOG          DD            VOL=SER=xxxxxx,DISP=SHR,UNIT=2314
//SYSIN            DD            *
   LISTCTLG                      VOL=2314=xxxxxx
```

9.2.2.2 Member Name Listing

To list the member names of either a private or system PDS, the example below may be followed.

```
//LIST             EXEC          PGM=IEHLIST
//SYSPRINT         DD            SYSOUT=A
//DD1              DD            UNIT=2314,VOL=SER=xxxxxx,DISP=SHR
//SYSIN            DD            *
   LISTPDS                       DSNAME=name,VOL=2314=xxxxxx
   LISTPDS                       DSNAME=name,VOL=2314=xxxxxx,FORMAT
```

The FORMAT option causes the fields of the directory entries in a PDS to be edited and formatted. The formatted listing may only be specified for a PDS whose members have been created by the linkage editor.

Up to 10 partitioned data sets may be listed by one execution of this utility. One control statement is required for each PDS listed. The PDS's may reside on several volumes; one DD card is required for each volume referenced.

9.2.2.3 Volume and Data Set Status

The user must constantly be aware of the status of the volumes and data sets with which he is working. This is particularly true of the special systems where many tasks may be using the same programs and data.

LISTVTOC provides the user with a choice of:

a. A formatted listing (FORMAT option) of the VTOC, giving an in depth description of the data sets residing on the volume.

b. An abbreviated, edited listing (default) of the VTOC.

c. A hexadecimal dump (DUMP option) listing of the DSCB's in the VTOC.

For general use, the FORMAT option is recommended.

The following example lists the VTOC on the four volumes designated; each control card is explained below.

```
//LISTVTOC          EXEC          PGM=IEHLIST
//SYSPRINT          DD            SYSOUT=A
//VTOC1             DD            UNIT=2314,VOL=SER=G1SYS1,DISP=SHR
//VTOC2             DD            UNIT=2314,VOL=SER=xxxxxx,DISP=SHR
//VTOC3             DD            UNIT=2314,VOL=SER=yyyyyy,DISP=SHR
//VTOC4             DD            UNIT=2314,VOL=SER=zzzzzz,DISP=SHR
//SYSIN             DD            *
   LISTVTOC
   LISTVTOC                       VOL=2314=xxxxxx
   LISTVTOC                       FORMAT,VOL=2314=yyyyyy
   LISTVTOC                       DUMP,VOL=2314=zzzzzz
```

Notes:

a. Each volume referenced by a control statement must have a corresponding DD card.

b. The default volume in the first control statement is the system's residence device.

The four control statements, taken in order, specify:

a. The abbreviated, edited format listing of the system's residence volume (default).

b. The abbreviated, edited format of the VTOC for volume xxxxxx.

c. The formatted listing of the VTOC for volume yyyyyy.

d. The hexadecimal dump of the VTOC for volume zzzzzz.

9.2.2.4 VTOC Data Set Entries

The user is often interested in knowing if a particular data set is on a specified volume, and the actual space occupied by the data set on that volume.

The following example will list the VTOC entry for the data set if it is present on the given volume.

```
//LISTVTOC          DD          PGM=IEHLIST
//SYSPRINT          DD          SYSOUT=A
//VTOC1             DD          UNIT=2314,VOL=SER=xxxxxx,DISP=SHR
//SYSIN             DD          *
   LISTVTOC                     FORMAT,VOL=2314=xxxxxx,DSNAME=name
```

9.2.3 IEHINITT

This utility program places standard volume labels on 7- or 9-track magnetic tapes. A STANDARD LABEL (SL) is one which has a format acceptable to the S/360 Operating System, and contains the user-specified volume serial number and owner name. GSFC users should use their programmer ID in place of owner name. In addition to the standard label, a dummy header record, containing the characters HDR1 and 76 EBCDIC zeros, is created, followed by a tape mark.

IEHINITT has the following capabilities:

a. Any number of 7- and 9-track tapes may be initialized in one execution.

b. One or more tape drives may be used.

c. Serial numbers may be numeric or alphanumeric, but cannot contain blanks, commas, apostrophes, equal signs, or special characters.

d. Each control statement must have a corresponding DD statement with the same name.

e. If serial numbers are to be incremented by the utility, they must be numeric.

f. After being labeled, the tapes may be rewound and unloaded, or rewound to load point.

An example of IEHINITT used to label two 9-track tapes follows below.

```
//LABELIT          EXEC          PGM=IEHINITT
//SYSPRINT         DD            SYSOUT=A
//MYTAPES          DD            DCB=(DEN=3),UNIT=(2400,1,DEFER)
//*                              Any ddname can be used.  A 1600 BPI,
//*                              9-track unit is assigned.
//*                              Deferred mounting must be requested.
//*                              As many similar tapes as desired may be
//*                              labeled, with one DD card.
//SYSIN            DD            *
MYTAPES          INIT          OWNER='ZCSAA',SER=1252H
MYTAPES          INIT          OWNER='ZCSAA',SER=1782G
```

The name on the INIT card (in this example, MYTAPES) must begin in column 1 and agree with the ddname of the DD card to which it refers.

It is impossible to guarantee that a tape may not be accidentally written over. It may erroneously be mounted by the operator or erroneously requested by someone else. However, the use of labeled tapes will insure that the correct tape is mounted when the labeled tape user requests it. This could avoid a situation in which the user's program processes data from an improperly mounted tape, or possibly from his own tape, which has been overwritten by someone else.

When labeled tapes are used to receive data, additional identifying information is written onto the label. The HDR1 record is filled with operating system and device-dependent data, and an HDR2 record, containing data set characteristics, overwrites the tape mark. A tapemark is written after the HDR2 record.

Because DCB information is written on labeled tapes when data are received, the DCB parameter is not required in the JCL when the tape is subsequently used as input.

Users should not attempt to use IEHINITT to label a tape which currently contains data because the first file will be lost.

Additional information may be found in the following manuals: IBM System/360 Operating System: Tape Labels (GC28-6680), and IBM System/360 Operating System: Utilities (GC28-6586).

9.2.4 IEHDASDR

The IEHDASDR utility is the primary maintenance program for direct-access volumes used under the IBM S/360 Operating System.

This utility can:

a. Dump and restore the contents or portions of a direct-access volume to one or more volumes of the same device type, magnetic tape, or to the system output device.

b. Label and analyze a direct-access volume for defective tracks, and assign alternate tracks for those found defective.

c. Assign alternate tracks for specified defective or questionable tracks without analyzing the tracks.

d. Change the label on a direct-access volume (on-line and off-line capability).

e. Perform track formatting functions without analyzing the tracks.

f. "QUICK DASDI" - bypasses all surface analysis and track formatting, writing only a VTOC, record zero, home address and volume label.

g. Dump a bad track to a printer.

Caution: Misuse of IEHDASDR can destroy an entire volume of data. This program should be used only under the direction of a system programmer.

For further information, refer to the IBM System/360 Operating System: Utilities Manual (GC28-6586).

9.2.5 IEFBR14

IEFBR14 is a program which is used for space allocation and dispostion handling of data sets and to check JCL for syntax errors.

All that is required is to use the EXEC statement:

```
//STEP1         EXEC      PGM=IEFBR14
```

followed by the JCL to the processed, such as:

```
//OUTPUT        DD        DSN=name,VOL=SER=xxxxxx,UNIT=2314,
//                        DCB=(RECFM=FB,LRECL=80,
//                        BLKSIZE=3200),DISP=(NEW,PASS)
//                        SPACE=(CYL,(10,5))
```

Any data sets allocated in this manner will have a disposition of DISP=OLD when used in later job steps.

9.2.6 IEHPROGM

The IEHPROGM provides several directory, VTOC, and catalog maintenance functions. It can be used to scratch, rename, catalog, or uncatalog a data set; and to scratch or rename a member of a PDS. These functions will be discussed in succeeding paragraphs. It is important to note that a DD statement must be included for each volume that will be referred to in the IEHPROGM job step.

The following IEHPROGM capabilities will not be discussed in this document. Users requiring these operations should contact the PAC in Building 3.

a. Building or deleting an index or an index alias.

b. Building and maintaining a generation data group index.

c. Connecting or releasing two volumes.

9.2.6.1 IEHPROGM versus IEFBR14

Both IEHPROGM and IEFBR14 can allocate, delete, catalog, and uncatalog a data set. Except for allocation, the IEHPROGM functions are explicitly stated in the utility control cards, while IEFBR14 relies on the DD statement parameters and the OS data management routines to perform the required allocation or disposition of data sets.

Although both utilities can be used to perform catalog functions, IEHPROGM has the following capabilities not available to DISP=(,CATLG), which would be used by IEFBR14.

a. The CATLG operation automatically creates lower level indexes when required.

   The UNCATLG operation may be used to delete entries from the lower level indexes of the catalog. (Note that when speaking of index levels, the highest level of index in A.B.C.D would be A, the lowest D.)

b. The CATLG operation can create entries in catalogs other than the system catalog. The UNCATLG operation can delete entries from catalogs other than the system catalog.

There are also some differences in execution which are discussed in the following recommendations:

a. Users should use IEFBR14 to allocate data sets. Both utilities depend on the DD statement for allocation, but IEFBR14 is more efficient and less likely to have conflicts with utility control cards.

b. For the scratch, catalog, and uncatalog functions, IEFBR14 is easier to use if the data set is known to still exist and its status does not conflict with the requested disposition. If a conflict does occur, it will cause an abnormal termination. IEHPROGM should be used if the volume is known and the user does not know the actual status of the data set. This utility will print a message describing the error condition, and continue to the next step. However, the bypassed step will get a condition code of 8.

9.2.6.2 SCRATCH Operation

The SCRATCH operation is used to scratch a data set or member. To scratch a data set with an unexpired expiration date, the PURGE operand should be used. The user must supply the DSNAME and VOL operands to scratch a data set; to scratch a member, the MEMBER operand must also be used. The MEMBER operand should be punched before the DSNAME operand. In this manner, if a mistake is made in continuing a card, the user will not inadvertently scratch the library while attempting to scratch a member.

SCRATCH does not automatically uncatalog a scratched data set.

The VTOC and SYS operands should not be used, except by those familiar with their operation. Improper use of these operands could wipe out a disk.

9.2.6.3 RENAME Operation

The RENAME operation is used to change the name or alias of a data set or member residing on a direct-access device. The user must supply the old data set name (DSNAME), the volume on which it resides (VOL), and the new name to be assigned (NEWNAME). If a member is being renamed, the old member name (MEMBER) must also be supplied.

RENAME does not automatically update the catalog.

9.2.6.4 CATLG Operation

The CATLG operation is used to catalog a data set. It can be used with a RENAME operation to catalog a data set under its new name. The user must supply the data set name (DSNAME) and the volume on which the data set resides (VOL).

The CATLG operation automatically creates lower-indexes, if required.

9.2.6.5 UNCATLG Operation

The UNCATLG operation deletes (uncatalogs) an entry from the lowest level index of the catalog. UNCATLG can be used to uncatalog scratched data sets and the old name of a RENAMED data set. The user must supply the fully qualified name (DSNAME) of the data set to be uncataloged.

9.2.6.6 SCRATCH/UNCATLG Example

```
//DELETE        EXEC        PGM=IEHPROGM
//SYSPRINT      DD          SYSOUT=A
//SCRATCH       DD          UNIT=2314,VOL=SER=xxxxxx,DISP=OLD
//SYSIN         DD          *
   SCRATCH                  DSNAME=DODS.FUNC.OPS,VOL=2314=xxxxxx,PURGE
   UNCATLG                  DSNAME=DODS.FUNC.OPS
```

This example scratches the data set DODS.FUNC.OPS from the volume on which it resides. It does this by removing the Data Set Control Block (DSCB) for the data set from the Volume Table of Contents (VTOC) on the volume on which the data set resided. The UNCATLG operation removes OPS, the lowest level index in the structure DODS.FUNC.OPS from the catalog. Note that PURGE must be used when scratching a data set with an unexpired expiration date.

9.2.6.7 RENAME/CATLG Example

```
//RENAME        EXEC        PGM=IEHPROGM
//SYSPRINT      DD          SYSOUT=A
//DATSET        DD          UNIT=2314,VOL=SER=xxxxxx,DISP=OLD
//SYSIN         DD          *
   RENAME                   DSNAME=DODS.FUNC.OPS,VOL=2314=xxxxxx,     X
                            NEWNAME=DODS.FUNC.COM
   UNCATLG                  DSNAME=DODS.FUNC.OPS
   CATLG                    DSNAME=DODS.FUNC.COM,VOL=2314=xxxxxx
```

After the data set DODS.FUNC.OPS has been renamed, the old name must be removed from the catalog and the data set cataloged under its new name.

9.3 DATA SET UTILITIES

9.3.1 IEBCOPY

A new version of the IEBCOPY data set utility program was included under Release 19. The new version provides additional functions and a considerable performance improvement. The examples presented here have been written using the new version of IEBCOPY. For complete details concerning the new IEBCOPY refer to the manual, IBM System/360 Operating System: Utilities (GC28-6586-11).

IEBCOPY may be used only with partitioned data sets. IEBCOPY is used at the data set or member level to:

a. Copy an entire PDS. This will compress the data set by excluding the space formerlv occupied by members which had been deleted.

b. Expand a data set by merging new members.

c. Compress a data set by copying selected members into a new data set.

d. Compress a data set by copying inplace, freeing the space occupied by a scratched member. The members are copied consecutively, and all free space is placed at the end.

e. Recreate a data set by copying and specifiying a larger space allocation than that allocated for the original data set. This is useful when the data set has exhausted its primary, secondary, or directory allocations.

If an entire data set is copied, no control statements are required and the SYSIN card is coded: //SYSIN DD DUMMY.

If selected members are to be merged or copied, the COPY and MEMBER statements are required.

The COPY statement specifies whether the member names listed in the MEMBER statement will be included or excluded, and the maximum number of names that will be used.

The MEMBER statement lists the member names to be included or excluded.

9.3.1.1 Copying a PDS

To copy a PDS in its entirety, the following example below should be followed.

```
//COPY          EXEC        PGM=IEBCOPY
//SYSPRINT      DD          SYSOUT=A
//INOUT4        DD          DSN=oldpds,UNIT=2314,DISP=(OLD,DELETE,KEEP),
//                          VOL=SER=xxxxxx
//INOUT5        DD          DSN=newpds,UNIT=2314,DISP=(NEW,KEEP),
//                          VOL=SER=yyyyyy,
//                          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),
//                          SPACE=(TRK,(20,5,4))
//SYSUT3        DD          DSN=TEMP1,UNIT=2314,VOL=SER=vvvvvv,
//                          DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSUT4        DD          DSN=TEMP2,UNIT=2314,VOL=SER=wwwwww,
//                          DISP=(NEW,DELETE),SPACE=(TRK,(1))
//SYSIN         DD          *
//COPYIT        COPY        OUTDD=INOUT5,INDD=INOUT4
```

Notes:

a. The old data set (oldpds) has been copied and given a new name (newpds).

b. If the volumes for INOUT4 and INOUT5 are the same, the dsnames must be different; if the volumes are different, the dsnames may be the same or different.

c. The SYSUT3 and SYSUT4 DD cards define data sets to be used if more space is required for the input PDS's directory entries and the output PDS's directory blocks respectively. Volumes vvvvvv and wwwwww represent volumes used for scratch (temporary) data sets (G1SCR1 through G1SCRA on the 360/95).

d. The input data set is scratched at the end of the job. DISP=(OLD,DELETE,KEEP) should be used so that if the step terminates abnormally, the old (input data set) will not be destroyed.

IEBCOPY expands an existing data set by merging selected members from another data set.

```
//COPY          EXEC        PGM=IEBCOPY
//SYSPRINT      DD          SYSOUT=A
//INOUT1        DD          DSN=OLDPDS,DISP=(OLD,KEEP),UNIT=2314,
//                          VOL=SER=xxxxxx
//INOUT2        DD          DSN=EXPANDIT,DISP=(OLD,KEEP),UNIT=2314,
//                          VOL=SER=xxxxxx
//SYSIN         DD          *
COPYIT          COPY        OUTDD=INOUT2
                            INDD=INOUT1
                SELECT      MEMBER=G,S,F,C
```

Notes:

a. Only members G,S,F,C, of the input data set (OLDPDS) will be copied onto the output data set (EXPANDIT).

b. Although the old and new data sets should have the same DCB characteristics it is not mandatory under Release 19 for fixed length or variable length records, which can be reblocked or deblocked within the program. However, undefined records (RECFM=U), keyed records or track overflow records cannot be reblocked or deblocked and care should be exercised in their use.

c. The two data sets are on the same volume but could be placed on different volumes.

9.3.1.2 Compress Inplace

Compress inplace frees the space occupied by scratched data sets.

```
//COMPRESS          EXEC      PGM=IEBCOPY
//SYSPRINT          DD        SYSOUT=A
//INOUT2            DD        DSN=OURPDS,DISP=OLD,UNIT=2314,VOL=SER=xxxxxx
//SYSUT3            DD        DSN=TEMP1,DISP=NEW,UNIT=2314,VOL=SER=yyyyyy,
//                            SPACE=(TRK,(1))
//SYSUT4            DD        DSN=TEMP2,DISP=NEW,UNIT=2314,VOL=SER=zzzzzz,
//                            SPACE=(TRK,(1))
//SYSIN             DD        *
COPYIT              COPY      OUTDD=INOUT2,INDD=INOUT2
```

Notes:

a. The INOUT2 DD card defines the PDS to be compressed.

b. The SYSUT3 and SYSUT4 DD cards define data sets to be used if more space is required for the input PDS's directory entries and the output PDS's directory blocks respectively. Volumes yyyyyy and zzzzzz represent volumes used for scratch (temporary) data sets (G1SCR1 through G1SCRA on the 360/95).

c. The data set will be compressed in place as the OUTDD and INDD operands both refer to the same DD card, INOUT2.

d. Compress inplace cannot be used for data sets using the track overflow feature.

e. Compress inplace should not be used on system data sets.

Caution: The user should create a backup copy before performing a compress inplace.

If a job aborts while performing a compress inplace, the entire data set will be destroyed. Aborted runs can be caused by I/O errors, machine malfunctions, the machine going down, etc. To be safe, a backup copy should be created.

9.3.2 IEBGENER

IEBGENER is the primary utility used to modify the data set organization and format of sequential and partitioned data sets. Its major uses are:

a. Copying sequential input from card, tape, or disk to either printer, punch, disk, or tape, with or without editing.

b. Creating or expanding a PDS from either a sequential data set or a partitioned member used as input.

c. Creating sequential output from a member of a PDS used as input.

d. Converting data format to packed decimal, unpacked decimal, BCD, or EBCDIC.

e. Changing the blocksize or logical record length of a data set.

f. Rearranging, omitting, or changing specified data fields.

9.3.2.1 Copy Sequential Data Sets

IEBGENER is frequently used to print or punch sequential data sets consisting of either data, source programs, JCL, or a combination of both. The following example should be used to punch the contents of a sequential data set:

```
//DUPE          EXEC        PGM=IEBGENER
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          DSN=name,VOL=SER=xxxxxx,UNIT=2314,DISP=OLD
//SYSUT2        DD          SYSOUT=B,SPACE=(CYL,(1,1)),DCB=(LRECL=80,
//                          BLKSIZE=3200,RECFM=FB)
//SYSIN         DD          DUMMY
```

Notes:

a. A comment should be placed on the job submission slip, informing the operator that punch output is expected.

b. Through proper specifications on the SYSUT2 card, this example could be used to copy a sequential data set to printer, tape, or DASD devices.

c. By changing the SYSUT1 card, the sequential input could also reside on magnetic tape.

9.3.2.2 PDS Member Copying

To copy a member of a PDS as a sequential data set to a standard labeled tape, the example below should be followed:

```
//UNLOAD        EXEC        PGM=IEBGENER
//SYSPRINT      DD          SYSOUT=A
//SYSUT1        DD          UNIT=2314,VOL=SER=xxxxxx,DISP=(OLD,KEEP),
//                          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),
//                          DSN=SRCLIB(MMBR)
//SYSUT2        DD          UNIT=9TRACK,VOL=SER=xxxxxx,DISP=(NEW,KEEP),
//                          DCB=*.SYSUT1,DSN=MMBR
//SYSIN         DD          DUMMY
```

In the preceding example, a member of a source library was copied to standard labeled 9-track tape.

The DCB=*.SYSUT1 parameter tells the system that SYSUT1 and SYSUT2 will have the same DCB characteristics. The user may refer to the manual, IBM System/360 Operating System: Utilities (GC28-6586), for further information on IEBGENER.

9.3.3 IEBPTPCH

The IEBPTPCH program prints or punches the contents of:

a. A partitioned or sequential data set in its entirety.

b. A member or members of a PDS.

c. Selected records from a PDS or sequential data set.

d. An edited version of a PDS or sequential data set.

e. A directory of a PDS.

This is particularly useful in printing members of a PROCLIB, private library, source libraries, or other collections of data.

9.3.3.1 IEBPTPCH versus IEHLIST

a. IEBPTPCH should be used to list the records within the data set or member.

b. IEHLIST should be used to list member names of a PDS.

9.3.3.2 Operations/Operands

The required operation is PRINT or PUNCH. The other operations are dependent upon the data set organization or optional requirements of the user.

All PRINT/PUNCH operands apply equally to both operations, except two operands which control the PRINT format, and two which control punch card sequence numbering. The other major difference is the default record size, which is 120 characters for PRINT and 80 characters for PUNCH. The operands for MEMBER, TITLE, RECORD, LABELS, and EXITS operations apply equally to PRINT or PUNCH.

The examples in the succeeding paragraphs require only minor changes, if any, to apply equally to PRINT or PUNCH operations.

9.3.3.3 Print a PDS

The records of each member of the PDS will be printed as separate groups.

```
//PRINT          DD          PGM=IEBPTPCH
//SYSPRINT       DD          SYSOUT=A
//SYSUT1         DD          DSN=dsname,DISP=SHR,UNIT=2314,VOL=SER=xxxxxx
//SYSUT2         DD          SYSOUT=A
//SYSIN          DD          *
   PRINT                     TYPORG=PO,MAXFLDS=3
   TITLE                     ITEM=('Model 95 DATA SET dsname',50)
   RECORD                    FIELD=(80,1,,21)
```

The SYSUT1 DD statement defines the input data set. The SYSUT2 DD statement defines the output (printed) data set.

In this example, the data set records are listed in groups of 80 characters, as specified by the RECORD operation. The MAXFLDS operand specifies that up to three FIELD statements can occur in the control data set. The FIELD statement shown will center the output on the page. The TITLE operation specifies a title record starting in print position 50. To print all records of a PDS, the MEMBER statement should not be used.

Note: TYPORG=PS, specifying an organized sequential input data set, is the system default when TYPORG is omitted.

9.3.3.4 Print PDS Records

To print the records of selected members of a PDS, the utility control cards listed below should be used with the JCL from example 9.3.3.3.

```
PRINT          TYPORG=PO,MAXFLDS=2,MAXNAME=2
MEMBER         NAME=membername1
  RECORD       FIELD=(80)
MEMBER         NAME=membername2
  RECORD       FIELD=(80)
```

MAXNAME refers to the maximum number of MEMBER statements that follow the PRINT statement.

If the RECORD statement is removed, the records will print in standard PRINT format, i.e., groups of eight characters, separated by two spaces.

9.3.3.5 Print Sequential Data Set Records

To print records from a sequential data set, the following control set should be used:

With editing defined by the RECORD statement:

```
PRINT          MAXFLDS=1
  RECORD       FIELD=(80)
```

Without editing (standard PRINT format):

```
PRINT
```

9.3.3.6 Record Punching

Records can be punched, assigning sequence numbers with a standard increment. This is useful in punching a deck from a source library and changing the sequence numbers of the statements.

```
//PUNCH          EXEC        PGM=IEBPTPCH
//SYSPRINT       DD          SYSOUT=A
//SYSUT1         DD          DSN=srclib,UNIT=2314,DISP=OLD,VOL=SER=xxxxxx,
//                           DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT2         DD          SYSOUT=B
//SYSIN          DD          *
   PUNCH                     TYPORG=PO,MAXFLDS=1,CDSEQ=00001000,
                             CDINCR=100,MAXNAME=1
   MEMBER                    NAME=srcdeck
   RECORD                    FIELD=(72)
```

This example punches the records of member SRCDECK of data set SRCLIB. Positions 1-72 in the output card are taken from the records in the data set; positions 73-80 are generated from the CDSEQ and CDINCR operands which start with the number 1000 and increment by 100 for each new card.

If a sequential data set is to be processed in the same manner, the following control cards should be used:

```
PUNCH          MAXFLDS=1,CDSEQ=00001000,CDINCR=100
RECORD         FIELD=(72)
```

9.3.4 IEBUPDTE

IEBUPDTE is used primarily to update source program libraries, procedure libraries, and other partitioned data sets with 80-byte logical records. This utility can be used to add copy and replace members of a PDS. It can create a PDS from sequential input, and can convert a PDS to sequential organization. Records within data sets or members can be added, deleted, replaced, or renumbered.

When a member is updated, the output may be (1) a new data set incorporating the specified changes, and leaving the source member unchanged; (2) an updated copy of the member written in the space originally allocated to the data set; or (3) a new data set written with the space actually occupied by the source member. In case (3), initiated by the UPDATE=INPLACE operand, IEBUPDTE permanently changes the source member and allows update only by replacement of existing records. However, if a new data set is created, it may reside on either the same or a different volume. If it resides on a different volume, it may have the same DSNAME; if it is on the same volume, it must have a different DSNAME.

Six types of utility control statements are provided to control the many capabilities of this program.

a. Four FUNCTION statements (ADD,REPL,CHANGE,REPRO) specify the general operation to be performed. Additional information, such as member name, organization, and sequence field columns, are provided by the operands.

b. Two DETAIL statements, NUMBER and DELETE, are used in conjunction with the FUNCTION statements for deleting and resequencing specified logical records. Beginning sequence numbers, increments, and inserted records can be specified by the operands.

c. A DATA statement is the actual data to be used as a replacement record, or the new data to be added to the existing data set. The input DATA statements must have a sequence number in the same positions as the existing data set, including leading zeroes, if punched, usually in columns 73-80. If the numbers are in columns other than 73-80, the length and relative position must be specified in a SEQFLD keyword within a preceding FUNCTION statement. Data statements with a corresponding sequence number in the existing data set will replace the existing data record; those with no corresponding number will be inserted following the last (if any) sequenced input card and/or according to the specifications on the NUMBER statement.

d. LABEL statements indicate specified records to be used as user labels.

e. ALIAS statements allow the user to retain and/or assign alias names to output data sets.

f. The ENDUP statement indicates the end of data for the SYSIN data set.

All utility statements except the DATA statements are identified by ./ in columns 1-2, and can have a name (optional), in columns 3-10. At least one blank must precede the operation field.

9.3.4.1 Updating a Member of a PDS

In the following example, a member of a PDS is updated by change cards and deletions. The copy is placed on a different volume with a different DSNAME.

For the sake of illustration, it is assumed that the old PDS is a source program library and that the project to be performed is to modify a member and compile, link, and go with the modified version. If the new version checks out properly, the old member will be replaced with the new version (see paragraph 9.3.4.2).

The data statements following the CHANGE control card will replace those statements which have corresponding sequence numbers in the input member; those with no corresponding sequence number will be inserted. The DELETE statement will delete all records with sequence numbers from 1540 through 1670.

```
//JOB card
//STEP1             EXEC        PGM=IEBUPDTE,PARM=MOD
//SYSUT1            DD          DSN=oldpds,DISP=(OLD,KEEP),UNIT=2314,
//                              VOL=SER=xxxxxx
//SYSUT2            DD          DSN=newpds,DISP=(OLD,PASS),UNIT=2314,
//                              VOL=SER=yyyyyy
//SYSPRINT          DD          SYSOUT=A
//SYSIN             DD          *
./   CHANGE                     NAME=oldmember,LIST=ALL

(Data statements with sequence number in columns 73-80)

./   DELETE                     SEQ1=1540,SEQ2=1670
//STEP2             EXEC        FORTRANG
//SOURCE.SYSIN      DD          DSN=newpds(oldmember),DISP=(OLD,KEEP)
//STEP3             EXEC        LINKGO
//GO.DATA5          DD          *

(Data for program being executed)
```

Notes:

a. The data set newpds on the SYSUT2 card existed previous to the execution of this job. If the SYSUT2 card had contained a new data set, space would have had to be allocated on the direct-access device receiving the data set, a DISP=(NEW,PASS) would have had to be specified, and the DCB parameters included. The DCB parameters should be the same as those specified on the SYSUT1 DD card for the input data set.

b. Only one member may be updated per step.

c. The old master (SYSUT1) is not changed.

9.3.4.2 Replacing a Member in a PDS

Assuming that the modified version from paragraph 9.3.4.1 has executed satisfactorily, the old member is now replaced with the new version of the same name.

```
//                EXEC    PGM=IEBUPDTE,PARM=MOD
//SYSPRINT        DD      SYSOUT=A
//SYSUT1          DD      DSN=newpds,DISP=(OLD,DELETE,KEEP),
//                        UNIT=2314,VOL=SER=yyyyyy
//SYSUT2          DD      DSN=oldpds,DISP=(OLD,KEEP),
//                        UNIT=2314,VOL=SER=xxxxxx
//SYSIN           DD      *
./ REPL                 NAME=oldmember
./ ALIAS                NAME=oldalias
./ ALIAS                NAME=newalias
```

The member name must be the same in both data sets to use the REPL function. Alias names must be updated by the ALIAS statement. The first ALIAS statement updates an old alias to point to the new member; the second ALIAS statement assigns a new or second alias to the new member.

9.3.4.3 Adding a New Member to a PDS

This example may be used to add a member to a source program library or to add a procedure to a procedure library. The new member is input as a sequential data set in the input stream.

```
//                EXEC    PGM=IEBUPDTE,PARM=MOD
//SYSPRINT        DD      SYSOUT=A
//SYSUT2          DD      DSN=oldpds,DISP=(OLD,KEEP),UNIT=2314,
//                        VOL=SER=xxxxxx
//SYSIN           DD      *
./   ADD                NAME=newmember,LIST=ALL
```

(DATA statements for newmembr; sequence numbers appear in columns 73-80)

The member is added after the last existing member, and the name is placed in its collating sequence in the directory.

9.3.4.4 Copy a Member of a Partitioned Data Set and Convert to Sequential Organization

A portable backup copy of a member can be made by copying to tape and converting to sequential organization. The source member remains unchanged. The example shown here is a copy and renumber only; DATA statements and the DELETE statement could be used to copy and modify the output.

```
//                  EXEC        PGM=IEBUPDTE
//SYSPRINT          DD          SYSOUT=A
//SYSUT1            DD          DSN=oldpds,UNIT=2314,VOL=SER=xxxxxx,
//                              DISP=(OLD,KEEP)
//SYSUT2            DD          DSN=newseq,UNIT=9TRACK,VOL=SER=yyyyyy,
//                              DISP=(NEW,KEEP),DCB=(RECFM=FB,LRECL=80,
//                              BLKSIZE=3200),LABEL=(1,BLP)
//SYSIN             DD          *
./   CHANGE                     NEW=PS,NAME=amembr,LIST=ALL
./   NUMBER                     SEQ1=ALL,NEW1=1000,INCR=100
```

This example:

a. Copies a member (amembr) of the partitioned data set (oldpds) to a 9-track unlabeled tape.

b. Converts the member to sequential organization (NEW=PS).

c. Renumbers the entire output data set with a starting number of 1000 and increments of 100.

d. Lists the output data set.

By changing the set of control cards, this example could be used to add, replace, or delete records or blocks of records. The following control cards illustrate these functions:

```
./   CHANGE                NEW=PS,NAME=amembr,LIST=ALL

     (DATA statements with sequence number in columns 73-80)

./   DELETE                SEQ1=1560,SEQ2=1600

     (DATA statements with sequence number in columns 73-80)
```

9.3.4.5 Update Inplace

UPDATE=INPLACE allows the user to update a member within the space it actually occupies in the data set on the direct-access device.

DATA statements can be replaced only; no adds or deletes are allowed. Renumbering and header label modification are the only other functions permitted.

```
//                  EXEC        PGM=IEBUPDTE,PARM=MOD
//SYSPRINT          DD          SYSOUT=A
//SYSUT1            DD          DSN=oldpds,DISP=OLD,
//                              UNIT=2314,VOL=SER=xxxxxx
//SYSIN             DD          *
./   CHANGE                     NAME=member,UPDATE=INPLACE
     DO 140    J=IBEGIN,LENGTH                          00016100
140  CONTINUE                                           00017500
160  IF(L.EQ.0) GO TO 130                               00017560
```

The FORTRAN statements numbered 16100, 17500, and 17560 are permanently replaced by the DATA statements in the input stream. The input DATA statements must have corresponding statements in the member to be updated. With the UPDATE=INPLACE operand, the SYSUT2 DD statement need not be coded. Only one member may be updated per job step.

9.3.5 IEBDG

Because of the great detail required to properly describe this utility, the following description will emphasize functional usage. The user is referred to the IBM System/360 Operating System: Utilities Manual, (C28-6586) for details and examples.

The IEBDG utility provides test data for use in debugging new programs and testing changes to old programs. It is important to use data which can be easily analyzed for correct results. For this reason, IEBDG provides a wide variety of IBM-supplied or user-supplied data formats.

Test data can be generated completely from utility control cards or from a combination of input data and control cards. Several input files may be defined, but only one output file is allowed per set of utility control statements. Input and output may consist of sequential, indexed sequential, or partitioned data sets.

In some production runs where several data sets are used concurrently, it may be more economical to combine the data fields being used into one data set. This may be accomplished with IEBDG by defining the existing data fields and creating records composed of only these fields.

9.3.5.1 Functional Concepts

Although the detailed operation may become quite complicated, the functional concept is easily understood. Control cards are used to provide the following functions:

a. DSD marks the start of a set of utility control cards, and names the input and output data sets.

b. FD names and defines the contents of all data fields being created by the program, and names and defines the location of all data fields being copied from input records.

c. CREATE defines the record format of each record type to be created, using the data field named in the FD statements.

d. REPEAT states the number of times that a group of records will be repetitively created.

e. END completes each set of control statements.

One set of control statements is required for each output file to be created. Execution of the program creates the specified number of records.

The most important concept of IEBDG is the capability to modify the data field for each record being created. These modifications are specified in the ACTION and INDEX/CYCLE/RANGE operands of the FD statement.

The user may select one of seven different actions by which the contents of each field defined by an FD statement may be modified each time the record is created. This allows the records in a group to vary instead of remaining fixed.

The INDEX/CYCLE/RANGE operands provide the ability to increment numeric fields by a specified value (INDEX) after a specified number of records have been created (CYCLE). Thus, each record or group of records may be assigned a unique identification field. The RANGE operand limits the value of the field being incremented.

9.3.5.2 Detailed Functions

Detailed functions are controlled by the FD, CREATE, and REPEAT statements.

a. The user defines the data fields to be used giving the data field name, format, length, starting location in the output record, and actions to be performed on the field. If the field is being extracted from an input data set, the DDNAME and input field location are also specified. One Field Definition (FD) statement is required for each field to be used.

b. One CREATE statement is used to define the contents of each record to be created. The CREATE statement initializes the output record with a fill character; places the input record, if any, left-justified in the output record; and places a CREATE statement picture in the output record in that order. The user may specify the number of records to be created by the CREATE statement. One CREATE statement is required for each record format.

c. The REPEAT statement allows the user to repetitively execute a group of CREATE statements.

The REPEAT/CREATE statements are analogous to the FORTRAN DO loop, where CREATE represents the inner DO and is executed in its entirety for each repetition specified by the REPEAT statement.

9.4 OTHER UTILITIES

9.4.1 MAPDISK

MAPDISK is a utility program supplied by GSFC to enable users to obtain precise information about the contents of any direct-access storage device (DASD):

This program will list the following information for every data set on the DASD:

- Data Set Name (DSNAME)
- Data Set Organization (DSORG)
- Record Format (RECFM)
- Logical Record Length (LRECL)
- Block Size (BLKSIZE)
- Number of extents used
- Beginning and ending track addresses
- Creation Dates
- Purge Dates
- File Types
- File Serial Numbers
- Volume Sequence Numbers
- Security Protections
- Total number of tracks allocated
- Total number of tracks used
- Number of Directory Blocks Allocated
- Number of Directory Blocks Used

In addition to this information about each file on the DASD, the MAPDISK program will list the amount of free space remaining on the volume and in the VTOC.

9.4.1.1 MAPDISK Procedure (Model 95 Only)

To execute MAPDISK on the model 95, only the Execute card

```
//MAP          EXEC      MAPDISK
```

is required. MAPDISK is a procedure in SYS1.PROCLIB which will provide the disk map for the following DASD:

- G1DRM1
- G1DRM2
- G1USR1
- G1USR2
- G1SYS1
- G1SYS2

In addition to these volumes, additional DD cards may be inserted after the Execute card to provide a map of any other DASD volume which is mounted. These DD cards must be of the form:

```
//ddname        DD        DISP=SHR,UNIT=xxxx,VOL=SER=ssssss
```

where:

- ddname=any name except DD1 through DD6; use of DD1 through DD6 will override the corresponding cards in the procedure
- xxxx=a valid unit type for a DASD
- ssssss=the volume serial number

9.4.1.2 MAPDISK Program (For Use on Model 95 and 75)

Because the MAPDISK procedure is not available on the model 75, the user must supply the following JCL cards:

```
//MAP            EXEC      PGM=MAPDISK
//SYSPRINT       DD        SYSOUT=A,SPACE=(CYL,(1,1))
//SYSUT1         DD        UNIT=DISK,SPACE=(TRK,(1,1))
//ddname         DD        UNIT=xxxx,VOL=SER=ssss,DISP=SHR
```

where:

- SYSUT1=the work data set
- ddname=any valid ddname
- xxxx=a valid unit type for a DASD
- ssss=the volume serial number of the volume to be mapped
- One DD card is required for each volume to be mapped.

This program should be used on the model 95 when one does not wish to map the volumes specified in the MAPDISK SYS1.PROCLIB procedure.

Note: When mapping a scratch disk, the user should not specify the same volume serial number for SYSUT1 and for the pack being mapped. This will cause an I/O error.

MAPDISK is limited in the number of free space fragments which it will accept; if this limit is exceeded; the results may include flagging all data sets as free space and adding large constant values to track addresses.

The number of allowable free space fragments is not known, but is estimated to be between 38 and 52. If this problem occurs, the user should copy and compress the volume, and map the compressed version. If the problem still exists, notify the Programmer Assistance Center.

9.4.2 PATRICK

PATRICK is a utility program designed for copying, dumping, and error checking any sequential data set on disk, tape, or cards. It can handle up to 255 files starting with any file and ending with any file. PATRICK is available on the 360/95 and 360/75, but not on the 360/65. It can be used to perform such functions as:

- Card to card Tape to card Disk to card
- Card to tape Tape to tape Disk to tape
- Card to disk Tape to disk Disk to disk

It should be noted here that some functions, such as card to tape and tape to card, should be done on the model 30 provided for that purpose.

9.4.2.1 PATRICK Execute Card

The following is a sample PATRICK EXEC card:

```
//COPY    EXEC      PGM=PATRICK,PARM='9TH,002,004',REGION=85K
```

The three PARM parameters indicate the routine to be performed, the first data set to be copied, and the last data set to be copied. The value of the third parameter must be greater than or equal to the value of the second parameter.

9.4.2.2 ROUTINE Parameter

The first parameter, called the ROUTINE parameter, consists of three characters, as follows:

a. The first character -- this describes the input data set:
   - 7 - indicates a 7-track tape
   - 9 - indicates a 9-track tape, disk, or card data set

b. The second character -- this indicates the operation to be performed:
   - T - create duplicate of input data set
   - N - no duplicate created

c. The third character -- this indicates the type of printed output or dump:

- O - for octal; used for 7-track tapes with convert and translate off.
- H - for hexadecimal; used for 9-track tapes with non-EBCDIC characters such as binary tapes, 7-track tapes with convert on, or used any time a hex printout is desired.
- B - for BCD or EBCDIC; used for 9-track tapes with EBCDIC characters or 7-track BCD tapes with translate on. This printout is formatted 80 characters per line.
- N - for no printout.
- L - for list; used to list each record, but not print out the record itself. It will print only record number, blocksize, and file number. This works best using RECFM=U in the DCB on the DD card. (See following information on DD cards.)

There are two exceptions to the format of the ROUTINE parameter:

TST - used for tape testing

DMP - used to generate a formatted dump of a magnetic tape

Figure 9.4-1 illustrates the valid entries for the ROUTINE parameter.

9.4.2.3 REGION Parameter

The region required for execution of PATRICK is computed by the following algorithm:

Program Size	10K
2 x (BLKSIZE of IN1 DD card)	? K
2 x (BLKSIZE of OUT1 DD card)	? K
OUT2 DD card	20K
TOTAL	(MIN. of 60K)

9.4.2.4 PATRICK DD Cards

PATRICK uses three DD cards to describe the input data set (IN1), output copy (OUT1), and printed output (OUT2):

- The IN1 DD card is the normal input data set and is always required.

OUTPUT IDENTICAL SET	TYPE OF PRINTOUT	INPUT DATA SET				
		9-TRK EBCDIC	9-TRK ANY	7-TRK (CONVERT ON)	7-TRK (TRANSLATE ON)	7-TRK (NONE ON)
YES	EBCDIC	9TB	---	---	7TB	---
YES	HEX	9TH	9TH	7TH	7TH	---
YES	OCTAL	---	---	---	---	7TO
YES	LIST	9TL	9TL	7TL	7TL	7TL
YES	NONE	9TN	9TN	7TN	7TN	7TN
NO	EBCDIC	9NB	---	---	7NB	---
NO	HEX	9NH	9NH	7NH	7NH	---
NO	OCTAL	---	---	---	---	7NO
NO	LIST	9NL	9NL	7NL	7NL	7NL
NO	DMP	DMP	DMP	DMP	DMP	---
TEST ONLY		TST	TST	TST	TST	TST

Figure 9.4-1. Entries for Routine Parameter for PATRICK

- The OUT1 DD card is required only if T (CREATE duplicate of input data set) is used in the PARM field on the EXEC card (e.g., 9TH).

- The OUT2 DD card is the SYSOUT card and is always required. If this card is omitted, a user 413 ABEND occurs.

9.4.2.5 Example of Copying Tape to Disk

This example illustrates copying a 9-track tape to disk with an EBCDIC printout of the data being copied. EROPT=ACC will accept an I/O error and print the error analysis message.

```
//COPY          EXEC      PGM=PATRICK,PARM='9TH,001,001',REGION=100K
//IN1           DD        UNIT=9TRACK,VOL=SER=xxxxxx,LABEL=(,SL),
//                        DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),
//                        DEN=3,EROPT=ACC),DISP=(OLD,KEEP)
//OUT1          DD        UNIT=2314,VOL=SER=yyyyyy,SPACE=(CYL,(5,5)),
//                        DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200,
//                        EROPT=ACC),DISP=(NEW,KEEP)
//OUT2          DD        SYSOUT=A,SPACE=(CYL,(5,1))
```

The DCB parameters must agree with the data actually present on the tape; the values shown here are for illustrative purposes only.

9.4.2.6 Tape Testing Example

The TST parameter causes the program to check a tape for errors, print out a message describing the error, and print the bad record in hex and EBCDIC if data was transferred. The number of bytes transferred will be printed as the record length. (This length is the length of the physical record [block] in which the error was detected minus the number of unreadable bytes.) The following information is printed:

- Jobname
- Stepname
- Unit address or unit detecting error
- Device type
- DD name of data set (IN1)
- Operation attempted (GET)
- Error description -- General Description; will vary by device

- Block number for tape or track address for direct access
- Access method (QSAM)
- Sense bytes in hexadecimal

The following is a sample program for tape testing:

```
//TEST          EXEC      PGM=PATRICK,PARM='TST,001,007',REGION=100K
//IN1           DD        UNIT=9TRACK,VOL=SER=xxxxxx,LABEL=(,SL),
//                        DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200,
//                        DEN=3,EROPT=ACC)
//OUT2                    SYSOUT=A,SPACE=(CYL,(5,1))
```

9.4.2.7 DMP Parameter

The parameter DMP will produce a dump of a tape like the dump given at ABEND. It will format the dump like an ABEND dump with a hex printout on the left side of the paper and EBCDIC on the right side. Any unprintable character will be made blank. This printout is effective only with 7-track BCD tapes with 'translate' on or 9-track EBCDIC tapes. Binary tapes can be dumped more efficiently using either a hex dump for 9-track or octal dump for 7-track.

To dump a few records from a tape, a small space should be allocated, e.g., SPACE=(CYL,1). This will print about 1000 lines of output and abend with a B37 completion code, but the output will be printed.

To estimate the amount of space required to print a data set, one cylinder should be allowed for each 1000 lines. A line contains 80 bytes for an EBCDIC printout, 48 bytes for hexadecimal, and 48 bytes for octal.

9.4.2.8 Other Uses

- Finding blocksize -- If the blocksize of a file on tape is unknown, the third character of the ROUTINE parameter has a list option (L) which will print only the record number, block size, and file number. The following JCL specifications should be made:
  - EXEC card - use 7NL or 9NL for the ROUTINE parameter, as appropriate
  - IN1 DD card - use RECFM=U,BLKSIZE=32000
  - OUT1 DD card - use DUMMY
  - OUT2 DD card - a small space should be allocated to dump only a few lines

- Multiple Printouts of Data Sets -- Multiple printouts of data sets on disk can be achieved by using several executions of PATRICK to print one copy per execution. When the data set is first created, it should be given a temporary data set name, and the DCB information, UNIT=DISK, and DISP=(NEW,PASS) should be specified. To reprint the temporary data set in the next job step with PATRICK, the following JCL changes should be made:

  - EXEC card - PARM='9TN,001,001'

  - IN1 DD card - UNIT=DISK,DISP=OLD,DSNAME=temporary data set name

  - OUT1 DD card - DUMMY

  - OUT2 DD card - SYSOUT=A,DCB=same as IN1; DISP=PASS must be used to pass the data set to each step until all PATRICK steps are executed

  The data set will be deleted at the end of the job because it was created as temporary.

- A tape-to-tape copy may be done by changing the OUT1 DD card of the tape-to-disk example to reflect tape output. The following changes can be used for more efficient tape-to-tape copying. RECFM=U and a blocksize equal to or larger than the largest block size on IN1 should be used. The user should do the same for OUT1. The records will be copied in their proper size, and less CPU time will be used for blocked records.

9.4.3 IEBFGR

IEBFGR is a utility whose function is to delete or rename members of a PDS. It can also be used to create an alias for an existing member of a PDS. The program was written by Frank G. Ross, Code 543.

9.4.3.1 Control Cards

The control cards are free format, and the information may appear in columns 1-63, as shown in the following examples:

```
DELETE      MEMBER=membername to be deleted

RENAME      MEMBER=oldname, NEWMEM=new name

ALIAS       MEMBER=membername, NEWMEM=alias
```

9.4.3.2 IEBFGR Example

```
//FRANK          EXEC     PGM=IEBFGR
//SYSPRINT       DD       SYSOUT=A
//DD1            DD       DSN=name,UNIT=2314,VOL=SER=xxxxxx,
//                        DISP=(OLD,KEEP)
//SYSIN          DD       *
DELETE                   MEMBER=membername
RENAME                   MEMBER=oldname,NEWMEM=newname
ALIAS                    MEMBER=membername,NEWMEM=newalias
```

where the input DD statement must be named DD1.

Diagnostics are issued for the action taken.

9.4.4 OSSLIP

The Operating System Source Library Inquiry Program (OSSLIP) is a program designed to store, retrieve, and update sets of card image records in a sequential data set. Each set of cards is called a "MEMBER" of the sequential data set. The first record of each set is called the HEADER RECORD and contains the member name and other identifying information. Each member is stored by name and may be retrieved, modified, or deleted by name.

The OSSLIP data may contain any type of source data such as source programs, data, or procedures. Object decks may also be stored, retrieved, and deleted. The order of the members (as defined by the user) does not have to be alphabetic.

9.4.4.1 Control Card Format

OSSLIP uses fixed-format control cards to specify the operations to be performed and related operands. The control card format is:

- Operation field - Columns 1-6 specify one of eleven (11) operations available.

- Options field - Column 7 may contain a character which specifies an available option.

- Member ID field - Columns 8-30 contain the name of the member to be processed and other optional information. The name contains one to eight alphanumeric characters, beginning with a letter and omitting imbedded blanks. The name may be specified in one of two ways. If it is the first string of non-blank characters in the field, it may be left-justified or preceded by one or more blanks; it must be followed by at least one blank. Any other information must follow the name. The name may appear anywhere in the field if it is preceded and followed by a delimiting character, i.e., xnnnnnnnny. The delimiting characters may be the same or different and are used by a SCAN parameter in other operations to identify the member name.

  The member ID field in the control cards must match the member ID field in the label header records in the SLIPIN data set.

- Member name field - Columns 31-38 are used only with the MEMBER operation.

- Comments - Columns 31-80 may contain any comments (39-80 in the MEMBER statement).

9.4.4.2 JCL Conventions

OSSLIP uses the standard SYSPRINT and SYSPUNCH DD statements. The input source file is defined by the SLIPIN DD statement; the output is defined by the SLIPOUT DD statement. SYSIN defines the normal input, including control cards and data. If an alternate input tape is used, it is defined by any valid DD name which is then used on the INCLUD control card. Members from a SLIP tape may be placed in a PDS called SLIPPDS, which must be defined by a DD statement.

The source tape, SLIPIN, is retained, and a new tape, SLIPOUT, is written by merging the SLIPIN data with the data from SYSIN or a tape containing card images. These data must be in the same sequence as the members on SLIPIN. The operations are discussed in detail in the following section.

The normal mode of operation for OSSLIP is to read decks as SYSIN data. However, OS/360 will scan any JCL cards imbedded in a library program, usually with undesirable results. It is preferable to "card-to-tape" new input for each OSSLIP run, at least if OS/360 programs are included.

9.4.4.3 Operations

The OSSLIP utility control statements may be divided into three general categories:

- Operations which directly modify members of the SLIPIN data set (UPDATE, DELETE, RENAME).
- Operations which specify a tape action (INCLUD, COPYTP, REWIND).
- Operations which process the SLIPIN data set (LSTALL, MEMBER, PUNCH1, LSTONE, LSTPCH).

9.4.4.4 UPDATE, DELETE, RENAME

UPDATE adds, replaces, or changes members in the SLIPOUT data set. An UPDATE card precedes each set of update cards. Columns 8-30 are compared to the header labels on SLIPIN. The old member, if any, and its header label are deleted. The new header label is formed from columns 8-80 of the UPDATE card, and the new member is read in from SYSIN (or from the DDNAME defined in an INCLUD card). If the UPDATE card does not match an existing header label, the new member is placed at the end of the SLIPOUT data set. A './END' card denotes the end of the input card deck and must precede another UPDATE card or other control card.

An "S" in column 7 of an UPDATE card will cause sequence numbers to be generated in columns 73-80 of the cards, as they are written on SLIPOUT. The default increment value is 10, but may be altered to 1-99999 by the execution parameter INCR=XXXXX.

A "P" in column 7 of the UPDATE card denotes a partial update in which cards are to be added, deleted, or replaced within a member. A card with the same sequence member as an existing record will replace it; cards with no matching number will be added. A card with './DELETE' in columns 1-8 and a sequence number in columns 73-80 will cause that record to be deleted. A './END' card denotes the end of the input data set for a partial update.

Each new deck to be added or changed on a library tape must be preceded by an UPDATE card and followed by an './END' card. A succession of such decks must be in ascending sequence by position on the tape.

DELETE deletes the member specified in columns 8-30 of this control card from the SLIPOUT data set.

RENAME replaces columns 8-80 of the header label for the member specified in columns 8-30 of the control card, with columns 8-80 of the card following the RENAME card.

9.4.4.5 INCLUD, COPYTP, REWIND

INCLUD directs OSSLIP to use DDNAME xxxxxxxx as input instead of SYSIN. If input is to be provided on an alternate input tape or direct-access device, the decks must first be written on the input device, in ascending order with an './END' card following each deck. The UPDATE cards are supplied on SYSIN following an INCLUD card. These UPDATE cards must match the decks on tape or disk one-for-one, since there is no way for OSSLIP to verify the correct correspondence between them.

Merging two SLIP data sets may be done by defining one as SLIPIN and the other on an INCLUD card. This INCLUD control card followed by a COPYTP control card will result in a SLIPOUT data set containing all the members contained on the tape or data set specified by the INCLUD card, followed by all those on SLIPIN.

COPYTP copies the SLIPIN data set onto SLIPOUT.

REWIND rewinds SLIPIN and resets all switches and counters. SLIPOUT and SYSPUNCH are closed with data set disposition provided by the DD cards.

9.4.4.6 LSTALL, MEMBER, PUNCH1, LSTONE, LSTPCH

LSTALL lists all header labels on SLIPIN and the number of cards on each MEMBER. Certain characters in column 7 of the LSTALL card will cause control cards to be punched for all MEMBERS on SLIPIN for later use.

Character in Column 7	Control Card Punched for Every MEMBER on SLIPIN
A	UPDATE
B	LSTPCH
C	MEMBER
X	LSTONE
Y	PUNCH1
Z	DELETE

MEMBER writes the member specified in columns 8-30 of this control card into the partitioned data set defined by SLIPPDS (a direct-access storage device defined on a DD card). The name given to the MEMBER of the SLIPPDS data set is defined in columns 31-38. An "S" in column 7 of the control card will cause sequence numbers to be generated in columns 73-80 of the card images as they are written on SLIPPDS.

PUNCH1 punches cards from the member on tape specified in columns 8-30. An "S" in column 7 causes sequence numbers to be punched in columns 73-80 of the cards. An "M" in column 7 causes the selected member to be written into SLIPPDS. The name assigned to the member in SLIPPDS is the value taken from the member ID field of the header label by one of two methods:

- The member ID field is scanned for the first non-blank character. The member name starts with the first non-blank character and is up to eight characters in length. It is terminated by a blank character.

- If the execution parameter SCAN=XY is used, the member name must be in the form XnnnnnnnnY. The member ID field is scanned for the first SCAN parameter (X). The member name is the one to eight characters which occur between the first delimiter (X) and the second delimiter (Y).

A "B" in column 7 causes both options specified by "S" and "M" to be performed.

LSTONE lists the cards for the member specified in columns 8-30 of the control card. See PUNCH1 for options. This operation may be used to closely check the update of a particular member.

LSTPCH lists and punches cards for the member specified in columns 8-30. See PUNCH1 for options.

9.4.4.7 UPDATE, RENAME, DELETE Example

```
//SLIP              EXEC        PGM=OSSLIP
//SYSPRINT          DD          SYSOUT=A
//SLIPIN            DD          UNIT=9TRACK,VOL=SER=xxxxxx,LABEL=(1,BLP),
//                              DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,
//                              BLKSIZE=3200),DSNAME=SLIPIN
//SLIPOUT           DD          UNIT=9TRACK,VOL=SER=xxxxxx,LABEL=(1,BLP),
//                              DISP=(NEW,KEEP),DSNAME=SLIPOUT,
//                              DCB=*.SLIPIN
//SYSIN             DD          *
UPDATEM        SRCEDECK

     (New deck to replace existing deck)
./END
UPDATEPASTRO

     (Change cards with sequence number in columns 73-80)

./END
RENAME STARFIRE

     (Rename card with new name, comments, etc.)

DELETE SATELLITE
```

In the above example:

- A new member replaces an old member with the same name (SRCEDECK).
- A partial update is made to the ASTRO member.
- The header label for member STARFIRE is replaced by columns 8-80 of the card following the RENAME card.
- The member SATELLITE is deleted.

9.4.4.8 INCLUD Examples

```
//SLIP              EXEC        PGM=OSSLIP
//SYSPRINT          DD          SYSOUT=A
//SLIPIN            DD          UNIT=9TRACK,VOL=SER=xxxxxx,
//                              LABEL=(,SL),DISP=(OLD,KEEP),DSNAME=SLIPIN,
//                              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
```

```
//SLIPOUT          DD          UNIT=9TRACK,VOL=SER=xxxxxx,
//                             DSNAME=SLIPOUT,
//                             DISP=(OLD,KEEP),DCB=*.SLIP.SLIPIN,
//                             LABEL=(,SL)
//SLIPNEW          DD          UNIT=9TRACK,VOL=SER=zzzzzz,LABEL=(,SL),
//                             DISP=(OLD,KEEP),DCB=*.SLIP.SLIPIN,
//                             DSNAME=SLIPNEW
//SYSIN            DD          *
```

The following sets of control cards may be used as input to SYSIN.

```
INCLUD    SLIPNEW
UPDATE   SRCEDECK
UPDATEMASTRO
UPDATEPSATELLITE
```

The card sets for the update must be on SLIPNEW; each set must be followed by a ./END card; the sets must be in the same sequence as the control cards, and have a one-to-one correspondence; both must be in the same sequence as the SLIPIN data set.

```
INCLUD    SLIPNEW
COPYTP
```

This set of control cards will copy SLIPNEW to SLIPOUT, followed by SLIPIN.

9.4.5 UPDATE UTILITY FOR SOURCE AND OBJECT FILES

Known locally as Charlie Newman's Utilities, these two routines were written by Charles R. Newman, Code 551. They allow a programmer to make symbolic updates to selected subprograms, compile these updated routines, and update an object file deleting the old object decks. The source and object files handled by these routines are sequential data sets. When used to update both source and object files or to list the source file, these routines result in a considerable saving of computer time in comparison to the OS/360 utilities. The routines are written in FORTRAN for OS/360 and are run on the IBM 360, models 91 and 95.

The object decks for the utility routines are on the G1USR2 disk pack (360/95) and may be used as follows:

```
Source Utility Program,
//stepname.SYSLIN    DD    DSN=G1.G7CRN.SUP,DISP=SHR,
//                         UNIT=2314,VOL=SER=G1USR2

Object Utility Program,
//stepname.SYSLIN    DD    DSN=G1.G7CRN.OUP,DISP=SHR,
//                         UNIT=2314,VOL=SER=G1USR2
```

The GSFC Computer Program Library number for these routines is D00145; they are described in the GSFC document X-551-69-409.

9.4.6 LOAD MODULE MAP PROGRAM (LMODMAP/IMBMDMAP)

The program IMBMDMAP is a release 19 version of the program currently known as LMODMAP. It is designed to aid the system programmer in the diagnosis of system or application program failures.

Used in conjunction with main storage dumps, the load module map enables the user to identify and locate individual control sections and their entry points, and to verify load module attributes and alias.

A load module map produced by LMODMAP contains edited information regarding the control sections (CSECTS), entry points (EP), aliases, external references, attributes, type codes, overlay segments, and hierarchy designations for each load module for which a map is requested.

LMODMAP may be used to map:

- A Systems nucleus
- The link pack area
- Any load module previously link edited into a PDS

The procedure to execute the program LMODMAP on the model 95 is called LDMODMAP; on the model 75, it is called MAPNUC (map nucleus). The model 65 does not have a LMODMAP procedure, but has a mapping procedure, IMBMDMAP, in SYS1.LINKLIB.

Documentation on LMODMAP is generally not available, but the description of IMBMDMAP in the release 19 IBM manual, Service Aids (GC28-6719), applies equally to LMODMAP.

SECTION 10

AUTOFLOW

10.1 INTRODUCTION

AUTOFLOW, a proprietary software system of Applied Data Research, Inc., automatically translates the source language of a program into flowcharts, and then prints them out on the printer. Alternatively, the programmer may choose to have his flowcharts on an output tape for later use.

Flowcharting of programs is a service done on the SSA S/360-75J in Building 21, for programs written for the IBM S/360 computers, as well as the CDC 3200, DDP 24, 124, 224, XDS 910, 920, 930, and Univac 1108 computers. Instructions for the use of AUTOFLOW are summarized in this section of the User's Guide.

10.2 GENERAL DESCRIPTION

AUTOFLOW accepts as input either decks or tapes in COBOL, FORTRAN, PL/I, or assembly language for the S/360; and also assembly language or FORTRAN decks or tapes for CDC, DDP, XDS, and Univac 1108 computers. AUTOFLOW, and its preprocessors for computers other than S/360, follows all the rules prescribed by the programming manuals of the particular computer being flowcharted.

Accompanying the flowcharts are tables of contents and cross references; tables of diagnostics pointing out logic errors, syntax errors, missing references, etc.; and an optional listing of the source program. (AUTOFLOW prints up to 1000 large flowchart pages for each program. This is enough for most large programs.)

In addition, AUTOFLOW has special chart-oriented languages. By punching special chart codes in the comments portion of cards, the programmer can adjust details in the flowcharts, or add more explanatory text.

AUTOFLOW accepts the following different kinds of source programs:

1. S/360 programs:
   - Cards or tape input.
   - Single program or multiple programs in one job.
   - One or more languages, including assembly, FORTRAN, PL/I, and COBOL.
2. CDC, DDP, SDS, and Univac 1108 programs:
   - Cards or tape input.
   - Single program or multiple programs in one job.
   - Assembly language and FORTRAN.
   - Univac 1108 List Tape.

10.3 AUTOFLOW JOB SUBMISSION

The three requirements that apply to all programs submitted for flowcharting, regardless of the computer the program was written for, or whether the source program is on cards or magnetic tape, or the type of output desired, are: a job submission slip for the computer facility in Building 21; a S/360 JOB card (a punched card that precedes the deck); and the applicable set of job control cards to execute AUTOFLOW.

10.3.1 JOB SUBMISSION SLIP

This form (obtainable at the dispatcher's desk in Building 21) must have, in addition to certain self-explanatory entries (such as tape numbers, if any) some added instructions written in the REMARKS space to indicate to the operator any special output, special printer-control, special paper, or required region size. If the user indicates special types of output, other instructions are required on the job submission slip; these are discussed in the following paragraphs.

10.3.1.1 Remarks for Standard AUTOFLOW Output

The following information is written under REMARKS:

Q = 8 LPI. Convex-fold, 2-part, plain paper. Region = 250K.

This AUTOFLOW output must be run on a special printer, at night; therefore, the user may not receive his job until the following morning; or at busy times, he may have to wait another 24 hours.

10.3.1.2 Remarks for Regular Tape or Plotter Tape Output

For quick turnaround, at the cost of appearance, AUTOFLOW output can be accepted on the regular printer, single copy, on lined paper instead of plain paper. Since the regular print control is set at 6 lines per inch (instead of the 8 LPI mentioned above), the flowcharts will be spread out over more paper, but will be quite readable; in this manner, a user's AUTOFLOW job will be completed in one day. To use this option, the job submission slip should include the following reemark: Region = 250K. All references to Q output or plain paper should be eliminated. This can be done by inserting the following DD card in the deck:

```
//SYSPRINT     DD   SYSOUT=A
```

10.4 THE JOB CARD FOR AUTOFLOW RUNS

The program (deck or tape) must be accompanied by a standard Goddard S/360 JOB card, and by applicable job control cards as described in the referenced documentation. Note that the JOB card and the job control cards must be punched on the Model 029 keypunch, no matter what type of keypunch was used for the source program.

The JOB card is the first card in the deck, and contains the standard required accounting information.

10.4.1 REQUIRED ENTRIES

The required format of the JOB card for an AUTOFLOW run is the same as the standard GSFC S/360 JOB card. Special AUTOFLOW parameter values are described in Table 10.4-1. For users of CDC, DDP, XDS, Univac, and computers other than S/360, see special instructions and an example of JOB card format in paragraph 10.4.2, Special Entries for Non-360 Users.

Table 10.4-1. AUTOFLOW JOB Card Parameter Values

Col. 30-35	The required program number is G00169 for all AUTOFLOW jobs on the S/360, no matter what program number the source deck usually runs under.
Col. 37-39	Minutes of estimated CPU time, padded at left with leading zeroes. Normally, 5 minutes should be enough for a source deck no larger than one carton of cards. If the user underestimates the required CPU time, his job will be terminated without a printout, and he will receive a system code of 322. In this case, he should repunch these columns on his JOB card to increase the time estimate, and should resubmit the job. (It is much safer to overestimate.)

Col. 40-42 Minutes of estimated I/O time. Usually AUTOFLOW I/O time runs about double the CPU time. However, the user should be generous in his estimate. (If he underestimates I/O time, he will receive a system code F22, and the job will terminate without completing the printout. In this case, he should repunch these columns to increase I/O time, and should resubmit the job.)

Col. 45-47 The user's box number at the S/360-75 computer room. If the user does not have a box there, he should punch DIS in these columns, and ask the dispatcher for the job.

In the event that the user submits two or more separate programs for AUTOFLOW flowcharts at the same time, or submits a second program before receiving the first one back, he must use a different job name on each of the jobs. (For example, if one job has a name of GAZZZJOE, he could vary this, for the second job, to GAZZZJO2, etc.) This is necessary because the S/360 OS MVT processes many jobs at the same time, and the presence of two jobs with the same name may cause unpredictable problems.

10.4.2 SPECIAL ENTRIES FOR NON-360 USERS

For AUTOFLOW users who do not have a programmer's ID and/or authorized time for S/360 computers (this would be the case for programmers on the CDC, DDP, XDS, Univac, or other computers at Goddard), the JOB card will be punched the same as described in paragraph 10.4.1, except that a special programmer's ID and a special sponsor number will be used.

10.5 AUTOFLOW JOB CONTROL CARDS

The third requirement for submitting any program for AUTOFLOW flowcharting is a set of job control cards appropriate to (1) the computer for which the program was written, (2) the symbolic language(s) used, (3) the type of source input (deck or tape), and (4) the type of output desired (computer printout or tape). In order to reduce the number of job control cards which the user must punch, cataloged procedures are provided for the various kinds of AUTOFLOW jobs.

10.5.1 CATALOGED PROCEDURES: ADRFLOW, PPEX, ADRPLOT

Cataloged procedures for executing AUTOFLOW programs exist in the S/360-75 procedure library. To use these sets of control cards, a control card should be furnished, including the EXEC instruction and the name of the procedure (e.g., // EXEC ADRFLOW). In addition, other job control cards should be supplied, as required. (Note: To permit retaining in the program the existing

job control cards that were put there for regular processing of the source program, a //INPUT DD DATA card should be used in all the deck setups, instead of a //INPUT DD * card.)

The cataloged procedures are:

1. ADRFLOW: For S/360 programs (assembly language, FORTRAN, PL/I, or COBOL).

2. PPEX: For programs written for CDC, DDP, XDS, or Univac computers, for which AUTOFLOW must use a preprocessor (assembly language or FORTRAN only).

3. ADRPLOT: Used to obtain special output prints or microfilm of the flowchart, by means of the S-C 4020 plotter (assembly language, FORTRAN, PL/I, or COBOL programs written for any of the computers listed above).

10.5.2 ADRFLOW PROCEDURE

```
//DEFAULT PROC OUT=Q,SET='G.S.F.C',CHAIN=PN,DEPTH=150,LINEAR=NO
//ADR     EXEC PGM=AUTOFLOW,REGION=250K,
// PARM='&SET,CHAIN=&CHAIN,DEPTH=&DEPTH,LINEAR=&LINEAR'
//STEPLIB      DD   DSN=SYS2.AUTOFLOW,DISP=SHR
//SYSAF01      DD   UNIT=2314,SPACE=(CYL,(20,10)),
//                  DCB=(,BUFNO=2,BUFL=7000,BLKSIZE=7000)
//SYSAF02      DD   UNIT=(2314,SEP-SYSAF01),SPACE=(CYL,(20,10)),
//                  DCB-(,BUFNO=2,BUFL=7000,BLKSIZE=7000)
//SYSAF03      DD   UNIT=(2314,SEP=(SYSAF01,SYSAF02)),SPACE=(CYL,(20,10)),
//                  DCB=(,BUFNO=2,BUFL=7000,BLKSIZE=7000)
//SYSAF04      DD   UNIT=(2314,SEP=(SYSAF01,SYSAF02,SYSAF03)),
//                  SPACE=(CYL,(20,10)),
//                  DCB=(,BUFNO=2,BUFL=7000,BLKSIZE=7000)
//SYSAF05      DD   UNIT=2314,SPACE=(CYL,(20,10)),
//                  DCB=(,BUFNO=2,BUFL=7000,BLKSIZE=7000)
//SYSAF06      DD   UNIT=2314,SEP=SYSAF05),SPACE=(CYL,(20,10)),
//                  DCB=(,BUFNO=2,BUFL=7000,BLKSIZE=7000)
//SYSPRINT     DD   SYSOUT=&OUT,DCB=(RECFM=FBM,LRECL=133,BLKSIZE=3325)
//*                                                              -CWB-
```

To execute this procedure, code:

```
//PLOT    EXEC      ADRFLOW
//INPUT   DD        DATA
  (source program deck, including JCL)
```

SECTION 11

OS EXECUTIVE FEATURES

11.1 SYSTEM VIEW OF DATA MANAGEMENT

Data Management is that portion of the operating system that carries out the identification, cataloging, storage, and retrieval of data sets. A data set for the OS/360 System is a collection of related data that is broken into groups of information called records. (Note: On second-generation computers a data set was referred to as a "file" of information.) A logical record may be defined in terms of the information it contains. For example, discrete information concerning users of the 360 system could be stored in a data set consisting of one logical record per user. A physical record may be defined by the manner in which it is stored and retrieved. Logical records may be grouped together (blocked) or be separated by a gap (unblocked) or broken into segments (spanning two or more physical records).

The standard unit of peripheral storage is a volume which may be a magnetic tape, disk pack, drum, or data cell. All direct-access volumes must have a volume label (magnetic tapes may or may not be labeled) so that they are readily identifiable to the system. The label for a Direct-Access Storage Device (DASD) contains the volume serial number and a pointer to the Volume Table of Contents (VTOC). The VTOC contains the name, description, and location of each data set stored on the volume, plus a record of the unused areas on the volume. For magnetic tape volumes, a standard label consists of a volume label and groups of data set labels. There is no VTOC on a tape volume. The volume label is the first record on the tape and contains the volume and owner identifications. This is followed by the data set header record, a tape mark, the data set, and the data set trailer record. Many data sets may be placed on tape in this manner. The final data set trailer record would be followed by two tape marks. The information contained in the header and trailer records is made available to the system and is used in conjunction with or in place of information normally supplied by the DCB parameter on the Data Definition (DD) card.

The Data Management System communicates with a program through a set of standard interfaces: the Data Control Block (DCB) which is built by a program at compile time; the Data Definition (DD) statement (in the job control stream); the Data Set Control Block or label (DSCB), which is actually part of the data set; and the system macro instructions which provide access to the Operating System facilities.

A job step can place a data set in direct-access storage by a DD statement in the job control stream; the DD statement must specify the type of unit, (e.g., disk), volume id, data set name, disposition, and space required. At the time the job step is initiated, the space is allocated and a label is created for each area requested. During job step execution, the label is completed and updated via OPEN and CLOSE macro instructions. The system and user can also create temporary data sets. Temporary data sets are designed to last only for the job step or the life of the job and are then deleted by the system.

The Data Management System provides the facility of cataloging frequently used data sets. The catalog is maintained in direct-access storage and contains information to identify the volume containing the cataloged data set and the device type, freeing the programmer from specifying that information on the DD card referring to the data set.

The OS/360 Data Management System provides the programmer with the capability of organizing a data set in one of four ways:

a. Sequential Organization

   Data sets residing on serial-access devices, such as magnetic tapes, must have sequential organization. This organization is optional for data sets residing on direct-access devices.

b. Indexed Sequential Organization (For direct-access devices only)

   Records are stored sequentially with a key (record-identifier) contained in the record. The system maintains an index of the record locations; this system allows the records to be accessed by key as well as sequentially.

c. Direct Organization (For direct-access devices only)

   Records are randomly accessed by specifying their relative position in the data set, or by the absolute record address appropriate to the device.

d. Partitioned Organization (For direct-access devices only)

   A data set is divided into members which are organized sequentially. Member names and locations are kept in a directory associated with the data set. The members consist of one or more blocks. This organization allows the random retrieval of named blocks of data which are sequentially organized.

In addition to having the capability of organizing data sets in the manner described above, the programmer also has a choice of different methods for accessing the data set. The queued access method, which only applies to sequential data sets, manages the buffers automatically for the programmer. The basic access method gives the programmer control over the blocking and buffering.

A Data Control Block (DCB) is associated with each data set referenced by a program. The DCB must be initialized before any data transfer can take place. The data control block is generated and partially initialized by the DCB macro instruction at compile time. The OPEN macro instruction generates a call at execution time to a routine which completes the DCB with information from a DD statement in the job stream, or from a data set label. The OPEN routine also loads and resolves the required access routines, prepares the buffer areas, generates channel command lists, and initializes data sets by reading (or writing) data set labels.

To summarize, the Data Management System provides a great deal of flexibility in organizing and accessing data sets, and, by dynamically loading the relevant routines at execution time, source programs can be generally device-independent.

11.2 SYSTEM-ORIENTED MACROS

The OS Executive provides to the problem program a wide variety of services through the use of assembly language macros. (FORTRAN and other higher level language programmers have many of the services available through the language itself.) These macro instructions allow the programmer to specify the function and required parameters in a straightforward manner; SVC in-instructions do not have to be used directly. A complete list of macro instructions with coding requirements and macro expansions is found in IBM Supervisor and Data Management Macro Instructions (GC28-6647).

The data management macro instructions provide the means to access data sets, build DCBs, buffer pools, etc., for the various access methods. A list of some of the relevant macro instructions with access methods is in Table 11.2-1.

In the area of supervisor macros, means are provided to manage the resources at the programmers' disposal, i.e., ENQ is used to request the use of a serially re-usable resource, and GETMAIN is used to allocate main storage.

Other services are provided to: (1) create subtasks (ATTACH macro), (2) wait for events (WAIT macro), (3) control the execution (CALL and RETURN macros), (4) bring load modules into main storage (LOAD macro), (5) load an overlay segment (SEGLD macro), and (6) dump portions of the program (SNAP macro). The ABEND macro can be used to abnormally terminate a task. The SPIE and STAE macros are used to control interrupt exit processing and ABEND processing.

A summary of supervisor macro instructions is contained in Table 11.2-2.

Table 11.2-1. Access Method Macros

Macro Instruction	Access Method						Macro Instruction Function
	QSAM	QISAM	BSAM	BPAM	BISAM	BDAM	
DCB	.	.	.	.	.	.	Generate a data control block
OPEN	.	.	.	.	.	.	Open a data control block
CLOSE	.	.	.	.	.	.	Close a data control block
BUILD	.	.	.	.	.	.	Structure named area as a buffer pool
GETPOOL	.	.	.	.	.	.	Allocate space to and format buffer pool
FREEPOOL	.	.	.	.	.	.	Liberate buffer-pool space
GET	.	.					Obtain a record from an input data set
PUT	.	.					Include a record in an output data set
PUTX	.	.					Include an input record in an output data set
RELSE	.	.					Force end of input block
TRUNC	.						Force end of output block
FEOV	.		.				Force end of volume
CNTRL	.		.				Control reader or printer operation
PRTOV	.		.				Test for printer carriage overflow
SETL		.					Set lower limit for scan
ESETL		.					Postpone fetching during scan
CHECK			.	.			Wait for I/O completion and verify proper operation
NOTE			.	.			Note where a block is read or written
POINT			.	.			Point to a designated block
FIND				.			Obtain the address of a named member
BLDL				.			Build a special directory in main store
STOW				.			Update the directory
RELEX					.	.	Release exclusive control of a block
FREEDBUF					.	.	Free dynamically obtained buffer
GETBUF			.	.	.	.	Assign a buffer from the pool
FREEBUF			.	.	.	.	Return a buffer to the pool
WAIT			.	.	.	.	Wait for I/O completion
READ			.	.	.	.	Read a block
WRITE			.	.	.	.	Write a block

Table 11.2-2. Supervisor Service Macros

Macro Instruction	Function
ABEND	Abnormally terminate a task
ATTACH	Create a new task
CALL	Pass control to a control section
CHKPT	Take a checkpoint within a job step
DELETE	Release a load module
DEQ	Release a serially re-usable resource
DETACH	Delete a subtask
ENQ	Request a serially re-usable resource
FREEMAIN	Release allocated main storage
GETMAIN	Allocate main storage
IDENTIFY	Add an entry point
LINK	Pass control to a load module
LOAD	Bring a load module into main storage
POST	Signal event completion
RETURN	Return control
SAVE	Save register contents
SEGLD	Load overlay segment
SNAP	Dump main storage and continue processing
SPIE	Specify program interrupt exit
STAE	Specify ABEND exit
STIMER	Set interval timer
TIME	Provide date and time
TTIMER	Test interval timer
WAIT	Wait for event
XCTL	Pass control to a load module

11.3 CONDITION CODES AND COMPLETION CODES

One useful feature of OS is the ability to pass codes to subsequent steps of a job. Two types of codes may be passed. When a job step terminates normally, the code is usually referred to as a condition code. When a job step terminates abnormally, the code is referred to as a completion code. In either case, not returning a code is equivalent to returning a code of zero.

When a job step terminates normally, a condition code is set by the processing program. The condition code can be a decimal value from 0-4095. In FORTRAN, PL/I, or an assembly language program, the code can be set by the programmer during program execution. Most system processors set the code to indicate the severity level of the most severe error encountered. There is only one condition code returned per job step. The condition codes can be tested by setting condition (COND) parameters in the JOB and EXEC statements; thus, subsequent processing of job steps can be made dependent on the condition codes of previous steps. Refer to Section 5 of this User's Guide and the IBM Job Control Language User's Guide (GC28-6703) and Job Control Language Reference (GC28-6704) for more information on the COND parameter.

When a job step terminates abnormally, a completion code is set, either by the programmer (as in the ABEND macro), in which case it is preceded by a "U" for user, or by the operating system, in which case it is preceded by an "S" for system. The meanings of the codes set by the system are explained in Messages and Codes (GC28-6631). The meanings of completion codes set by problem programs (in this sense the compilers, etc., are problem programs) must be explained in their associated documentation.

11.4 DUMPS OF VARIOUS KINDS AND HOW TO GET THEM

The OS Executive provides the capability of dumping the executing program areas (and relevant portions of the system) upon abnormal termination of a job step. When a dump is taken, it is written on a data set. A DD statement for this data set must be included in the job control statements pertaining to the job step.

There are two kinds of abnormal termination dumps, characterized by the DD names that must appear on the DD statements defining the associated data sets. These are the SYSABEND dump and the SYSUDUMP. The SYSABEND dump routine prints out the system nucleus, the trace tables through supervisor calls, and the contents of the dynamic program area. The SYSUDUMP only prints out the dynamic program area. To insure that the dump card is positioned properly in the deck, the user should precede the SYSABEND or SYSUDUMP ddname by the procedure stepname (where applicable) and place it after any other card inserts for that procedure step, e.g.:

```
//GO.SYSABEND  DD   SYSOUT=A,DCB=(BLKSIZE=7265,LRECL=137,RECFM=VBA)
```

or

```
//GO.SYSUDUMP  DD   SYSOUT=A,DCB=(BLKSIZE=7265,LRECL=137,RECFM=VBA)
```

A SYSUDUMP is sufficient for problem program debugging. The SYSABEND should only be used at the request of a system programmer, as it produces a larger volume of output.

For printing by another JOB, the normal parameters for a NEW data set should be used, but a condition disposition of KEEP should be made, e.g.:

```
DISP=(NEW,,KEEP)
```

so that the data set is not deleted when the step terminates abnormally.

If more than one dump DD statement is used in a job step, only the first one is honored; the rest are ignored. Refer to subsection 21.4 (Dumps) and IBM SRL GC28-6670 (Programmer's Guide to Debugging) for information on interpreting dumps and dump formats.

11.5 CHECKPOINT/RESTART

When a long job terminates abnormally, time is lost if the job must be entirely rerun. In order to minimize the lost time, the OS Executive supports a checkpoint/restart feature which, under program control, dumps the program area and relevant system tables to a checkpoint data set. The job step may be restarted from one of these checkpoints. Alternatively, restart at the step level can be performed, either automatically or at a later time. Automatic step restart can be requested via the RD parameter, in a JOB or EXEC statement. (See Job Control Language User's Guide (GC28-6703) and Job Control Language Reference (GC28-6704) for details.) An automatic restart must be authorized by the computer operator. All data sets in the restart step with the disposition of NEW are deleted; those data sets with the disposition of OLD, MOD, or PASS are kept. When requesting automatic restart, MSGLEVEL=(1,0) or MSGLEVEL=(1,1) must be coded on the JOB statement. Using the RD parameter on the JOB statement causes any RD parameters on the EXEC statements to be ignored. Step names must be unique; if a step that does not have a unique name terminates abnormally, the first step having that name is restarted. The Checkpoint/Restart Manual, GC28-6708, has a list of the completion codes that qualify a step for automatic restart. A deferred step restart can be accomplished by resubmitting the job with the RESTART parameter coded on the JOB statement.

The checkpoint feature is called via the CHKPT macro instruction (ALC), RERUN statement (COBOL), or CALL IHECKPT (PL/I). Restart from a programmed checkpoint can be automatic (using the RD parameter) or deferred. Great care must be taken in the design of programs using the checkpoint feature to insure that the portion of the program between checkpoints is repeatable. This means that input data sets should not be altered; for example, a matrix should not be inverted on top of itself (i.e., a separate array should be used).

Note: As of this printing Checkpoint/Restart does not function correctly under Release 19. Step restart does. Before attempting to use Checkpoint/Restart contact the PAC in building 3, room 133A, on extension 6768.

11.6 ROLL-OUT/ROLL-IN

ROLL-OUT/ROLL-IN is described in Planning for Roll-Out/Roll-In (GC27-6935). This feature is not presently supported at GSFC. Its use requires a cataloged data set on a direct-access device, the inclusion of certain routines in the system nucleus at IPL time, and specification of the ROLL parameter in the JCL.

The use of ROLL-OUT is to the REGION parameter what a secondary allocation is to the SPACE parameter. Its value is that it allows a dynamic program to specify a minimum region size, so that it does not tie up more storage than necessary, and to get more than this minimum when required during execution. Without the ROLL-OUT feature, a job step must request the maximum region size it will use at any time before it can start executing; it cannot obtain more core after it starts than declared in the REGION parameter. ROLL-OUT allows a step to use any unallocated core in the system, regardless of region size. Once all core is allocated, the OS Supervisor will attempt to roll-out a lower priority job, if more core is requested.

11.7 SUPERVISOR PROCEDURES

Several sets of system procedures exist in the PROCLIB. They are executed not by a job, but by a START command issued by the operator. Three of these procedures which directly affect each job run, are known as reader, initiator, and writer procedures. A job is entered into the queue by the reader-interpreter, executed by the initiator, and, if any system output was produced, printed by a writer procedure.

As these procedures are initiated by a START command, the programs they call operate under a system protection key of zero.

11.7.1 READER-INTERPRETER PROCEDURES

The reader-interpreter reads the input stream. The input stream consists of JCL and data. The interpreter converts the JCL to tables and places them in the job queue to form the input queue. The data are spooled as they are read.

The JOB card is converted into a Job Control Table (JCT). Each EXEC card is converted to a Step Control Table (SCT). The JCT points to the first SCT, which points to the next SCT, etc. Each DD card is converted to a Job File Control Block (JFCB). All JFCBs for a step are chained together; the SCT points to the JFCB chain.

A JFCB is also built for each SYSIN data set encountered. For SYSOUT data sets, space is reserved in the output queue, but nothing is actually placed in the output queue at this time.

The reader procedure consists of an EXEC card for the reader program and three DD cards. A parameter containing reader default values is supplied to the reader program. The reader reads the input stream until it reaches an end-of-file, when it terminates. At GSFC, the operators keep the reader active by entering NULL cards between JOBs. Otherwise, they would have to restart the reader for each job.

The parameter supplied to the reader supplies default values for the job steps read by the reader. Those defaults which affect jobs at GSFC are the primary and secondary space allocations for SYSOUT data sets and REGION size. (See Table 11.7-1 below.)

Table 11.8-1. SYSOUT/REGION Space Allocations
(in tracks)

SPACE ALLOCATION		S/95	S/75	S/65
SYSOUT Allocation	Primary	50	20	100
	Secondary	10	100	50
REGION Size		80K	80K	100K

Several other values which are supplied by the reader at other installations are superceded by the job stream manager or the accounting routines at GSFC.

The three data definitions in the reader procedure define the input stream (this is usually a card reader, but can be a tape or direct-access data set), the procedure library, and the SPOOL unit for data in the input stream. Several of these values may be overridden by the operator when he starts the procedure. Since the procedure is simply a named member in SYS1.PROCLIB, several different versions can and do exist.

The DD which defines the PROCLIB can define any library or concatenation. The "DODS-Reader" concatenates DODS.PROCLIB to SYS1.PROCLIB. The S/95 readers concatenate the user PROC Library. Several readers may be active at once, each reading an input stream from a different device.

The DD which defines the SPOOL device has a value for the blocking factor. This may be overridden on the DD card defining input data.

The interpreter scans the JCL. If any format errors are detected, the job is immediately flushed. However, the JCL cards are still scanned for syntax errors. Certain errors in data set and device allocation cannot be determined until the job is initiated. The interpreter reads any procedures referenced, merges the override cards, and makes the required symbolic substitutions.

11.7.2 INITIATOR-TERMINATOR PROCEDURES

The initiator-terminator consists of several modules which perform the tasks of job initiation, step initiation, step termination, and job termination, as necessary. Appropriate exits are taken to the accounting routines and

to device allocation and de-allocation routines. Each initiator is started by the operator to a specific class or classes of jobs (see Table 18.3-2). Input queues (in priority order) for these classes are searched, and jobs initiated.

An initiator requires a 60K region for its own operation. The problem program overlays the initiator when it is loaded. If the problem program requires more core than is available, it must wait until the resource becomes available. Before the advent of "express cancel," a job could not be cancelled until it was initiated. Therefore, jobs which requested a resource that was permanently unavailable (i.e., more core than the machine had or a non-existent I/O device) could not be cancelled, but remained in the WAIT state until the next IPL. A job may have to WAIT on core, I/O units, or data sets (see DISP in Section 17).

The initiator loads the JCT, SCT, JFCB, and other tables from the input queue into the step's region. SPACE for new direct-access data sets is allocated, as well as units (unless DUMMY or DEFER is specified). It is only at step initiation time that an OLD data set (which is not there) or a NEW data set (which is there) are treated as JCL errors. The initiator checks the COND status to see if the step should be executed. If the step passes, the program requested is loaded. Otherwise, and after the main program executes a return, the terminator is called. The terminator disposes of the data sets according to the JFCBs, and puts the system messages on the output queue. This is why programs which incorrectly address core can cause initiators to ABEND. If these tables are incorrect, the terminator can ABEND, and system output from that step is lost, since the output queue will not have been completed.

The terminator determines from the SCT chain whether the job is complete. If so, it queues all the SYSOUT data on the output queue. Space was reserved for these queues by the interpreter. The terminator returns to the initiator, which initiates the next step of the job or the next job from the input queue.

11.7.3 SYSTEM OUTPUT WRITERS

System output writers are started by the operator. They are assigned to an output device and class(es). Jobs in the output queue are selected and printed on a priority basis. The priority is the same as was assigned to the job when it was executed. The operator has the option of raising or lowering the priority of jobs waiting to be printed.

The operator can also assign data sets waiting to be printed to other classes. There are some exceptions, however. For example, RITS output (CLASS=R) and RJE output (CLASS=Z) cannot be changed by the operator while those systems are in operation.

SECTION 12

GRAPHICS

12.1 2250

12.1.1 GENERAL HARDWARE CAPABILITIES

The 2250 display unit basically consists of a CRT screen on which images are displayed under programmed control from a System/360 central processing unit. Optional features enable a 2250 user to enter data into the computer.

Images on the 2250 screen fade rapidly and must be continually regenerated. Regeneration is accomplished by programming (for an unbuffered display unit) or, automatically, by the display control or the buffered 2250, model 1.

The basic 2250 displays graphic information in several formats: as points, as horizontal and vertical vectors of unrestricted length, or as 45 degree vectors of limited length. This information is used to form such displays as characters, graphics, charts, and sketches.

The M&DO IBM System/360 models 95 and 75 are configured with 2250s, model 1. The M&DO IBM System/360 model 65 is configured with 2250s, model 3, with one 2840, model 2. The features included are:

- Absolute vectors
- Light pen
- Character generator
- Alphameric keyboard
- Programmed function keyboard

The absolute vectors feature allows vectors of any length at any angle to be drawn on the 2250 screen.

The light pen feature enables man-machine communication. The light pen is a pen-like device which, when pointed at a portion of the display image, causes an interrupt. By means of a computer program, this interrupt is interpreted (the portion of the displayed image being pointed to is determined) and the appropriate action is taken. Such action, for example, might be the addition, deletion, or rearrangement of displayed data.

The character generator feature enables the 2250 to translate one System/360 eight-bit-byte representation of an alphameric character into a sequence of analog signals which trace the character on the CRT display area. A standard character set of 63 alphabetics, numerics, and special symbols is provided; two character sizes are program-selectable.

The alphameric keyboard feature is a typewriter keyboard. It is another means of man-machine communication. The user may compose messages consisting of letters, numbers, or symbols, or may perform editing functions. If the 2250 is equipped with a character generator, messages are sent to the buffer; otherwise, messages are sent directly to the computer.

The programmed function keyboard feature consists of keys, indicators, and overlay code sensing switches. For each key or overlay code, there is usually an associated computer program (subroutine). When a key is depressed, an interrupt is generated. The interrupt is interpreted by a computer program and control is passed to the associated subroutine for appropriate action.

In addition to the features just described, the 2250, model 3's, and the 2840, model 2, on the IBM System/360, model 65, at GSFC are equipped with the graphic design feature.

The graphic design feature replaces the light pen feature. Both features cannot be on the same 2250. The graphic design feature is available only if the absolute vector feature is included and the 2250 is equipped with an operator control panel. The graphic design feature adds six graphic orders, which provide expanded light-pen capabilities and additional modes for drawing vectors. It provides tracking, sketching, incremental point plotting, and vector drawing capabilities.

The fiber-optic light pen provided with the graphic design feature allows fast detection response for light-pen tracking and similar operations. This pen is equipped with a spring loaded tip switch which replaces the foot switch provided with the normal light-pen feature. Graphic orders added with the graphic design feature can enable or disable light pen detection interrupts, and can permit light pen detection interrupts to occur independently of light pen switch action.

Display (buffer) programs written for the 2250, model 1, equipped with the graphic design feature can be operated in their entirety by a 2250, model 3, and by a 2840, model 2.

12.1.2 GRAPHIC JOB PROCESSOR (GJP)

The graphic job processor is a program which allows the user to pass job control information to the IBM System/360 Operating System from an IBM 2250 Display Unit, model 1 or 3. GJP converts the information to System/360 Job Control Language (JCL).

Some of the functions provided by GJP are as follows:

1. Identification of the user to the system
2. Job step definition
3. Data description
4. Initiation of a job
5. Processing of a named procedure as a job
6. Communication with the system operator
7. Modification of previously completed operations
8. Display of SYSOUT data sets
9. Cancellation of a job

A job (e.g., a graphic program, an assembly, a service program, etc.) can be processed as a foreground or a background job.

A foreground job retains full control of the display unit for the duration of the job. No other control information may be entered until the foreground job is complete. However, the foreground job allows the user to interact with his program.

When a background job is initiated, the user receives no further communication regarding the status of his job. That is, he is unable to interact with his program. However, he may continue to enter control information and initiate additional foreground and/or background jobs.

GJP establishes communication between the operating system and the user by means of displayed frames. A frame is a display requesting response for job control information. The user's response is entered in an area with the light pen, indicated by the cursor, which appears as a short underscore or a rectangular box.

For a complete description of GJP operations, refer to the publication IBM System/360 Operating System, User's Guide for Job Control from the IBM 2250 Display Unit, form GC27-6933.

12.1.3 GTS

Graphics Terminal Services (GTS) provides services which support the IBM 2250 and 2260 display terminals. GTS operates as a problem program

in a region of MVT and requires a region size of 35K for the 2260 and 70K for the 2250. The services provided by GTS allow the graphics terminal user to be largely independent of the normal computer room procedures.

GTS services may be grouped into five major functions which are discussed in the following paragraphs. The five functions are:

1. Log-on/Log-off
2. Data set editing
3. Job scheduling
4. Job output processing
5. Job status

Because of the hardware differences of the 2250 and 2260, these functions have different operational characteristics; however, the functional characteristics are identical.

12.1.3.1 Log-On/Log-Off

The GSFC graphics terminal user must identify himself to the system in much the same manner that a JOB card identifies a job to be run. He must enter his five-character programmer ID and all accounting information normally entered within the parentheses on a JOB card, including the commas and a three-character box number; all information is provided in the same sequence as in the JOB card.

When the user is finished with the terminal, he performs the Log-off operation, as follows, thus clearing his name and account number from the accounting log.

1. To terminate the 2260, the user types in an "F," and an "enter."
2. To terminate the 2250, the ALT code and CANCEL key are used. The user should point the light-pen at the TERMINATE portion of the display, and repeat this operation until the keyboard becomes locked.

12.1.3.2 Data Set Editing

GTS allows the user to create and maintain sequential or partitioned card-image data sets on direct-access files. The data sets may be blocked or unblocked. These data sets can be modified by inserting or deleting card-images or by changing existing cards.

The data set can be considered to consist of pages of card images on a continuous scroll. The user may specify "UP" or "DOWN" to position the scroll to a new page.

To initiate the data set editing feature, the user must specify the DS name, member (if any), and volume serial number of the volume containing the data set.

Any number of data sets can be created and/or edited during any one GTS session.

12.1.3.3 Job Scheduling

Non-graphics jobs submitted through GTS to the 360 computer are treated as any other jobs entering the system. Refer to Job Stream Manager, Section 18.3.

Prior to scheduling, the user sets up his run "deck" by specifying the JCL required. He may reference a cataloged procedure, display the procedure on the screen, and make any changes desired. The JCL edit facility is limited to 100 card-images on the 2250, and 200 card-images on the 2260.

If no procedure exists, the user enters all JCL through the terminal.

For reruns or jobs using similar JCL, the GTS user may "RECALL" the JCL from the previous job and modify that JCL (if necessary) for the current job.

If a graphics job (one that requires use of the terminal to execute) is scheduled, GTS automatically terminates itself and is scheduled to regain control of the terminal when the graphics job terminates.

Selection of the "OTHER" option allows the user to attach a program as a subtask from the GTS monitor. When the task terminates, the completion code is displayed on the screen.

Up to eight parameters may be passed to the attached program. The passing format of these parameters is as follows:

> Register 1 points to a list of ADCONS. The last one has the high-order bit set to 1 (i.e., '80xx xx xx'), which points to an 8-byte area containing the parameter passed. All ADCONS and 8-byte areas are on full word boundaries.

"OTHER" is a GSFC-added option.

12.1.3.4 Job Output Processing

After a job has terminated, the GTS user may view the output data sets and select any or all to be printed by the system printers. The terminal may be manipulated to view any portion of the output data sets.

12.1.3.5 Job Status

This service allows the GTS user to review the status of all the jobs that he has entered into the system from his terminal.

12.1.3.6 References

For further information and assistance with GTS, contact Mr. Frank Ross, Code 543, extension 6796.

12.1.4 GRAPHIC SUBROUTINE PACKAGE (GSP)

The Graphic Subroutine Package (GSP) enables a programmer to create displays (consisting of figures constructed with points, lines, and characters) on one or more IBM 2250 Display Units attached to an IBM System/360 Computing System.

GSP has facilities for data scaling, concurrent display of multiple graphic data sets, scissoring, displaying, and editing of both textural and purely graphic information. It provides basic software support for the handling of light pen, typewriter and function key interrupts.

GSP also enables the user to group graphic display elements (lines, points, words) into nested sequences and to manipulate the resultant sequences as single display elements. All these features make GSP an extremely flexible and powerful graphic display package.

The set of subroutines provided is to be used in conjunction with the FORTRAN, COBOL, or PL/I languages. The execution of each subroutine is requested by using the CALL statement.

Errors that occur while the GSP program is communicating with the 2250 are handled automatically by standard IBM error-handling routines.

GSP may be used with any IBM System/360 Operating System that contains Graphic Programming Services (GPS) for the 2250, with basic attention handling. The 2250s attached to the system may be any combination of models 1 and 3, but must be equipped with absolute vectors and a buffer.

GSP can be used by programs written in the E, G, or H levels of the FORTRAN IV language in COBOL (F), in PL/I (F), or in assembly language.

12.1.4.1 GSP Facilities and Capabilities

GSP provides the following facilities and capabilities:

- Data scaling
- Displaying multiple data sets concurrently
- Scissoring
- Displaying multiple connected line segments, i.e., simple figures with a single subroutine call
- Grouping displays into n nested sequences and manipulating the resultant sequences
- Updating and editing both textual and purely graphic information
- Software support for handling the light pen, typewriter keyboard, and function keys

GSP is an extremely flexible and powerful package and, except for the initialization procedures, is fairly easy to use.

12.1.4.2 Programming Requirements

Preparation of the GSP graphic program requires that the programmer:

1. Establish communication links between the program and GSP (initiation subroutines).
2. Identify the 2250s on which displays are to be produced (initiation subroutines).
3. Define one or more graphic data sets (initiation subroutines).
4. Define the characteristics of the data used to produce the display (option definition subroutines).
5. Create the graphic orders and data necessary for the display (image generation subroutines).
6. Cause the display to be produced on the 2250 screen (image control subroutines).
7. Modify the images making up the display as desired (image control subroutines).
8. Establish communication between the GSP program and the 2250 operator, if desired (attention related subroutines).
9. Terminate the display the use of GSP (termination subroutines).

12.1.4.3 JCL

The GSFC Linkage-Editor procedure LINKGO requires the following JCL for using GSP with FORTRAN:

```
//STEP2              EXEC      LINKGO
//LINK.OBJECT        DD        *
  INCLUDE SYSLIB(IHCGSP03)
```

The GSFC Linkage Editor procedure LINKGO requires the following JCL for using GSP with PL/1:

```
//STEP2              EXEC      LINKGO
//LINK.OBJECT        DD        *
  INCLUDE SYSLIB(IHEGSP01)
  INCLUDE SYSLIB(IHEGSP02)
  INCLUDE SYSLIB(IHEGSP03)
```

12.1.4.4 PL/I Restrictions

The following restrictions are imposed when using GSP on a PL/I program:

1. Program status cannot be checked after a CALL to a GSP subroutine (e.g., return codes).

2. Input to GSP subroutines must be specified as either full word, binary fixed-point data, or binary floating point data.

3. Arguments for GSP subroutines must be scalar constants, variables, or expressions, depending on the requirements for the particular argument.

4. When a structure is used, the qualified name of the element must be passed as the argument in calls to GSP subroutines.

5. Fixed- or variable-length character strings may only be used as the 'text' argument for PTEXT and PLSTR, or as the 'storageloc' argument for GSPRD. The 'count' argument in each CALL must be equal to the current length of the string.

6. All arrays specified in calls to GSP subroutines must be subscripted.

12.1.4.5 References

A detailed description of each of the subroutines is described in the publication IBM System/360 Operating System: Graphic Subroutine Package (GSP) for FORTRAN IV, COBOL, and PL/I, form GC27-6932.

12.1.5 GRAPHIC PROGRAMMING SERVICES (GPS)

The IBM 2250 Graphic Programming Services provide the user with the facilities for developing, both at assembly time and at execution time, the data used to define an image. By means of a graphic access method, GPS also provides the facilities for transmitting the image-defining data to a display unit and for writing routines for man-machine communication. The graphics access method must be included at SYSGEN in order to use GPS. GPS is currently available on all M&DO computers.

GPS is designed primarily for assembly language programmers. It is more difficult to use than GSP, but provides more direct control over display functions. The control blocks for graphic routines must be supplied by the user (they are handled automatically by GSP). Data buffer management and buffer graphic programs are provided by the user by means of assembly language macros.

Assembly initialization and service macro instructions provide a means of controlling the following assembly-time counters:

1. X-coordinate beam-position counter

2. Y-coordinate beam-position counter

3. Buffer-location counter

All of these counters may be initialized and reset as desired by the programmer.

A graphic program consists of a sequence of graphic orders interleaved with data bytes. Graphic orders and data bytes can be created by using order and data generation macro instructions. The graphic order determines the type of operation to be performed (e.g., drawing a dot, a character, or a line). The data bytes are interpreted with respect to the type of operation defined (e.g., coordinates of a point or alphameric characters).

GPS provides a Graphic Data Output Area (GDOA) and its associated Output Area Control Block (OACB) to aid the programmer in handling lengthy or numerous graphic order programs. The GDOA is a programmer-defined area in main storage where graphic orders and data are stored prior to their transmission to a graphic-device buffer. The OACB is defined by the programmer and contains parameters used in controlling the storage of graphic orders and data in the GDOA (e.g., location and size of GDOA, positioning of graphic orders and data in the GDOA, ultimate location of graphic orders and data in the buffer, and the address of the programmer's overflow routine to avoid exceeding the limit of the GDOA).

Problem Oriented Routines (PORs) generate sequences of graphic orders and data, dynamically, at object time. By means of input parameters, a wide variety of images can be produced from graphic order program segments generated by PORs.

When multiple graphic devices share a buffer, the buffer management facilities provided by the graphics access method help optimize the allocation of buffer storage among them. Buffer storage is shared, in the 2840 Display Controller, among two or more 2250 model 2 or 3 display units.

Input/output functions for graphic devices are analogous to those for other input/output devices. The graphics access method provides macro instructions generally similar to those provided by other access methods (e.g., DCB, OPEN, CLOSE, GREAD, GWRITE, GCNTRL, and GREADR). These macro instructions, however, have a special form and somewhat special functions when they apply to graphic devices.

Man-machine communication is made available by attention handling routines which act on interrupts received when the operator depresses a key on the alphameric keyboard or the programmed function keyboard, or by touching a part of the existing display with the light pen. The graphics access method provides two levels of attention handling, "basic" and "express."

The Basic interrupt handler branches directly to the user attention handling routine. This allows immediate action on an interrupt, eliminates the need for polling, and queues interrupts that occur at moments when they cannot be handled. The Express interrupt handler sets flags in control blocks but returns control to the interrupted user program which periodically checks the flags to detect the interrupt.

A detailed description of each of the macros provided by GPS is described in the publication, IBM System/360 Operating System, Graphic Programming Services for IBM 2250 Display Unit, form GC27-6909.

12.1.6 SCOPLT

The OS/360 SCOPLT (Scope Plot) routine provides the capability for users of the IBM 2250, model 1, Graphic Display Device (Scope) to obtain a hard copy of the screen image upon demand. This routine may be used with either on-line or off-line Calcomp plotting systems, including the model 835.

Application programs for the IBM 2250 are usually written in FORTRAN, with the aid of a special graphics subroutine library called GPAK (developed jointly by IBM and the SHARE User's Group). The GPAK subroutines are called by the FORTRAN program to create a sequence of graphic orders and to "write" these into the Scope buffer. The buffer orders then produce the image on the screen, independently of the main computer.

The programmer arbitrarily assigns one of the keys of the Function Keyboard for the purpose of requesting a plot of the screen image. When the key is pressed, the application program calls the SCOPLT routine, which then reads the IBM 2250 buffer into core storage, translates the graphic orders into equivalent plot commands, and returns to the calling program with an error code.

The application program must call PLOTS before calling SCOPLT in order to open the plot tape and initialize the PLOT subroutine with the buffer address and size. Since SCOPLT may be invoked several times, it does not write a "999" block address at the end of each plot. This should be done by the calling program when the Scope user is finished. Since GPAK allows partitioning of the Scope buffer into "areas," each of which may be loaded separately, the SCOPLT routine also permits any number of these buffer areas to be plotted. This is specified in the calling sequence by defining the first and last positions (0-8191) of each buffer area.

12.1.6.1 Plot Generation

The IBM 2250 screen is about 12" square and contains 4096 x 4096 raster units (r.u.). However, coordinate values must be a multiple of four when actually used in a graphic order, so the net resolution is 1024 x 1024. SCOPLT scales the X,Y screen coordinates to page coordinates which results in a 10.24" x 10.24" plot with the same accuracy as the screen-image, and 400 r.u. per inch resolution. The Beam bit (blank/unblank) is recoded to the appropriate plotter pen status code (up or down). The PLOT and SYMBOL routines are called to generate the required plotter commands. (Note: The revised standard SYMBOL routine should be available to assure proper scaling of characters.)

The plot is drawn with "corner lines" to indicate the boundaries. Coordinates that exceed the maximum are indicated by drawing a box symbol just outside the boundary. "Null" characters are considered spaces. Character strings that run outside the plot are not truncated, nor do they "wrap-around" unless a "new-line" character is detected.

Only the Basic graphic character size is currently supported, resulting in a 0.14" character height, with 74 characters per line border-to-border. Screen "points" are drawn as plus (+) symbols to assure ink flow from the pen.

12.1.6.2 Availability and References

SCOPLT resides in the same library (SYS2.GSFCLIB) which contains the PLOT and SYMBOL routines.

For further documentation, contact Mrs. Pat Barnes, extension 6796, in the GSFC Program Library in Building 3.

12.2 2260

12.2.1 GENERAL HARDWARE CAPABILITIES

The IBM System/360, models 95 and 65, at GSFC, is configured with IBM 2260 Display Units, model 1, with an IBM 2848 Control Unit, model 3. The IBM System/360, model 75, at GSFC, is not equipped with any 2260 devices.

This section describes the general capabilities of the 2260, model 1, with a 2848, model 3.

The IBM 2848/2260 provides the ability to access and display computer data conveniently and faster than by more conventional means. It makes the data available as a visual display that can be read directly and is ideally suited for applications that require immediate data acquisition capabilities.

Provision has been made for display station input and inquiry capabilities by including the optional keyboard feature. Input messages generated at the keyboard are displayed on the CRT as they are composed. This permits the user to verify a message before it is transferred.

An inquiry can be quickly entered into the computer, processed, and the desired information displayed on the screen. The displayed data can be analyzed, updated, and returned to the computer for additional processing.

The 2260 Display Unit can operate at distances up to 2000 cable feet from the associated 2848. A maximum of 12 rows, each containing 80 characters, may be displayed on the CRT when using the model 3 2848 control unit.

For additional information, refer to the publication IBM System/360 Component Description: IBM 2260 Display Station, IBM 2848 Display Control, form GA27-2700.

12.2.2 SOFTWARE SUPPORT

Graphic Programming Services (GPS), a set of macro instructions and control routines, may be used with one or more IBM 2260 Display Stations associated with an IBM 2848 Control Unit which is connected directly to the processing unit through either a multiplexor channel or a selector channel.

Input/output control macro instructions are used for: (1) data transmission and control functions, (2)creating Data Control Blocks (DCBs), and (3) establishing and terminating system communication between problem programs and display stations. Input/output control routines create Channel Command Words (CCWs), issue supervisor calls to execute channel programs, and control data transmission between main storage and display station buffer storage.

Interrupts are generated at the keyboard and may be processed either by the "Basic" or "Express" attention handling facilities.

Basic attention handling automatically detects interrupts and routes them to specified user-written routines.

Express attention handling does not automatically detect interrupts. The programmer must periodically check for the occurrence of interrupts; when received, control is transferred to specified user-written routines.

For a detailed description of the GPS macro instructions and control routines, refer to the publication IBM System/360 Operating System, Graphic Programming Services for IBM 2260 Display Station (Local Attachment), form GC27-6912.

12.2.2.1 2260 Subroutine Package

In response to many requests by 2260 terminal users, a set of routines to utilize the 2260 (2260 Subroutine Package) was written by Mr. Frank Ross, Code 543, extending the standard 2260 capabilities. This set of subroutines is primarily designed to be called from FORTRAN programs, but may also be called by assembly programs. They are stored in SYS2.GSFCLIB.

This set allows the caller to open, close, read/write a buffer display, read/write a line, insert a cursor, read keyboard input, specify an attention routine, wait for an interrupt, and delete the routine.

The calling formats, functions, and interdependence follow for each subroutine. The attention handling is basic, and the option of individually addressing multiple units is supported. The entire package uses less than 2K bytes of memory, plus the access methods. All routines require a common (they all share it) seven-word array which is used for communication and return codes.

1. GOPEN (IFT,COMM)

   a. Where: IFT is an integer (I4) variable equal to the FORTRAN unit number describing the 2260 device.

      * COMM is the first word of the seven-word work area common to all graphic subroutines using the device.

*This seven-word array is not to be used by the caller. It can be an integer or real array.

b. Purpose: To open and initialize the graphic device.

c. When called: Called first before any other subroutine (graphic). Only should be called once.

d. Dependence: None, but must be the first subroutine (graphic) called.

2. GCLOS (IFT,COMM)

a. Where: IFT is an integer (I4) variable equal to the FORTRAN unit number describing the 2260 device.

COMM is the first word of the seven-word subroutine work area.

b. Purpose: To close the graphic device.

c. When called: The last graphic subroutine called. Only should be called once.

d. Dependence: "GOPEN" must have been issued before "GCLOS" is called.

3. GSPAR (COMM)

a. Where: COMM is the first word of the seven-word work area.

b. Purpose: To specify and enable the honoring of 2260 interrupts.

c. When called: After "GOPEN" and before "GWAIT", "GCUR","GDAR", or "GCLOS."

d. Dependence: Designed to be used with "GWAIT" or "GCUR." This routine can only be called once unless a "GDAR" has been issued; then another "GSPAR" may be used.

4. GWAIT (COMM[,UNIT])

a. Where: COMM is the first word of the seven-word work area.

UNIT (optional) is an (I4) integer variable which will be filled with the tube number for the interrupt if multiple tubes are being used.

b. Purpose: To place the program in a wait state until a 2260 interrupt is received.

c. When called: After "GOPEN" and "GSPAR". Before "GDAR" and "GCLOS".

d. Dependence: Needs "GSPAR" to set up the routine to take caller out of wait state when interrupt is received. No limit to the number of calls.

5. GDAR (COMM)

a. Where: COMM is the first word of the seven-word work area.

b. Purpose: To delete and disable the honoring of 2260 interrupts.

c. When called: After "GOPEN" and "GSPAR"; before "GCLOS".

d. Dependence: Both "GOPEN" and "GSPAR" must have been called before calling GDAR. Can only be used after a "GSPAR". This routine cancels a "GSPAR," and therefore can be called once for each "GSPAR" called.

6. GICUR (COMM,AREA,TYPE,LINE[,UNIT])

a. Where: COMM is the first word of the seven-word work area.

*AREA is the Dummy Variable (I4,R*4)

*TYPE is the Dummy Variable (I4,R*4)

LINE is the Integer Variable (I4) from 1 to 12 which specifies the line number into which cursor and start symbol are to be placed.

UNIT is the (optional) integer variable (I4) specifying the unit for the opening of multiple units.

*Presently ignored

b. Purpose: To insert a start symbol "→" and cursor into the specified line.

c. Dependence: Must have called "GSPAR" before calling GICUR. Must call "GWAIT" in order that the interrupt be honored from the end-of-keyboard-data interrupt of the data being entered.

d. Return: The fifth word of the seven-word work area contains zero if the operation was successful.

7. GRCUR (COMM,AREA,TYPE,LEN[,UNIT])

a. Where: COMM is same as in "GICUR" and all other routines.

AREA is the first byte of array into which the data will be transferred.

*TYPE is the Dummy Variable (I4).

LEN is the integer variable (I4) specifying the number of characters to transfer.

UNIT (optional) is the same as in "GICUR".

*Presently ignored

b. Purpose: To read and transfer data between the start symbol and the cursor. (Note that the maximum characters per line with start symbol is 79; since start symbol takes one position; start symbol not transferred.)

c. Dependence: Must have previously called "GICUR" and "GSPAR". The fifth word of the seven-word work area contains the following codes and meanings (in I4 format):

=0 Everything correct
=4 Data entered from keyboard less than requested read length
=8 Data entered from keyboard greater than requested read length
=12 Did not call "GICUR" previously to calling "GRCUR"

8. GWBUF (COMM,AREA,KEY[,UNIT]) and GRBUF (COMM,AREA,KEY[,UNIT])

These are the write buffer and read buffer routines, respectively.

a. Where:

COMM is the first word of seven-word work area.

AREA is the first location of array having data to be transferred or to receive data. This array must be 960 bytes long. Data transferred in "A" format (characters).

KEY is the integer variable whose value determines whether keyboard restore is to take place.

=0 with keyboard restore
≠0 without keyboard restore

UNIT is the optional integer variable specifying the unit when opening multiple units.

b. Purpose: To transfer user data (960 bytes) to and from the 2260 buffer. User's array should be in "A" type format.

c. When called: After "GOPEN" and before "GLCOS." No limit to number of calls.

d. Dependence: "GRBUF" should not be called, unless a previous "GWBUF" has been used, since garbage will be read in.

e. Return: The fifth word of the seven-word work area contains zero if the read/write operation was successful. It contains a non-zero value if the read/write failed.

9. GWLIN (COMM,AREA,KEY,LINE[,UNIT]) and GRLIN (COMM,AREA,KEY,LINE[,UNIT])

a. Where:

COMM is the first word of the seven-word work area

AREA is the first byte of the area which contains 80 bytes of data to be transferred or to receive 80 bytes of data ("A" format data).

KEY is the integer variable (I4) whose value determines whether keyboard restore is to take place.

=0 with keyboard restore
≠0 without keyboard restore

LINE is the integer variable (I4) giving line number (1-12) to be acted upon.

UNIT is the optional integer variable specifying the unit when opening multiple units.

b. Purpose: To transfer one line of 80 characters to or from the 2260 device. Data should be in "A" type format.

c. When called: After "GOPEN" and before "GCLOS."

d. Dependence: "GRLIN" should not be called unless the 2260 has had a previous display produced on it.

e. Return: The fifth word of the seven-word work area contains zero if the operation was successful. It contains a non-zero if the read/write failed.

10. GEBUF (COMM[,UNIT])

This subroutine complete erases the 2260 screen.

a. Where: COMM is the first word of the seven-word work area.

UNIT (optional) is the integer variable specifying the unit when multiple units have been opened.

b. Purpose: Erases the 2260 screen.

c. When called: After "GOPEN" but before "GCLOS."

d. Dependence: None

e. Return: The fifth word of the seven-word work area contains zero if successful, non-zero if unsuccessful.

12.3 PLOTTERS

The system library on the model 95 contains software plotting aids callable from FORTRAN programs for the following plotting devices:

1. Stromberg-Carlson 4020 Plotter
2. Stromberg-Datagraphics 4060 Plotter
3. CalComp 570, 770, and 780 Plotters
4. On-line printer

12.3.1 STROMBERG-CARLSON 4020 PLOTTER (SC-4020)

This is a cathode ray microfilm plotter with a plotting area of 7" by 7" (hard copy) or 35 mm microfilm.

When generating the plot tape, the following JCL cards are required to include the plot routines from the system library and to define the output tape using FT10:

```
//PLOT                    EXEC   LINKGO
//LINK.SYSLIB       DD         DSN=SYS2.SC4020,DISP=SHR
//GO.FT10F001       DD         DSN=useridxxx,UNIT=7TRACK,
//                             LABEL=(,BLP),DISP=(NEW,KEEP),
//                             VOLUME=SER=xxxxxx,DCB=(BLKSIZE=1200,
//                             TRTCH=C,DEN=1)
```

where:

userid = the programmer's regular five-character ID, plus three characters of his choosing.

VOL=SER=xxxxxx: For xxxxxx, the user substitutes his own tape number.

SC-4020 may also be executed using the LOADER procedure.

12.3.1.1 References

For more detailed information about the SC-4020 plot routine, see the SC-4020 Microfilm Recorder User's Manual, prepared by Computer Sciences Corporation under NASA contract NAS5-9758.

The SC-4020 plotter is no longer available at the GSFC. The Stromberg Datagraphics 4060 (paragraph 12.3.2) has a 4020 simulator and for a limited time tapes generated for the 4020 will be processed on the SD-4060. Users are urged to convert their programs for direct processing by the SD-4060.

12.3.2 STROMBERG-DATAGRAPHICS 4060 PLOTTER

The Stromberg-Datagraphics 4060 plotter was formerly called the Stromberg-Carlson 4060 plotter; the plot routines' names currently reflect the former corporation name, i.e., SYS2.SC4060.

This plotter is located in Building 23, and a generated plot tape for the purpose of obtaining hard copy or microfilm may be submitted at the Univac 1108 dispatcher's desk there.

When generating the plot tape, the following JCL cards are required to include the plot routines from the system library and to define the output tape on SC4060ZZ.

```
//PLOT                  EXEC   LINKGO
//LINK.SYSLIB       DD         DSN=SYS2.SC4060,DISP=SHR
//GO.SC4060ZZ       DD         DSN=useidxxx,UNIT=7TRACK,
//                             LABEL=(1,BLP),DISP=(NEW,KEEP),
//                             VOL=SER=xxxxxx,DCB=(DEN=1,TRTCH=C,
//                             RECFM=F,BLKSIZE=1024)
```

Where:

userid is the programmer's regular five-character ID, plus three characters of his choosing.

xxxxxx is replaced by the user's tape number.

SC4060 may also be executed using the LOADER procedure.

12.3.2.1 P360 - Output To Microfilm

P360 is an SD-4060 processor that allows the user to process a 7-track tape generated on the IBM 360 series of computers. The output is 16-mm positive-appearing microfilm. Both standard ASA (American Standard Association) and machine codes for the IBM 1403 printer carriage control characters are accepted. A print list (print all characters, i.e., no control characters) is included.

P360 restrictions are as follows:

1. The processor is currently limited to 7-track, 556-cpi tapes.

2. Standard OS tape labels are required.

3. Any BLKSIZE, RECFM, and LRECL is valid provided the BLKSIZE is less than or equal to 2560 bytes. Since most microfilm applications entail large amounts of output, you should normally use the largest blocking factor within the above restrictions.

4. If you use RECFM=V, you must use TRTCH=C. When TRTCH=C is specified, the largest blocksize allowable is BLKSIZE=1920.

TRTCH=ET gives invalid output for special characters such as (,), &, and =. In addition, the carriage control character "+" will not work properly if you use TRTCH=ET.

Example of overriding the normal SYSOUT writer output (assuming the DCB parameters are compatible with your program).

```
//GO.FT06F001  DD   UNIT=2400-7,DISP=(,KEEP),LABEL=(1,SL),
//                  DCB=(BLKSIZE=1920,LRECL=137,RECFM=VBA,TRTCH=C,DEN=1),
//                  DSN=--------,VOL=SER=------
```

Tapes to be processed should be brought to the SD-4060 dispatcher in building 23, room 119. Care should be taken when filling out the instruction card to ensure that the name P360 is written under "Scrip Processor." For "number of frames" of microfilm output, insert the number of pages of printed output usually received.

12.3.2.2 References

Refer to the Universal SD-4060 System and Software Manual, dated June 22, 1970. This manual may be obtained from either Mr. Donald Kennedy, Building 23, Room W105, extension 6346, or Mr. George Fleming, Building 23, Room W103, extension 5129.

12.3.3 CALCOMP 570 PLOTTER

The CalComp 570 plotter is a drum plotter with a plotting surface of 11" x 120". The user produces a magnetic tape, to use for off-line plotting in the S/360-91 computer room in the basement of Building 1.

When generating the plot tape, the following JCL cards are required to include the plot routines from the library and to define the output tape:

```
//PLOT                EXEC   LINKGO
//LINK.SYSLIB      DD        DSNAME=SYS2.CAL570,DISP=SHR
//GO.PLOTAPE       DD        DSN=useidxxx,UNIT=7TRACK,
//                           LABEL=(,BLP),VOL=SER=xxxxxx,DCB=DEN=0
```

Where:

userid is the programmer's regular five-character ID, plus three characters of his choosing.

VOL=SER=xxxxxx: For xxxxxx, the user substitutes his own tape number. The plot tape will be a 7-track tape recorded at 200 BPI. There is no operator at the plotter, but an operating instruction manual is there. For additional assistance, see personnel in Building 1, Room 159, extension 5436.

12.3.3.1 References

Refer to the CalComp Digital Recorder User's Manual, prepared by Computer Sciences Corporation, dated January 1967, with Update Packages A, B, C, and D, for further information on all CalComp plotters.

12.3.4 CALCOMP 770/780 PLOTTING SYSTEM

The CalComp 770/780 plotting system is a drum plotter with a plotting surface of 30" x 120". The user produces a magnetic tape and takes it for off-line plotting to Building 21, Room 273, (extension 6277).

The 770 plot routines are described in the CalComp Digital Recorder User's Manual. The 780 routines are a later, improved version from CalComp, for which no documentation is generally available.

When generating the CalComp 770/780 plot tapes, the following JCL is required:

For CalComp 770 plot tapes, use:

```
//PLOT              EXEC   LINKGO
//LINK.SYSLIB  DD         DSNAME=SYS2.CAL770,DISP=SHR
//GO.PLOTAPE   DD         DSN=useidxxx,UNIT=7TRACK,
//                        LABEL=(,BLP),VOL=SER=xxxxxx,DCB=DEN=1
```

For CalComp 780 plot tapes, use:

```
//PLOT              EXEC   LINKGO
//LINK.SYSLIB  DD         DSNAME=SYS2.CAL780,DISP=SHR
//GO.PLOTTAPE  DD         DSN=useidxxx,UNIT=7TRACK,
//                        LABEL=(,BLP),VOL=SER=xxxxxx,DCB=DEN=1
```

Where:

userid is the programmer's regular five-character ID, plus three characters of his choosing.

VOL=SER=xxxxxx: For xxxxxx, the user substitutes his own tape number. The plot tape generated will be a 7-track tape at 556 BPI.

Note: Notice the different spelling in the DD cards above: PLOTAPE for the 770 routine, but PLOTTAPE for the 780 routine.

12.3.4.1 References

Refer to the CalComp Digital Recorder User's Manual for further documentation.

12.3.5 CPLOT PROGRAM

This is a set of subroutines, simple to use, designed to aid in plotting on the CalComp 570. (The CalComp 770 may be used as the output device, but only 10 inches of the 30-inch plotting area will be used.) The routines automatically perform scaling and labeling to produce plots on a user-defined grid.

The CPLOT routines are contained in the standard FORTRAN library, FORTLIB, but the user must also include the LINK.SYSLIB and the DD control cards described for either the CalComp 570 or the CalComp 770 subroutines. For example, in addition to using the applicable calling sequences for the CPLOT routines, the user must add the following JCL cards:

```
//PLOT                EXEC   LINKGO
//LINK.SYSLIB    DD        DSNAME=SYS2.CALnnn,DISP=SHR
//GO.PLOTAPE     DD        DSN=useidxxx,UNIT=7TRACK,
//                         LABEL=(,BLP),VOL=SER=xxxxxx,DCB=DEN=n
```

Where:

CALnnn is CAL570 or CAL770
xxxxxx is replaced by the users tape number
DEN=n - n is replaced by 0 for 200 BPI, for the CAL570 routines;
by 1, for 550 BPI for the CAL770 routines

12.3.5.1 References

For further documentation, contact Mrs. Pat Barnes in the GSFC Program Library, Building 3, extension 6796.

Refer to the memorandum from Mr. D. Y. Sumida, Mathematics and Computing Branch, Subject: CPLOT, dated September 1, 1967, and a manual titled CPLOT CalComp Plotter Routine, Direct-Couple System for FORTRAN II, FORTRAN IV, Theoretical Division.

12.3.6 PRPLOT PROGRAM

The PRPLOT program is a S/360 version of the 7090/7094 UMPLOT routines. This is a set of subroutines designed to plot on the standard computer printer (SYSOUT=A). It is callable from FORTRAN and requires no DD or SYSLIB cards.

The output and usage are similar to CPLOT, to allow the user to check out his program with quick turnaround before using a plot tape.

12.3.6.1 References

For documentation on PRPLOT, refer to the Laboratory for Theoretical Studies, System 360, Bulletin No. 5, from P. Smidinger, dated September 26, 1966, Subject: PRPLOT, A 360 Printer Plotting Program, with attachment from the University of Michigan Computing Center, dated March 1, 1961, on UMPLOT subroutine.

12.3.7 GERBER VP822

The Gerber VP822 plotter routines are not available on the M&DO computers. They are contained on the system libraries of the SESD S/360 computers. The Gerber VP822 plotter is a flat-bed plotter with a plotting surface of 50" x 60". The generated tape can be plotted in Building 11, Room S-26. The following JCL cards are required to include the plot routines from the system library and to define the output tape:

```
//PLOT                  EXEC   LINKGO
//LINK.SYSLIB       DD        DSN=SYS2.GERBER,DISP=SHR
//GO.GERBER         DD        UNIT=7TRACK,LABEL=(,BLP),VOLUME=SER=xxxxxx
```

The plot tape will be 7-track, at 556 BPI, and in even parity.

SECTION 13

REMOTE JOB ENTRY

13.1 GENERAL DISCUSSION

13.1.1 NATURE OF REMOTE JOB ENTRY (RJE)

The Remote Job Entry (RJE) facility of the operating system provides, for users of the IBM 360/95, an efficient and convenient method for entering jobs submitted from remote work stations into the job stream. Once a job has been entered into the job stream by RJE, execution of the job proceeds under the supervision of the operating system job management routines. All data sets created by the job are handled by the operating system data management routines. Output data sets that have been created by remotely submitted jobs and that are to be returned to the remote user are placed in a separate output class. These data sets are removed from this output class and returned to the remote user under the direction of the RJE program. This type of operation provides a remote user with the same batch-computing facility that is available at the central installation.

13.1.2 RJE FACILITIES

13.1.2.1 Hardware

For M&DO users, the central computer is the IBM 360/95, equipped with a 2703 Transmission Control Unit. The remote terminals are IBM 2780 Data Transmission Terminals, each containing a card reader, a punch unit (with one exception) and a line printer. The terminals are connected to the central system either by leased-line or dial-up data phone units. A box containing a red and a green light has been installed in the immediate vicinity of each terminal. The green light when on indicates that it is permissible to submit and receive work through the RJE terminal. The red light on is an indication that no work should be submitted from the terminal.

13.1.2.2 The RJE System

The RJE user communicates with the RJE system (actually a high priority task running on the 360/95, using 150K of core) through the use of work station commands. Work station commands can be placed between jobs in the input stream; however, they must not be placed within the physical limits of a job.

Through the use of work station commands:

a. The user can specify that the job output be returned immediately or deferred until requested. In the event of a cold start, all RJE jobs currently in the system will be lost. The operator will attempt to notify the user through dispatch by returning a card containing the job name and a message, "LOST IN IPL." A warm start will cause the remaining steps in the job to be flushed. No completion code or other indication will be provided. A warm start should be suspected when the job stops executing for no apparent reason after running normally for awhile.

b. There are two sets of ID's established for the terminal and user. The first is established with the RJSTART command and is the Terminal ID.

```
.. RJSTART GSFC#5
```

The second set (id, password) is established with the LOGON command and is the userid. At GSFC, each terminal has a unique dedicated userid (see 13.1.3).

```
.. LOGON       GA5,KEY
```

In this case, the id GA5 is the userid corresponding to terminal GSFC#5. No terminal id or userid should ever be used unless it is properly assigned to the user's terminal. The user can direct job output to his userid, to an alternate userid, or to the control system output devices. Only the userid used when submitting the job, or an alternate userid specified by the submitter, can be used to remotely receive job output.

c. The user can request notification of job completion, including an indication of normal or abnormal termination.

d. The remote operator can discontinue in-process printing to read in cards and continue the printing at a later time by command. Some duplication may appear when printing resumes, since it starts at the beginning of the buffer. Messages pertaining to the cards being fed in will be printed at the conclusion of the interrupted printout, except for error messages, which are printed out immediately.

e. Should other than the standard forms be required, the user can specify the requested form numbers. The RJE system will automatically discontinue output and send a message to the remote operator, who may continue output after satisfying the form change request.

Work station commands precede the job to which they apply. If a work station command is received in the middle of a job, it will be processed as part of the job, not as an RJE command. The commands are of the form:

		COLUMNS		
1 & 2	3	4-71	72	73-80
..	BLANK	COMMAND	*	Sequence number (Optional)

*A non-blank character in column 72 is required to continue a command. The continued command must have periods in columns 1 and 2 and be blank from columns 3 to 15. The operand being continued must begin in column 16, or it will be treated as a comment. (The JED command is the only one which may be continued.)

Table 13.1 illustrates the available commands and their usage.

RJE COMMANDS

	Command	Operands	Function	Example
1.	RJSTART	Terminal id	Attaches and identifies a work station to the system. This must be the first command received from an inactive (not attached) work station.	.. RJSTART GSFC#1 This will attach station #1 to the RJE system.
		,BRDCST=YES		.. RJSTART GSFC#1,BRDCST=YES This will print messages of system interest on the remote printer after attaching the work station to the RJE system.
		,BRDCST=NO		.. RJSTART GSFC#1,BRDCST=NO This is the default value and is the same as .. RJSTART GSFC#1.
2.	RJEND	(no operand)	Detaches a work station. All messages directed to the work station are transmitted before the station is logically detached. No more job output will be returned to the work station after the RJEND command is accepted.	.. RJEND This will detach a work station from RJE.
3.	LOGON	userid,key	Begins a user session. As currently set up at GSFC, there is usually only one userid, e.g., GA1, per work station. 'Key' for all GSFC stations is actually the word KEY.	.. LOGON GA1,KEY User #1 is ready to begin a session. A user cannot be logged on at more than one work station at a time. If a user desires to change work stations, he must log off at his old work station before logging on at the new work station.

	Command	Operands	Function	Example
4.	LOGOFF	(no operand)	Completes a user session.	.. LOGOFF A user has completed a session. No more work will be accepted from the station until another LOGON command is entered. However, immediate output will still be directed to the work station.
5.	OUTPUT	J=jobname	Requests specific job output.	.. OUTPUT J=ABCDE001 Print and punch outputs from job ABCDE001 are returned to the terminal from which they were submitted if the user originated it or if the user has been named as a valid recipient of the output.
		U=userid	Requests all deferred output for an id.	.. OUTPUT U=GA1 All deferred jobs for which GA1 is a valid recipient are returned. This is valuable for jobs requiring punch output.
		*	Requests for all jobs naming the current id as a valid recipient.	.. OUTPUT * Jobs submitted under other id's naming the current id as a valid recipient of output will be returned. The user may route a job to another work station and obtain the output with this command.
		(no operand)	This is identical to U=userid.	.. OUTPUT All deferred jobs submitted by the current id will be returned.

	Command	Operands	Function	Example
6.	CONTINUE		Requests discontinued output. Output may have been discontinued because of a forms change, operator intervention, or equipment failure during an output operation. When interrupted output is being held for a work station, no output will be returned to the work station until a CONTINUE command is received. However, RJE will continue to accept work from the station. The full JCL is always returned to the RJE terminal (see bottom of page 13.2-2).	
		BEGIN	Retransmits entire data set.	.. CONTINUE BEGIN Data set is retransmitted from beginning - this command is used when there is a paper problem.
		NO	Discontinues printing data set. Method: Press STOP, then START on the printer. TERM ADDR light will come on. Turn dial to OFFLINE and then back to TSM/TRSP. Ready the printer. Place a CONTINUE NO card in the card reader, ready the EOF button, and press START to read the card. The alarm will sound until the printer resumes printing.	.. CONTINUE NO The current data set is lost. One CONTINUE NO command is required for each data set in a job if the printing is to be suppressed. This will allow suppressing a dump (accidentally set remote) while still obtaining the remaining output. JCL cannot be discontinued through the use of the CONTINUE NO command.

	Command	Operands	Function	Example
6.	CONTINUE (Cont'd)	(no operand)	Continues interrupted output.	.. CONTINUE This card may be used after an equipment failure is corrected (fix the paper or ready the punch). With blocked records, there is a possibility of duplicate output, since transmission is resumed at the beginning of a block.
7.	DELETE	jobname J=jobname	Deletes the job name from the system if the user (id) is identical to the one submitting the job. All output directed to the terminal is deleted. If the job has already run, permanent data sets created by the job remain intact. Output to the central computer will be printed.	.. DELETE ABCDE001 The job ABCDE001 is deleted when the message, "IHK107IJOB DELETED ABCDE001 GA1" is returned. The user may not re-use the job name until the message is returned.
		(no operand)	Removes from the system all jobs under the current LOGON id.	.. DELETE This is used (very cautiously) to delete all jobs in the current user stream when the possibility of duplicate jobnames must be avoided.
8.	ALERT		Requests that the system notify the user that his deferred output is ready.	

	Command	Operands	Function	Example
8.	ALERT (Cont'd)	J=jobname jobname	Notifies a specific completion.	.. ALERT ABCDE001 The command remains pending until job ABCDE001 is completed. If the job is not in the system, a message is returned to that effect and the command is rejected.
		*	Notifies any job for which the current userid is a valid recipient. As currently set up at GSFC, there is usually only one userid, e.g., GA1, per work station.	.. ALERT * This command is useful when the user expects job output from another user and wishes to be notified when it is available.
		/	Cancels all pending ALERT commands issued by this work station.	.. ALERT / There is no selective canceling of ALERT commands.
		(no operand)	Alerts information returned for all user jobs.	.. ALERT This is similar to a status command in that a message is printed when a job finishes.
9.	STATUS	J=jobname	Returns the status of a job.	.. STATUS J=ABCDE001 Returns the status of ABCDE001.
		U=userid or (no operand)	Specifies all current jobs submitted by id. No operand treated as current id.	.. STATUS GA1 or .. STATUS All jobs submitted by GA1 or the current user are displayed on the printer.

	Command	Operands	Function	Example
9.	STATUS[1] (Cont'd)	*	Specifies all jobs for which the current user is a valid recipient, e.g., at GSFC it could mean anyone who had submitted a job from a particular terminal.	.. STATUS * A user may wish to know of jobs sent to him by other users, as well as his own jobs.
		T	Specifies all jobs submitted by a work station.	.. STATUS T A work station may use this command just before initiating closedown to determine if any job is waiting to print.
10.	BRDCSTR	none	Returns a copy of broadcast messages sent by the central operator.	.. BRDCSTR Messages such as the next scheduled closedown may be returned.
11.	MSGR	M='text'	Sends a message of 40 or fewer characters to the operator if the T and U parameters are omitted.	.. M='MESSAGE' The word 'MESSAGE' and the userid are displayed to the central operator or the user specified by T or U.

1 If the STATUS response indicates that the job is not in the system, the following possibilities may have occurred:

a. Immediate output - The output may have already been returned.
b. Deferred output with alternate recipient - Alternate has retrieved output using an OUTPUT command.
c. Shared userid - Other persons sharing userid may have received output.
d. Job may have been deleted by user or another user sharing that userid.
e. Central operator retrieved the job with a CENOUT command.
f. A cold start may have occurred.

Command	Operands	Function	Example
11. MSGR (Cont'd)	U=userid	Routes the message to the user-id named, if he is logged on.	.. MSGR M='MESSAGE',U=GA1
	T=termid	Sends a message to the terminal specified. The message is held until the terminal is active.	.. MSGR M='MESSAGE',T=GSFC#1 The word 'MESSAGE' is printed on terminal #1 when it comes on line. U and T can be used together to send a message to a special user of a terminal if he is logged on.
12. JED		The JED statement is not required as part of the job entry. If it is omitted, the following system defaults are assumed: 1. Immediate output. 2. No notification of job completion. 3. All output returned to user. The JED statement is the only control statement which may be continued. (See page 13.1-2 for rules on continuation.) If the JED statement contains syntax errors, the statement will be rejected; however, the job is accepted and processed, using the above-mentioned defaults.	
	OUTPUT=IMMED	The default option requests that the output go to the submitter when the job is finished.	.. JED OUTPUT=IMMED

	Command	Operands	Function	Example
12.	JED	OUTPUT=DEFER	Holds output until requested by an OUTPUT card.	.. JED OUTPUT=DEFER This is useful for punch jobs or jobs requiring form changes. This is especially useful when a dump is expected. The user can get the beginning of the dump and cancel the remaining section with a CONTINUE NO card.
		OUTPUT= (DEFER, userid)	Defers output, and both the submitter and the alternate recipient may request the only copy of the output.	.. JED OUTPUT=(DEFER,GA1) The original submitter or the id GA1, may obtain the output.
		NOTIFY=SOURCE	Notifies the original submitter of job completion.	.. JED NOTIFY=SOURCE
		NOTIFY=BOTH	Notifies both the original submitter and the alternate named in the OUTPUT parameter when the job has been completed.	.. JED NOTIFY=BOTH
		NOTIFY= (SOURCE, 'text')	Presents the original submitter with the text, limited to 25 characters, when the job finishes.	.. JED NOTIFY=(SOURCE,'CALL DOE, 9999,FOR PUNCH.') This is useful when punch output is expected, and the user wishes to be present to claim his deck.
		NOTIFY=(BOTH, 'text')	Presents the original and an alternate named on the OUTPUT card with the text, limited to 25 characters.	.. NOTIFY=(BOTH,'MESSAGE HERE.')

	Command	Operands	Function	Example
12.	JED (Cont'd)	CENTRAL=ALL	Returns the JCL for the particular job to the remote printer. All other SYSOUT data sets (as specified on the job's DD cards), including punched cards, are sent to the central system output devices.	.. JED CENTRAL=ALL The user is encouraged to use MSGLEVEL=(2,0) on the job card, whenever possible, to avoid long JCL expansions on the remote printer.
		CENTRAL= (stepname. ddname,stepname.ddname)	Sends the named data sets to the central devices. All other SYSOUT data sets are returned to the remote terminal.	.. JED CENTRAL=(SOURCE.SYSPRINT, GO.SYSUDUMP) This will send the printed output of the compiler and a dump, if produced, to the central printer. The program output will still return to the remote printer.

NOTES:

- Do not stack jobs if the second, third, etc., job has a JED card. This, on occasion, will cause all but the first JED to fail. To feed in multiple jobs containing JED statements, feed them one at a time and only after the message "IHK117I JOB ACCEPTED jobid term userid.....DEFAULT or JED" is returned.
- Jobs requiring large volumes of punched or printed output should be sent via central site (.. JED CENTRAL=ALL).
- If the terminal does not have a punch unit, the CENTRAL= parameter must be used to route all punch output (SYSOUT=B) to the central punch. Otherwise, a term address error (see error procedures) will occur when the job needs a punch.
- On dial-up units (which are disconnected automatically by the RJE system after several minutes of disuse), OUTPUT=DEFER is recommended to prevent remote output from interfering with other users' input operations when they log on.

13.1.3 LOCATIONS OF RJE TERMINALS

There are currently six RJE Terminals connected to the IBM 360/95. These terminals are installed in the following locations:

Location	Room	Work Station or Termid	Userid	Person in Charge	Extension
NASA/GSFC Bldg. 23	C108	GSFC#1	GA1	Mr. Charles Newman	5666
NASA/GSFC Bldg. 18	111	GSFC#2	GA2	Mr. Robert Laukaitis	4732
NASA/GSFC Bldg. 17	S43	GSFC#3	GA3	Mr. Walt Dennison	6506
NASA/GSFC Bldg. 7*	235	GSFC#4	GA4	Mr. Reg Mitchell	5549
Computer Sciences Corporation Silver Spring	625	GSFC#5	GA5 GB5**	Mr. Laurel Joyce	589-1545 X395
NASA/GSFC Bldg. 11	S125	GSFC#6	GA6	Mrs. Marjorie Johns	6544

*Dial-up data phone units - current phone number of the 360/95 is 474-5230.

**The userid GB5 is used by GSFC#5 at night.

All locations are equipped with the IBM 2780 Data Transmission Terminal, Model 2, except for the GSFC location in Building 7, which contains a DATA 100 terminal, which has card read and print capabilities. The 2780-2 has card read, punch, and print capabilities.

13.1.4 COMPUTERS SUPPORTING RJE

The only M&DO computer supporting RJE is the 360/95. Refer to paragraph 2.3.2 of this User's Guide for job submission procedures.

The SESD Model 91K (see paragraph 2.4.1) also supports RJE, but has only one terminal, the IBM 1130, available for use.

13.1.5 TAPE MOUNTS

Refer to paragraph 2.3.4 of this User's Guide for a discussion of the policy and procedures pertaining to the use of tapes in RJE submitted jobs.

13.1.6 POLICY AND RESTRICTIONS

Anyone having a valid programmer id and sponsor number (see paragraphs 2.1.2 and 2.1.3) is authorized to use the RJE facilities. During daytime operations (8 A.M. to 8 P.M.), however, programs with a CPU or I/O time exceeding 11 minutes, or which use more than 700K bytes of memory, will fail with the message:

"***JOB FAILED FOR EXCEEDING CORE SIZE LIMIT."

13.1.7 REFERENCES

1. IBM System/360 Operating System Remote Job Entry, Form GC30-2006.

2. IBM 2780 Data Transmission Terminal - Component Description, Form GA27-3005.

13.2 OPERATING THE RJE TERMINAL

13.2.1 OPERATING GUIDELINES

The following guidelines are presented to enable the user to operate and submit jobs to the 360/95 via RJE.

To ready the IBM 2780 Transmission Terminal for operation, the user must:

a. Turn on the power switch located on the right side of the card reader. This provides power to the reader and printer.

b. Depress the NPRO key to remove cards that may be in the reader.

c. Set dial (rotary mode switch) to TSM/TRSP.

d. Depress the START key on the printer.
Dial-up only: Call computer (tel.no.: 474-5230) with data phone in talk position. After hearing high pitched tone, punch VBDATA button and hang up phone.

e. Log on to the computer by placing the RJSTART and LOGON cards into the card hopper.

f. Depress EOF key. END OF FILE and DATA SET READY lamps will then be lit. If END OF FILE lamp does not light up and the AUTO TURN-AROUND light is on, turn the AUTO-TURNAROUND light off by depressing the button, and then depress the EOF key once again.

g. Depress and hold START key to ready the card reader. The message "RJSTART ACCEPTED USER LOGGED ON GAx" will be displayed on the printer as soon as the computer accepts the log on. The message is not readily visible, however. Hit CARRIAGE STOP and CARRIAGE SPACE key several times to display the message. Push the START on the printer to ready the printer.

h. Silence the operator attention alarm, which will sound to notify the user that the system is ready to accept cards. To silence the alarm, push STOP on the card reader and START on the printer.

i. Put cards in hopper and depress EOF key.

j. Depress and hold START key on reader to ready reader and read cards. (Key must be held down until several cards have fed in.)

k. When the alarm sounds, indicating the job has been read in, and a message indicating the job has been accepted is printed, hit STOP and CARRIAGE SPACE on the printer to silence the alarm; view the message, and hit START on the printer.

To close down a terminal:

a. Read LOGOFF and RJEND cards into the system.

b. Wait for acceptance message to come out through the printer.

c. Turn the power switch to the POWER OFF position.

Once the terminal is logged on, submit jobs exactly as to the dispatcher; i.e., the JCL and other cards in the deck are identical. To submit a job:

a. Depress the NPRO key to remove any cards that may be in the reader.

b. Straighten deck and place face down with 9 edge in the back of the card hopper, and replace the lid on the deck with metal edge forward.

c. Rotate the rotary dial (mode switch) to reset terminal, if necessary. Set the mode switch to TSM/TRSP before reading cards.

d. Press END OF FILE key to light END OF FILE light.

e. Press START key on printer to ready printer.

f. Press and hold START key on reader until READY light comes on.

After the deck has been read in, a message will be displayed on the printer:

"IHK117I JOB ACCEPTED jobname GAx SCHED nnnn DEFAULT"
JED

where:

- jobname is the job name from the JOB card
- GAx is the terminal designation
- nnnn designates the order of execution scheduled
- DEFAULT indicates that the printout will be directed to the terminal as soon as it is ready (OUTPUT=IMMED)
- JED indicates that a JED card preceded the JOB card - output will be handled as directed by the JED commands

If the printout is long, or if punch output is expected, refer to paragraph 13.1.2 of this User's Guide for examples of how to route part or all of the printout to the system printers.

RJE will always return the JCL to the local user. To avoid long JCL expansion on commonly used procedures, code the parameter:

MSGLEVEL=(2,0)

after the accounting information on the job card. This will cause the system to print only the JCL cards supplied by the user. The printout starting with "XX" and all of the allocation messages will be eliminated.

13.2.2 PUNCHED OUTPUT (MODEL 2 ONLY)

Since the 2780-2 has just one card feed which functions as both reader and punch, the system does not begin punching whenever it has punch output. The TERM ADDR lamp will light to indicate an "error" condition which must be cleared. To punch data:

a. Clear the error condition by rotating the dial from TSM/TRSP to off-line and back to TSM/TRSP. This will also place the printer in a not ready state.

b. Ready the printer. (Push the START key on the printer.)

c. Load a CONTINUE card and sufficient blank cards into the reader.

d. Press the END OF FILE and AUTO-TURNAROUND keys on the reader/punch unit.

e. Press the START key on the reader until the READY light comes on. The read/punch unit may pause for a few seconds before it starts punching cards.

f. After the punching stops, and the program continues printing, press the AUTO-TURNAROUND key on the reader to turn out the light, and then remove the excess blank cards from the reader hopper. Press the NPRO key on the reader/punch to remove the two cards remaining in the unit.

13.2.3 OPERATOR ATTENTION ALARM

Unless output has been deferred (see paragraph 13.1.2 for examples of deferred output), the job will start to print out as soon as it has run and the terminal is in a ready state. When output is completed, or a condition exists that requires an operation such as punching cards, an alarm will sound. If no error conditions exist, remove the output from the printer by pressing CARRIAGE STOP and CARRIAGE RESTORE. Ready the printer and the terminal. If the alarm sounds and error lights are on, see paragraph 13.2.4 of this guide.

13.2.4 ERROR PROCEDURES

The following is a list of procedures to be followed if an error lamp is lit on the card reader:

HOPR -

a. Remove card deck.

b. Depress the NPRO key.

c. Replace deck in card hopper.

d. Depress the EOF key.

e. Depress and hold START key to ready the reader.

LINE - There may be several reasons for this lamp to be lit:

a. The 360 system is inoperative.

b. The RJE system is inoperative.

c. The system is overloaded and will not accept any additional jobs.

Correction procedures for any of the above errors are as follows:

- Wait a few seconds to determine if the system will return to normal operations.

- Remove card deck, depress the NPRO key, read in RJSTART and LOGON cards, and wait for an error message to be printed.

- Call the 360 computer operator (tel. no.: 982-5395) to determine if the system is inoperative.

If the job card is read in, the name on the job card must be changed since the system will not duplicate jobnames, e.g., change ABCDE001 to ABCDE002.

INCP and EQUIP CHK -

a. Remove the card deck from the hopper.

b. Depress the NPRO key - two cards will come out of the card reader.

c. Place the two cards in the card hopper with the remainder of the job.

d. Depress the EOF key.

e. Depress and hold the START key to ready the reader. The INCP light will come on if the user and terminal are not logically connected to RJE. In this case, RJSTART and LOGON cards must be entered before further input can take place.

PUNCH STA -

When this lamp is illuminated, a card jam has occurred. If an experienced operator is available, he will be capable of removing the card jam. In the event that it is necessary for the programmer to remove the damaged cards, the following steps should be taken:

a. Clear the top of the card reader of all card decks, pencils, and other items.

b. Standing at the back of the equipment, open the right-hand side cover and then the rear cover.

c. Locate the jammed card under the card guide of the read or punch station.

d. Lift the two clear card guides.

e. Pull black spring and lift steel photocell holder guide.

f. Remove cards from read station.

g. To remove jam in punch station:

1. Leave machine power on.

2. Rotate hand wheel clockwise 1/2 revolution.

3. Depress spoon-shaped lever and remove card by hand.

h. Close steel photocell holder and two plastic card guides.

EQUIP CHECK -

a. Press STOP key on reader.

b. Remove cards from hopper.

c. Press NPRO key to clear feed.

d. Press CHECK RESET key.

e. Remove last two cards from stacker and place in hopper.

f. Reload cards.

g. Depress and hold START key on reader to ready the reader.

DATA CHECK -

a. Press STOP key on reader.

b. Remove cards from hopper.

c. Press NPRO key to clear feed.

d. Remove or correct card containing illegal punch.

e. Press CHECK RESET key.

f. Reload cards.

TERM ADDRESS -

The printer or punch was not ready to receive output.

a. Rotate the MODE switch (dial) to OFF LINE and then back to TSM/TRSP.

b. Place cards in hopper, press EOF, press START on printer, and read in the CONTINUE card.

SYNC CHECK - OVERRUN

a. Open front cover of printer. The latch is under the front center of the cover.

b. Set typebar motor switch to TYPEBAR REMOVAL.

c. Rotate typebar thumbwheel to align upper-left edge of typebar guides with red arrow on guide holder. The thumbwheel is just to the front and right of the ribbon spool.

d. Turn typebar motor switch back to ON position.

e. Close cover on printer.

f. Press RESET.

g. Rotate dial to OFF LINE and then to TSM/TRSP.

h. Press START on printer.

i. Read in a CONTINUE card.

NOTE: This condition occurs without any error light indication on some units. The only clue is that the printer READY light goes off and will not come on when START is pushed. Executing steps (a) through (c) above will verify if the typebar has slipped.

END OF FORM - Form check

a. Turn off alarm by hitting STOP on card reader.

b. Open cover on printer by lifting latch on underside of front cover. Hit CARRIAGE RESTORE until all the paper has been fed out.

c. Lift all four paper guides and print position indicator (located directly in front and distinguished by its raised plastic bar with numbers).

d. Set form brake to bottom position. Form brake is located on far-left side of printer, having numbers 0 to 5.

e. Turn clutch (black teardrop shaped button located to left of paper guide) to OUT and push CARRIAGE RESTORE.

f. Slide paper in from left side of printer and place on guides; close guides and print-position indicator.

g. Line paper perforation up with top of ribbon by turning black knob located on right side of carriage.

h. Set form brake to 2.

i. Set clutch to IN and press carriage restore to check height. If incorrect, repeat steps (e) and (g).

j. Follow steps (e) through (i) of the procedure given for SYNC CHECK.

OTHER ERROR CONDITIONS NOT NOTED - When other conditions not noted have occurred, use the RJSTART and LOGON cards. Then see local person in charge of terminals (see 13.1.3). If he cannot help, he will call IBM service for aid.

13.3 PROGRAMMING CONSIDERATIONS

13.3.1 CODE STRUCTURE

The IBM 2780 Transmission Terminal can operate with any of three code structures: Six-Bit Transcode, EBCDIC, and USASCII. All terminals on the IBM 360/95 use EBCDIC, which is the primary code structure for the S/360.

13.3.2 CARD READ/PUNCH

The card read/punch unit of the IBM 2780 Data Transmission Terminal provides the terminal with card input and output capabilities. The card read/punch unit can read up to 400 cards per minute (cpm) and punch up to 355 cpm, except for the terminal in Building 7 which has no punching capabilities. (The actual throughput speed of the card read/punch will vary, depending on the number of card columns that are read or punched and the type of transmission facilities used.)

The card read/punch has a hopper with a capacity of about 1200 cards, a card path with read and punch stations, and a single radial stacker with a capacity of 1300 cards. The stacker can be emptied without stopping the unit. Cards feed parallel from the hopper into the card path, move serially through the read and punch stations to a cornering station, and pass parallel into the stacker transport. Since cards move serially through the read and punch stations, simultaneous reading and punching operations are not possible.

13.3.3 PRINTERS

The IBM 2780 Print Unit, which is similar in appearance and operation to the IBM 1443 Printer, provides printed output for the terminal when operating on-line, and enables a card reader-to-printer listing operation to be performed when operating off-line (see subsection 13.4).

The maximum rated speed of the printers served by the IBM 360/95 RJE facility is 200 lines per minute. All printers use the EBCDIC character set and print up to 144 characters per line. Character density is 10 per inch, thereby providing a printing line of 14.4 inches.

All characters of the character set are mounted on a typebar that travels horizontally on the paper. The typebar ensures that each character of the character set successively passes each print position. To print, a magnet releases a spring-loaded hammer at the proper time, so that the desired character is pressed against the ribbon and paper. Characters to be printed are checked for parity while in the buffer and before printing takes place. A parity-check error at the receiving terminal results in an EOT (end-of-transmission) character being encoded in place of the normal block-checking response. This may result in a partially printed line, depending on when the error was detected.

13.4 OUTPUT

Output at the work stations involves a number of options which are specified in the job entry definition statement (JED) and work station commands:

- The output is directed to the source work station by default as soon as the job is completed and the work station is available to receive it. The output may be deferred or routed to another station (see paragraph 13.1.2 for examples).

- The output may be left at the central system until the user requests it.

- The output may be directed to an alternate user by the originator.

- Output may be requested at any work station, either by the originator or by a user named as the alternate recipient. The recipient who first requests the output receives the only copy of the output.

- The remote user may make multiple copies of his output available to either himself or an alternate by writing his output to a named data set and submitting a job step that executes an OS data set utility program, IEBPTPCH, to copy the output to SYSOUT. The IEBPTPCH program is described in paragraph 9.3.2 of this User's Guide.

- Notification of job completion may be requested. This notification includes indication of normal or abnormal termination.

Details of the Job Entry Control Language specifications for output control are given in the Table of RJE Commands in paragraph 13.1.2 and in the IBM System/360 Operation System Remote Job Entry publication, Form GC30-2006.

13.5 USE OF THE IBM 2780 TERMINAL OFF-LINE

The RJE terminal is capable of listing cards at any time. The IBM 360/95 does not have to be functioning to use this facility. Before a transmit operation, the operator can run an off-line listing of the cards to be transmitted, in order to check the accuracy of the cards. The listing will also indicate any invalid characters, as printing will stop on illegal characters. Any necessary corrections can then be made before the actual transmit operation, thus providing a more efficient and smoother transmit run. When operating off-line, the card read/punch reads cards and prints them on the printer, one line per card.

For the IBM 2780 terminal to be operational, the main-line switch must be turned on and all interlocks satisfied. To operate in the off-line mode, first check:

a. That paper and carriage tape are properly positioned at channel, and press CARRIAGE RESTORE to position paper to top of page.

b. That all interlock conditions are satisfied, i.e., all cabinet doors are closed.

Then:

a. Load the cards into the hopper.

b. Turn dial to OFF LINE.

c. Press START on printer to ready the printer.

d. Press START on the reader until the READY light comes on and the listing starts.

e. When off-line operation is complete, rotate the dial to TSM/TRSP and ready the printer to allow other output to print.

The 2780 terminal will read cards and print until the machine runs out of cards. The printer will stop with a DATA CHECK if an illegal character is encountered. To clear this condition:

a. Remove remaining cards from the hopper.

b. Press NPRO to clear the reader.

c. Press RESET to clear the error condition.

d. Press START on the printer to ready it.

e. Reload the remaining cards in the reader and press START on the reader.

13.5.1 NORMAL STOPS

13.5.2 HOPPER EMPTY

The READY light turns off (card read/punch panel) after the last card has been read and processed. If more cards are to be processed, load them into the hopper and depress the START key.

13.5.3 STACKER FULL

The READY light turns off when the stacker is full. To restart, remove cards from the stacker and press the START key on the card read/punch to resume listing.

When the off-line operation is finished, the dial should be returned to TSM/TRSP, and START pressed on the printer to ready the 2780 terminal in case output is waiting.

SECTION 14

CONVERSATIONAL REMOTE TERMINAL SERVICE (RITS/CRBE)

14.1 GENERAL DISCUSSION

The Remote Input Terminal System (RITS) and the Conversational Remote Batch Entry (CRBE) are two file manipulation and job submission systems supported by the IBM 360 computing systems and the IBM 1050 series communication terminals. The user and the RITS/CRBE system actually enter into a dialogue. The user is in constant communication with the system, which responds to his queries and commands. This system provides complete facilities for the storage and maintenance of source code, JCL, and data, and allows the user to submit jobs through remote 1050 terminals. A FORTRAN prescan is provided to detect FORTRAN syntax errors on a line-by-line basis as the code is entered. RITS is also useful for the running of operational programs requiring rapid turnaround. Any job requiring 700K or less core storage and 11 minutes or less execution time can be submitted through RITS just as through dispatch. Jobs which are larger or run longer may be submitted through RITS in the evenings. Simple utilities are available for placing card files on disk for access by RITS and CRBE or for obtaining an actual deck from the disk files, if it is required. The RITS and CRBE user has the capability of creating his own files directly at the terminal keyboard, thus eliminating the need for cards. A new user already familiar with OS/360 should be able to obtain reasonable proficiency at RITS or CRBE after only a few days of use because the required commands are English-like in their structure and easy to use. This section is primarily concerned with RITS on the 360/95, and supplements the information in the RITS User's Guide. The SESD CRBE system is currently being refined and expanded. Information on CRBE may be found in the CRBE User's Guide, the SESD User's Guide, and the GSFC Computer Newsletters.

14.1.1 LOCATION OF TERMINALS

There are numerous 1052 communication terminals located throughout GSFC. Each of these terminals is open for "public" use. Terminals located in individual offices may be used by anyone requiring access to the RITS or CRBE facilities. Problems of terminal accessibility and location should be referred to Mr. Harry Bitting, Building 3, Room 128, extension 6886. There is a 1050 data communication system located in Building 3, Room 137. This terminal is capable of reading small amounts of cards and paper tape while on-line to the 360 system.

Refer to the IBM 1050, Operator's Guide (Form GA24-3125) for detailed instructions and background information on the use of the 1050 for reading cards.

14.1.2 COMPUTERS SUPPORTING RITS AND CRBE

Supporting Computer	Service	Telephone Number	Console Number	System Administrator
360/95	RITS	982-3116 (Dial 36 at GSFC)	5395	Government Monitors, Ext. 6781, Room 171, Building 3
SESD 360/75	CRBE	982-3113 (Dial 33 at GSFC)	6539	Mr. Paul Heaps, Ext. 6596
SESD 360/91	CRBE	982-3112 (Dial 32 at GSFC)	6015	Mr. Paul Heaps, Ext. 6596
Manned Flight 360/75	RITS	982-3115 (Dial 35 at GSFC)	5130	Mr. Fred King, Ext. 4736

Note: The RITS System on Manned Flight 360/75 is available on a very limited basis to selected MFLT personnel only.

Requests for user IDs and notification of changes in sponsor numbers should be forwarded to the appropriate system administrator. New users are required to complete a user's course (see paragraph 14.1.9 for course information) or show proficiency in RITS, CRBE, or some similar system.

14.1.3 HOURS OF SERVICES

The normal operational hours are from 8:00 a.m. to 8:00 p.m., Monday through Friday. RITS on the IBM 360/95 may not be available when the computer is being used to support a launch. If the user needs RITS at other than normal working hours, the 360/95 computer operator (extension 5395) should be contacted for assistance.

14.1.4 TAPE MOUNTS

Paragraph 2.3.4 of this User's Guide contains a discussion of the policies and procedures pertaining to the use of tapes in RITS-submitted jobs. The SYSTM.TAPES file, available on the IBM 360/95, contains the latest information about tapes for the model 95. To list this file, the following command must be entered:

```
/F   SYSTM.TAPES
%LIST
```

14.1.5 NOMINAL SPACE ALLOCATIONS

Each authorized user is assigned one cylinder on a 2316 disk pack to hold his permanent files. Additional space may be obtained on a temporary basis by contacting Mr. Harry Crispell, extension 6797, for 360/95 RITS, or Mr. Paul Heaps, extension 6596, for CRBE or 360/91 CRBE. Indications of space becoming filled are given by frequent user library condensations, resulting in the compression of the user's space (i.e., by removing those files which have been purged or replaced and are no longer accessible). The user can limit his library condensations by using smaller files when frequent modifications become necessary. For example, FORTRAN programs can be divided into sections during debugging when frequent program changes are necessary. The use of an FLIST (see paragraph 14.2.6) to join the sections for submission to the FORTRAN compiler will eliminate any possibility of confusion when running the jobs. Each cylinder holds about 1200 cards.

14.1.6 NEWS FILES

The RITS System, on the IBM 360/95, maintains an active news file (SYSTM.NEWS) with up-to-date information of interest to users. This file can be fetched (F SYSTM.NEWS) and listed (%LIST) periodically by interested users, to learn the latest RITS information.

14.1.7 ASSISTANCE

Programmer and RITS assistance for the IBM 360/95 is available during normal working hours in Building 3, Room 133, extension 6768. CRBE assistance is available from the system administrator, Mr. Paul Heaps, in Building 21, Room 230, extension 6596. All I/O errors while reading or writing into RITS files should be reported immediately to Mr. Harry Crispell (extension 6797) so that an attempt may be made to regenerate the files (360/95 only) to insure against hardware malfunctions causing lost data. Suspected 1050 hardware problems should be directed to the system administrator.

14.1.8 REFERENCES

The information related to RITS and CRBE can be found in the following manuals: (1) CRBE Conversational Remote Batch Entry - User's Guide and (2) RITS Remote Input Terminal System - User's Guide. The CRBE manual is available from Mr. Paul Heaps, Room 230, Building 21. The RITS manual is available from Mrs. Pat Barnes, Room 133, Building 3.

Subsection 14.4 of this guide contains examples of utilities useful for printing, punching, or building files used by RITS. Section 9 of this guide contains examples of utilities of general interest which may be used through RITS-submitted jobs.

14.1.9 RITS AND CRBE CLASSES

CRBE classes are arranged through Mr. Paul Heaps, Building 21, Room 230, extension 6596. RITS 95 classes are arranged through Mr. Harry Crispell, Building 3, Room 129, extension 6797. An ID will not be issued to anyone until he completes a class or can demonstrate proficiency in RITS or CRBE.

14.2 PROGRAMMING CONSIDERATIONS

14.2.1 GENERAL DISCUSSION

The 1050 series terminals offer a slow printer intended for data sampling and not for long output. RITS and CRBE support facilities which permit the user to choose those data sets he wishes through the terminal. Data sets to be printed fully can be sent to the system printers available at the 360 computers.

14.2.2 LINE LENGTH

The 1050 terminals support a line length of 120 characters. Override cards are always required for changing the record length and blocksize of data sets to be displayed on the 1050s.

These override cards should also reflect the output class (SYSOUT=R) for output to be displayed through RITS. For example, the output data set containing the FORTRAN listing from the compiler may be retrieved through RITS if the following card is included:

```
//COMPILE EXEC FORTRANH,PARM='OPT=2,NOMAP,NOXREF,NOLIST'
//SYSPRINT DD SYSOUT=R,DCB=(RECFM=FB,LRECL=120,BLKSIZE=3600)
```

The output from the user's program can be retrieved through RITS by adding the required override card in the LINKGO or LOADER step. This card should reflect the programmed line length as closely as possible. The programmer should attempt to confine his lines to 80 characters, remembering that RITS "loses" the first character in a manner similar to the carriage control characters. For example:

```
//LOAD EXEC LOADER (OR LINKGO)
//FT06F001 DD SYSOUT=R,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3600)
```

Other files to be displayed through RITS may be specified in a similar manner. The message "BLKSIZE INVALID" in response to a BRING indicates the need for an override card to change the DCB of the data set being displayed.

14.2.3 SELECTION OF DISPLAYED FILES

The user who submits jobs through RITS must decide which printed data sets he wants to display through RITS (SYSOUT=R) and which data sets he wishes to assign to the system printer (SYSOUT=A). This decision must take into account the relative value of the printed data sets and the speed of the 1050 series printers. It is important for the user to remember that only one copy of each SYSOUT data set exists.

The user may, after displaying part of a data set, decide to print the entire data set on the system printer. This may be done using simple utilities provided for this purpose. To do this, he should save the desired printed output in his RITS file (SAVE TEMP,PURG). This file may then be printed, using PRINT. (See paragraph 14.4.4 of this guide.)

14.2.4 RETRIEVING OUTPUT THROUGH RITS

Each data set created with an output class of SYSOUT=R will have a pointer to it on a file called SYSMSG.jobname. This file will also contain all the JCL if MSGCLASS=R is specified on the job card. The user should carefully consider the message level when submitting jobs with MSGCLASS=R. The use of MSGLEVEL=(2,0) on the job card will provide a listing of all input JCL and error messages and SYSOUT=R data sets, while eliminating the procedure expansion listings. To BRING this data set, the user may:

```
/BRING D=SYSMSG.jobname,V=G1SCR5
%LIST
```

If the message level is MSGLEVEL=(0,0) or MSGCLASS=A, the message data set will contain the name and location of all SYSOUT=R data sets created in that job. If a higher message level is used and an MSGCLASS=R is used, the name and location of SYSOUT=R data sets will be scattered throughout the JCL located on SYSMSG.jobname. The MSGLEVEL and MSGCLASS are coded on the job card after the accounting information. The BRING is identical on CRBE, but the SYSMSG data sets are cataloged and there is no need to provide a volume. RITS 95 always places the SYSMSG data set on G1SCR5.

14.2.5 SPECIAL NOTES ON BRING

A RITS user may BRING, LIST, and, if he so desires, SAVE any sequential or partitioned data set member stored on a volume recognized by RITS. These data sets need not be cataloged, but there must be a DD card in the RITS startup deck for the direct-access volume upon which the data sets reside. Currently, a user may BRING from the following 2316 disk packs: G1SYS1, G1USR1, G1USR2, RITS04, RITS08, G1SCR1 through G1SCRA, and from the G1DRM1 and G1DRM2 2301 drums. For example, when modifying a procedure by inserting override cards, the user must place these cards in the same relative position as they appear in the procedure. A copy of the procedure may be brought and displayed by:

```
/BRING D=SYS1.PROCLIB(FORTRANH)
%LIST
```

Note: SYS1.PROCLIB is normally a cataloged data set. If it is not, the volume identification must be specified:

```
/BRING D=SYS1.PROCLIB(FORTRANH),V=volume id
%LIST
```

This lists the FORTRANH procedure to permit the user to override SYS1.PROCLIB correctly. The library, SYS2.USERPROC, is also available in the same manner. Request for private disk packs cannot be honored since the pack may not be on-line at all times when RITS is up. A user cannot BRING data sets from the data cell. However, jobs that make use of data sets on the data call may be submitted through RITS. (See paragraph 14.3.2 of this guide.)

14.2.6 USING AN FLIST

An FLIST can be created with the command:

```
/CREATE name, FLIST
```

The resultant file may contain any valid card images, i.e., FORTRAN[1] source statements, JCL, or data. In addition, it may contain RITS files names, preceded by a "=" symbol. For example, the FLIST:TESTRUN could be built to compile, load, and test a program against simulated data by entering:

```
            /CREATE TESTRUN, FLIST
100         =jobcard
200         //COMPILE EXEC FORTRANH,PARM='OPT=2,NOMAP,NOXREF,NOLIST'
300         =program
400         =SUB1
500         //LOAD EXEC LOADER
600         //SYSIN DD *
700         =data
800         %SAVE,PURG
```

To submit the entire job, the user need only enter:

```
/SUBMIT GAABCRIT
=TESTRUN
endinput
```

and RITS will fill in the files listed.

Do not submit an FLIST from the active file (=*) as this will cause the abnormal termination of the job you are submitting.

[1] Note: The use of FORTRAN source statements within the FLIST file itself should be held to a minimum as an FLIST file is limited to approximately 99 card images.

14.3 INTERACTION WITH OS FROM RITS

14.3.1 SUBMISSION OF JOBS

Jobs may be submitted through RITS or CRBE to run under OS as "normal" batch jobs. Except for those files that the user wishes to display through the terminal (SYSOUT=R), no modification of the JCL is needed. Jobs requiring more than 700K core storage or 11 execution time minutes are not permitted during the first-shift time (8:00 a.m. to 4:30 p.m.).

14.3.2 CATALOGED DATA SETS

A user may display members of selected cataloged data sets directly from the terminal. These data sets must reside on permanently mounted volumes. For example, to display the current LOADER procedure, a user may:

```
/BRING D=SYS1.PROCLIB(LOADER)
%LIST
```

Other data sets which do not reside on permanently mounted volumes may still be referenced through RITS by using the IEBGENER utility. For example, to BRING and modify members of DODS.SRCLIB, the user may move a copy to his RITS file by submitting this RITS job as follows:

```
/SUBMIT GAABCRIT
=jobcard
//NEED    EXEC      PGM=IEBGENER
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   DSN=DODS.SRCLIB(membername),DISP=SHR
//*DODS.SRCLIB IS CATALOGED, NO VOLUME AND UNIT NEED BE SPECIFIED
//SYSUT2       DD   SYSOUT=R,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3600)
//SYSIN        DD   DUMMY
endinput
```

IEBGENER will produce on SYSUT2 a listing of the specified member. RITS will produce, on the SYSMSG.GAABCRIT file a line showing where this (SYSUT2) listing was placed. To retrieve a copy of the selected member, use:

```
/BRING D=SYSMSG.GAABCRIT,V=G1SCR5
%LIST
```

This tells the user where his copy has been stored. He may then BRING and SAVE this in the normal manner:

```
/BRING D=(name assigned),V=(volume assigned)
```

The file may be resequenced by saving and refetching it, as required:

```
%SAVE PROG,PURG
/FETCH PROG,(100,10)
%
```

This will resequence the file PROG, starting at 100 and counting by 10. PROG should then be SAVE'd to reflect the resequencing.

14.4 UTILITIES IN RITS

RITS and CRBE support, without exception, all standard OS utilities. However, there are a number of special uses of utilities through RITS 95 to aid in running jobs. These may be exercised, with small modifications, under CRBE.

14.4.1 BUILDING RITS FILES FROM DECKS

The procedure RITSDECK may be used to store decks for later processing by RITS, i.e., placing FORTRAN routines in separate files from JCL and using one subroutine per file to simplify editing. To place a file on disk, use:

```
//RITS EXEC RITSDECK,ID=GAAZZ001
//SYSIN DD DATA
./ ADD NAME=JCL
(First JCL cards go here)
./ ADD NAME=PROGRAM
(Program goes here)
./ ADD NAME=SUB1
(Subroutine goes here)
./ ADD NAME=DATA
(Final JCL and DATA go here)
./ ENDUP
/*
```

The result is a partitioned data set on RITS08 with members JCL, PROGRAM, SUB1, and DATA.

Notes:

1. For "ID," use RITS ID plus three other alphameric characters to insure uniqueness for each execution of RITSDECK.

2. /* cards must be removed from the deck being placed on RITS, since the program terminates when it reads a /* card. The final /* is required to terminate the "//SYSIN DD DATA" field.

3. To bring the files to RITS, use:

```
/BRING D=GAAZZ001(JCL),V=RITS08
%SAVE JCL,PURG
```

   The member of the PDS to be brought is specified in parenthesis. The JCL file may be resequenced by:

```
/F JCL,(100,100),SEQ,OTHER
```

JCL should always be sequenced; the FORTRAN program is initially brought and saved in the same way. However, the first resequencing requires the use of:

```
/F PROGRAM,(100,100),SEQ,SCAN,FORTRAN
```

14.4.2 LISTING RITS FILES

The procedure LISTRITS will print an entire user's library with one control card, as follows:

```
//LIST EXEC LISTRITS,USRID=GAAZZ
```

Notes:

1. If "USRID=" is omitted, the system news files are listed.

2. A few members can be listed by using the punch utility (see paragraph 14.4.3).

14.4.3 PUNCHING SELECTED RITS FILES

To punch a deck from a RITS file, the following RITS job must be executed:

```
/SUBMIT GAABCRIT
=jobcard
=SYSTM.PUNCH
=first file to be punched
=next file to be punched
endinput
```

Notes:

1. Any number of RITS files may be punched at one time.

2. If only a small amount of punch output is expected, the operator should be informed by:

```
/SEND OP,'RITS JOB GAAZZ001 EXPECTS PUNCH OUTPUT.
```

3. The file SYSTM.PUNCH uses the utility IEBGENER which punches copies of anything it sees as data.

14.4.4 PRINTING SELECTED RITS FILES

The file SYSTM.PRINT may be used to list selected members of the RITS library if a total library listing is not desired. To obtain a list, the following RITS job should be submitted:

```
/SUBMIT GAABCRIT
=jobcard
=SYSTM.PRINT
=the first RITS file to be printed
=the next file to be printed
endinput
```

Notes (1) and (3) above apply here for the printing of files.

14.4.5 TAB

The file SYSTM.TAB contains a column guide which assists in using the tab function or in entering FORTRAN statements. To use this column guide, enter:

```
/FETCH SYSTM.TAB
%LIST
```

14.4.6 CARD SIMULATOR FOR RITS/CRBE JOBS

A "RITS/CRBE simulator" is available for the SESD and M&DO 360 systems. With this facility, a programmer can submit a job through the dispatcher and can access files and FLISTS from RITS/CRBE libraries.

The simulator has the following features:

1. CRBE or RITS jobs that cannot be submitted from remote terminals during the day because of time or memory restrictions may be submitted for loading by the operator, in the evening, through use of a small card deck.

2. A programmer may submit jobs that consist of RITS/CRBE library members combined with card decks in the input stream. This avoids the problem, formerly encountered by the user, of having half his program in card form and half in his CRBE or RITS library. He may now enter the entire program using a combination of this simulator and card decks.

To use the RITS simulator on the 360/95, the following control cards are used:

```
//jobname       JOB ...
//stepname      EXEC       RITSCARD,ID=usrid
//SYSIN         DD         DATA
SUBMIT jobname
     .
     .jobstream just as entered from the keyboard
     .
endinput
/*
```

Any valid identification (five characters) for a user having a RITS/CRBE library on disk may be used for "userid." The pack must be specified if the user is not on RITS04. The data set RITS.LIB.userid will be opened throughout the generation of the job to be submitted. To user should request that his RITS pack be mounted on the job submission slip.

Note: Submit only one job slip per deck. If the operating system goes down while a job accessing RITS libraries is running, that job will not be reloaded.

SECTION 15

APL - A PROGRAMMING LANGUAGE

15.1 GENERAL

The APL language was developed by K. E. Iverson and first outlined in the book, A Programming Language. This work has become a classic in programming literature. The language supplies the conciseness necessary to maintain a clear picture of a problem, and includes the precision required for essential detail. Most of the notation used in the language is common to ordinary mathematics.

APL is a Program Product supplied by IBM. It is a conversational time-sharing system having all of its input and output through remote terminals. Currently, no facilities exist in the Program Product for using the central readers, printers, or tapes. All APL programs execute line by line as they are entered from a terminal, rather than executing in the usual Compile, Link, Go sequence. This results in two distinct differences between APL and other languages:

1. JCL is never required to use APL; thus, a high degree of computer independence is maintained.

2. If the user sees an error developing as his program is running, he can stop the program, correct program or data errors, and resume processing.

The total range of uses for APL appears limitless; however, there are some areas in which it is a particularly convenient language. Vector and matrix functions are fully developed and easy to use. All conceivable mathematical operations, including a number of integration schemes, many as elementary functions, are available. Literal data handling is straightforward. Large data bases are easily handled. The example below demonstrates the conciseness of APL. It finds the average of a vector A, of size N, in FORTRAN and APL.

FORTRAN	APL
DO 10 I = 1, N	TOTAL ← +/A ÷ ρA
10 TOTAL = TOTAL + A(I)	
TOTAL = TOTAL/N	

15.2 LIBRARIES

Mrs. G. M. Wilson, GSFC extension 6797, has compiled a document (Guide For APL Libraries, X-543-71-69) which contains all of the library functions currently available at the GSFC for the APL user, their uses, and ways to use them. The document has been sent to all registered APL users.

A list of holdings in each library can be obtained by typing:

```
)LIB      x
```

where x is 1, 3, 4, 5, 6, 7, 9, 12, 100, or 999. A description of a holding can be obtained by typing:

```
)LOAD     x    INDEX
DESCRIBE
```

with the exceptions of libraries 999 and 1. Library 999 uses INDEX2 rather than INDEX. Library 1 has no index.

Library 1 contains a news file updated to reflect the latest APL system status. To receive a copy, type:

```
)LOAD     1    NEWS
APLNOW         MM DD YY
```

This will print all news items since the MM (month), DD (day), and YY (year) specified.

15.3 USING APL

An APL/360 system comprises a control computer, the M&DO 360/95, an APL operator's terminal (currently a 2741), and a number of remote terminals.

Virtually any remote terminal will work with APL; however, only IBM 1050s (RITS Terminals) are currently available at GSFC. A special APL character set is utilized by APL to provide unambiguous communication between the computer and the user. This character set is available on a selectric printing element (golfball) number 1167988. In conjunction with the typing element, a special set of APL character overlays (IBM form X20-1783) is available for attachment to the terminal keyboard. The application form for obtaining an APL id is provided at the end of this section. If the user is interested, he should complete the form and return it to Mrs. May Wilson, Code 543.1.

15.4 SIGN-ON PROCEDURE - 1050 TYPE TERMINALS

The user must mount the APL golfball. He must turn on the terminal power switch, and set the following switches on the control panel to the indicated settings:

SWITCH	SETTING
ATTEND/UNATTEND	ATTEND
PRINTER	SEND-RECEIVE
KEYBOARD	ON
EOB	MANUAL

The TEST switch should be OFF. The RESET LINE key should be pressed if the RECEIVE ALARM light is on. If the receive alarm light does not shut off, the user may be out of paper or have a terminal hardware problem. Hardware problems should be reported to the Government Monitor on extension 6781. The DATA CHECK button should be depressed if the adjacent white light is on.

The user may dial the computer by pressing the data phone TALK button, lifting the receiver, and dialing 982-3114 (34 at GSFC). When the computer "answers" with a high-pitched tone, he must press the "DATA" button on the data phone and replace the receiver. (If the computer does not "answer" by the second ring, the user should hang up and try again later.)

Then the user presses the REQUEST key. When the PROCEED light goes on and the keyboard unlocks, he enters:

```
)xxxx
```

where xxxx is the user's APL number. (See subsection 15.6 for the procedure to get an APL id.) To terminate the line, the user must press RETURN, and then simultaneously press the ALTN CODING and EOB buttons.

At the completion of the user's first session, he must assign himself a lock (password) for his APL id. He does this by typing:

```
)OFF:yyyy
```

where yyyy is the password. The user must remember it, as there is no other record of this password. The password may be any length; however, only the first eight characters are checked by the system.

The next time the user signs on, he enters:

```
)xxxx:yyyy
```

where x and y are as defined above.

The user may change his password, using the sign-off step with the new lock.

The user need not re-enter his password each time he signs off; he merely enters:

```
)OFF
```

15.5 OPERATOR COMMUNICATION

The 360/95 operator is not the APL system operator. Messages routed to the APL operator go to the APL operator's terminal in Room 129, Building 3, extension 6797. Questions directed to the APL operator may go unanswered if there is no one present when the message is sent.

15.6 APL COURSE

There are currently two different APL courses available in the GSFC APL library. They are referenced in Mrs. May Wilson's document, Guide For APL Libraries, X-543-71-69. In the future, an APL course consisting of 16 one-hour video-tapes may be offered. A general announcement will be made when the course is available.

Library 1 contains a self-teaching APL course called APLCOURSE. One may access it by:

```
)LOAD     1    APLCOURSE
DESCRIBE
```

The resultant description is clear.

Library 999 also contains a self-teaching APL course. It may be accessed by typing:

```
)LOAD     999  A
START
```

For an introduction to the course:

```
)LOAD     999  CATALOG
DESCRIBE
```

Note that all references to Library 9 should read Library 999.

15.7 REFERENCES

1. A Programming Language, by K. E. Iverson (Wiley, 1962) is a basic reference and one every serious student will want to read. It completely outlines the language and illustrates hundreds of programs and applications for which APL is suited.

2. The APL 360 Primer (IBM Form GH20-0689), provides an introduction to APL 360 for a programmer with no APL background. At least the first three chapters of the Primer should be read before using the APL terminal.

3. The APL 360 User's Manual (IBM Form GH20-0906), provides some elementary and many advanced examples. It describes sample work sessions and covers advanced APL topics.

4. The APL 360 Reference Manual, by Sandra Pakin (Science Research Associates, 1968), is an adequate reference for experienced APL users. Miss Pakin uses the APL notation to describe each available APL function.

5. Time Sharing Languages: APL (Auerbach Info., Inc., 1970), is an excellent history and language summary. This report is useful in the beginning phases of learning the language, since it provides a clear explanation of the primitive APL operations most used by the APL student.

6. APL/360 An Interactive Approach, by Leonard Gilman and Allen J. Rose (John Wiley & Sons, Inc., 1970), is almost a complete course in itself. It is highly recommended.

COMPUTER SYSTEMS BRANCH

APL APPLICATION APL - A PROGRAMMING LANGUAGE

Date _____

INSTRUCTIONS: Complete this Form and send to Code 543, Attention: Mrs. May Wilson

Name _____

GSFC Programmer ID _____ Sponsor number(s) _____

☐ Goddard Employee ☐ Contract Employee

Code _____ Company: _____

GSFC contact _____ Code _____

Telephone _____ Bldg. _____ Room _____

List any previous APL experience

I am experienced in the use of: ☐ RITS/CRBE ☐ FORTRAN

☐ 360 Assembler Language ☐ PL1

☐ Other _____

Are you currently an APL user? ☐ No

☐ Scientific Time Sharing Corp.
user id. _____

☐ Other _____

List suggestions or comments?

(Signature) _____

NOTE: You will be required to maintain a LOCK on your ID. Users failing to do so will be deleted from the system.

540-23(7/70)

SECTION 16

MEMORY USAGE

This section describes: (1) ways of specifying the core required by a job step, (2) means of reducing the storage required by a program via the use of program overlays, (3) the technique of breaking a program into a series of job steps, and (4) the use of the ATTACH, LINK, and XCTL macros to dynamically load and transfer control to load modules.

16.1 GENERAL MVT CONSIDERATIONS

One significant feature of MVT, with respect to memory, is that the programmer does not know where his program will be loaded; the load origin can change from run to run. Since the M&DO computers have high-speed and low-speed core, the load origin can materially affect run timing. The higher speed core is installed in the lower hardware addresses (starting at 0); however, the normal loader allocates core from the top of core down. Models 95 and 75 use a special version of the Loader which allocates from the bottom of core up. Without memory hierarchy support, there is no way a programmer can specify which type of core he wishes to allocate to his program. Memory hierarchy support is a SYSGEN option currently not taken on the M&DO computers. Programs requiring more than 700K of core should be overlaid; otherwise, they can only be scheduled for overnight runs of the model 95 (see subsection 18.3).

16.2 REGION PARAMETER

The REGION parameter is the means by which a programmer tells the system the amount of main storage required by the job or job step. The REGION parameter may be used on the JOB or EXEC statements. When it is used on the JOB statement, that amount of space is allocated for each job step, and any REGION parameters specified on EXEC statements are ignored.[1] If the REGION parameter is omitted from the JOB and EXEC cards, a default value (as established by the input reader procedure) is assigned. The form of the REGION parameter is:

[1] Previously, by specifying a REGION parameter on the JOB card, the same core allocation was kept and used for each job step within a job. This was a good way to insure having core available for a larger step (or steps) within a job. This is no longer true; the system will reallocate core for each job step within the job. Since the REGION parameter on the JOB card no longer serves the purpose it was originally designed for and may needlessly tie up valuable core, its use is discouraged. On all M&DO 360 computers, if the REGION specification appears on the JOB card, a JCL error will result, and the following message will be issued.

IEF632I FORMAT ERROR IN THE REGION FIELD

a. REGION=nnnnnK
   or
b. REGION.procstepname=nnnnnK

where nnnnn is the number of contiguous 1024-byte blocks required.

Only (a) may be used on the JOB card. When (a) is used on an EXEC card, all REGION parameters in a cataloged procedure will be overridden. The use of (b) overrides the REGION parameter in the named step only.

The amount of core requested on the REGION parameter affects the class of the JOB in scheduling, with the size cutoffs at 400K and 700K and over. A job requesting more than 700K will be run overnight. Jobs requesting 2000K or more will be held until the programmer confirms by phone that there was no error in the REGION parameter.

See subsection 18.3 for a discussion of job class and priority on the M&DO computers.

The form of the REGION parameter that specifies storage hierarchy requirements should not be used, as storage hierarchy support is not available on any of the M&DO computers.

16.3 MULTI-STEPPING

Very large and long-running programs are better handled by dividing them into separate job steps. This allows the programmer to generate a restart capability in case a portion of the program terminates abnormally (see subsection 11.5). Further, the subdivision into job steps can result in better use of the system; only that amount of space required for each step is used, instead of the maximum amount required for the job.

The most convenient method of calling one program (such as the Sort package) from another program is by means of a separate job step. This method eliminates recoding or recompiling the program when a change in the Sort parameters is required; only the JCL need be changed.

16.4 ATTACH, LINK AND XCTL MACRO INSTRUCTIONS

In a very large and complex program, sometimes the required overlay structure cannot be statically determined. For these dynamic structures, the programmer can maintain control over which modules are to be loaded, and when this loading is to occur.

The ATTACH macro instruction causes a specified load module to be loaded (if necessary), creates a new task (with the specified priority), and transfers control to the new task while the old task continues execution.

The LINK macro instruction is used to load a module (if necessary) and to transfer control to the new load module at the specified entry point. Return is made via the RETURN macro instruction.

The XCTL macro instruction is like the LINK macro, except that the calling module is deleted after the call; thus, no return is possible. Care should be taken not to mix XCTL macros with direct branch instructions between modules; instead, the module that issues the XCTL should have control passed to it via a LINK. Refer to IBM Supervisor and Data Management Services (GC28-6646), for information, and IBM Supervisor and Data Management Macros (GC28-6647), for coding examples.

16.5 OVERLAYS

When a program becomes large (especially if is is over 700K), the overlaying technique should be used. Overlaying consists of analyzing the program to find independent areas of the program. These mutually exclusive areas can be loaded into the same region of core storage, with one part "overlaying" the other as necessary. Data are usually passed through common areas in segments that are in the common path of the exclusive segments.

Each time a reference is made to a routine in one of the overlay segments, that segment is brought into core. Care should be taken not to cause very frequent calls between segments in the same region, as this will increase the program's execution time.

Reference may be made to an overlay segment by a CALL statement (or macro instruction), by a branch instruction, or by a SEGLD or SEGWT macro instruction. Refer to IBM Supervisor and Data Management Macros (GC28-6647) for more information on these macro instructions.

A program's overlay structure is declared via the Linkage Editor statements INSERT and OVERLAY. Refer to Section 22 and the manual, Linkage Editor and Loader, IBM (GC28-6535), for examples of overlay structures and coding.

16.6 MEMORY HIERARCHY SUPPORT

Hierarchy support is explained in the IBM manual, Planning for Hierarchy Support (GC27-6942). It is a SYSGEN option and is not supported at GSFC. Hierarchy support is normally used on machines with two types of storage, one faster than the other. It allows the programmer to assign an executable code to the faster storage and data to the slower storage.

If storage hierarchy support is generated into an operating system, it affects all jobs run by that system. In order to use hierarchies efficiently, a job must be compiled, linked (or loaded), and executed with hierarchy support in mind. If not (and the overwhelming majority of software is not), system performance may be seriously degraded. The Assemble and Linkage Editor are presently the only processors which support the hierarchy feature.

16.7 MEMORY TRADE-OFFS

One of the decisions that a programmer commonly makes is whether to save execution time at the expense of using additional memory. These considerations affect many areas of a program, such as whether to use double buffering on I/O at the expense of the extra buffer, whether to use a smaller blocksize which degrades I/O time and space but requires less core, or whether to use subroutines instead of inline code, i.e., trading off a slight decrease in execution time in order to increase the program area caused by repetition of the inline code. A trade off that does not deal directly with memory is the use of the FORTRANH AND PL/I (F) option 2, which, while taking longer to compile, provides gains in speed of execution and better core usage.

These considerations are primarily the concern of programmers developing programs for a production environment. A program that will only run once does not particularly benefit from the extra time a programmer spends in fine tuning.

SECTION 17

DATA MANAGEMENT TECHNIQUES

17.1 GENERAL ASPECTS OF DATA MANAGEMENT

The IBM manual, Supervisor and Data Management Services, GC28-6646, describes the access methods provided by OS. Subsection 5.5 and 5.6 discuss the format and coding of the basic DD parameters. This section presents some techniques and pitfalls of significance on M&DO computer systems.

17.1.1 USE OF NAMES

The following names are of importance to data management: DSNAMEs, DDNAMEs, UNIT names, and VOLUME names. When writing a program, the programmer codes instructions which act on files. The programmer codes a data control block (DCB) explicitly in assembly language and implicitly in FORTRAN and PL/I. Within the DCB is a reference to a data definition statement or DDNAME. The DD card refers to a data set or sets by DSNAME. These data sets reside on VOLUMES which are mounted on devices (UNIT name on DD card). Volumes are referred to by serial numbers and devices are referred to by unitnames; unitnames are discussed in Section 19.

17.1.1.1 DDNAMES

The use of DDNAME as an operand is described under "Postponing Data Set Definition" in paragraph 5.6.6.4. In this section, the term ddname will refer to its use in the name field of the DD card. The ddname coded in the DD card corresponds to the filename defined in the program. This filename is defined by the name field of the DCB macro in ALC, and by the filename in PL/I. In PL/I, input files default to filename SYSIN, and output files default to SYSPRINT. In FORTRAN, ddnames are formed from the FORTRAN Reference Number (FRN) and the FORTRAN sequence number. FRN nn defines a ddname FTnnFyyy, where yyy is the FORTRAN sequence number. The FORTRAN sequence number is somewhat of an anachronism in OS 360, as direct-access devices do not have files and the LABEL parameter provides the file sequence for tape. Historically, FORTRAN has used certain FRNs for specific purposes, e.g., five for card-to-tape input, six for printer output, and most FORTRAN procedures have DD statements to support this usage. In fact, the FORTRAN coder can use any available FRN for any purpose by appropriately defining his DD card. The FORTRAN programmer

can bypass the limit on FRNs by indicating that he is using a new FORTRAN sequence number. The coding to accomplish this is described in the FORTRAN G&H Programmer's Guide, GC28-6817, under FORTRAN sequence number. It is not recommended unless necessary, and comments should be inserted to indicate what is occurring.

Each job step may contain up to 254 ddnames. By concatenation, these DDs may reference many more data sets. If the ddname is not referenced by the program, data management allocates and de-allocates these data sets.

Five ddnames are listed by the system for special purposes. Therefore, they should not be used as ddnames by problem programs. These names and their use are:

ddname	DESCRIPTION	REFERENCE
JOBLIB	Defines a private program library for the duration of the job.	8.10
STEPLIB	Defines a private program library for the duration of the step.	8.1.1
SYSABEND	Defines a dump data set.	21.4
SYSUDUMP	Defines a dump data set.	21.4
SYSCHK	Defines the checkpoint data set.	GC28-6538

17.1.1.2 DSNAMES

The DD card associates a file defined (in the DCB) in the program with data for a particular run. Since the DD card can be changed from run to run, programs can process many distinct data sets as long as the attributes are within the scope of the program.

The DSNAME field can be up to 44 characters long. A maximum of 8 characters can be used at each level of indexing with each level separated by a period (.). Although 22 levels are possible, since the fully qualified name, i.e., including all levels, must be used in the DSNAME parameter; this can prove tedious if the user tends to use long names.

Higher levels of indices must be built (by IEHPROGM - Section 9) before indexed data set names may be cataloged by data management. IEHPROGM will generate index levels, as necessary, if it is used to catalog a data set. If the programmer does not use the catalog, the entire data set name is treated as a single field by data management. In searching the catalog, each index level is matched until the lowest level is found. This level contains user-supplied unit and volume information for the data set. When referring to the catalog, the data set is defined as existing (DISP=OLD or SHR) while the UNIT= and VOL=SER= parameters are not coded on the DD card. Of course, if the data set is not where the catalog says it is, a JCL error will occur when the VTOC or label is searched. Scratching a data set does not uncatalog it unless it was deleted by the job that referenced it via the catalog. Every entry in the catalog must be unique. On the other hand, the same DSNAME may appear in every VTOC and every tape file. This is not recommended, however. Backup copies are sometimes maintained by keeping data sets on two volumes and switching the catalog pointer from one to the other.

17.1.2 VOLUME STATES AND ATTRIBUTES

Volumes can be assigned different states, depending on the type of data set being defined and the manner in which a volume is requested. The volume state controls volume sharing and volume demounting.

A request for a volume can be specific or nonspecific. A specific volume request includes the volume's serial number(s), e.g.:

```
VOLUME=SER=nnnnnn
```

which states the serial number explicitly.

Reference can be made to a previously defined data set, e.g.:

```
VOL=REF=*.stepname.ddname

VOL=REF=*.stepname.procstep.ddname

     or

VOLUME=REF=dsname
```

which states that the serial number is the same as that defined in the reference.

A nonspecific volume request has neither the SER parameter nor the REF. For a complete discussion on the VOLUME parameter and the subparameters defined above refer to the Job Control Language User's Guide (GC28-6703) and the Job Control Language Reference (GC28-6704) manuals, and to PRESRES in the System Programmer's Guide (GC28-6550) for an indepth explanation of volume states. Refer to the OS Operator's Guide (GC28-6540) for a discussion of the MOUNT, UNLOAD, and vary commands.

17.1.2.1 Physical Volume Attributes

Because direct-access volumes can be used concurrently by more than one job, they can assume different volume states than tape volumes, as shown in Table 17.1.2-1. The following discussion applies to direct-access volumes, except where noted.

A mounted volume may have the physical attribute of being permanently resident, reserved, or removable. This characteristic of a mounted volume determines if and when a volume is demounted.

Table 17.1.2-1. Volume States and Their Characteristics

Volume State	Type of Volume Request: Temporary Data Set	Type of Volume Request: Nontemporary Data Set	How Assigned	How Demounted
Public/Permanently Resident	Nonspecific or Specific	Specific	PRESRES Entry or by default	Always mounted
Private/Permanently Resident	Specific	Specific	PRESRES Entry	Always mounted
Storage/Permanently Resident	Nonspecific or Specific	Nonspecific or Specific	PRESRES Entry	Always mounted
Public/Reserved	Nonspecific or Specific	Specific	PRESRES Entry or MOUNT command	UNLOAD command
Private/Reserved (Tape and direct access)	Specific	Specific	PRESRES Entry or MOUNT command (Only MOUNT command for tape.)	UNLOAD command
Storage/Reserved	Nonspecific or Specific	Nonspecific or Specific	PRESRES Entry or MOUNT command	UNLOAD command
Public/Removable	Nonspecific or Specific	Specific	VOLUME=PRIVATE is not coded in the DD statement.	When drive is required by another volume.
Private/Removable (Tape and direct access)	Specific	Specific	VOLUME=PRIVATE is coded in the DD statement (Specific request or a nontemporary data set for tape also causes this assignment.)	After its use, unless RETAIN or PASS is coded.
Scratch (Tape only)	Nonspecific or Specific	Nonspecific or Specific	Any tape data set (Scratch volume becomes private if VOLUME=PRIVATE is coded, specific request is made, or data set is nontemporary.)	When drive is required by another volume.

1. Permanently Resident. Under normal conditions permanently resident volumes are never demounted. This may be because of a system constraint (the IPL volume and the volumes containing the system data sets), or an IPL parameter making reference to the volume. For example, the system checks to see that all permanently resident volumes are actually mounted. A list of the permanently resident volume serial numbers available to users is found in paragraph 3.2.5.

2. Reserved. A volume becomes reserved as a result of an operator MOUNT command, or by a specification at IPL time. Once mounted, the volume remains mounted until an operator UNLOAD command is issued. Volumes to be used by a group of related jobs are usually designated reserved to avoid repeated mountings and demountings. On the M&DO computers, jobs are usually batched by the volumes they require; direct-access volumes which are not permanently resident are usually reserved.

3. Removable. A removable volume is one that is neither permanently resident nor reserved. The system mounts and demounts volumes to satisfy the requirements of the jobs being processed. Refer to paragraph 5.6.6.9 for information on avoiding repeated mountings and demountings within a job. Tape volumes on the M&DO computers (as well as most scientific computing centers) are always treated as removable volumes.

17.1.2.2 Logical Volume Attributes

In addition to the physical attribute, a volume has a logical attribute, which determines a volume's availability for allocation of data sets: public, private, storage, or scratch (tapes only). These characteristics govern the system's selection of a volume when a nonspecific request is made, i.e., when neither VOL=SER nor VOL=REF is coded.

1. Public. A public volume is one on which the system can allocate space to a temporary data set when a nonspecific volume request is made, and PRIVATE is not coded as a VOLUME subparameter.

2. Private. A private volume is one on which the system cannot allocate space to a temporary data set when a nonspecific volume request is made unless PRIVATE is coded as a VOLUME subparameter. When a PRIVATE nonspecific request is made, the system requests the operator to mount a volume. This volume then assumes the characteristics of a private, removable volume. If the user desires to use an existing private volume, he must make a specific request for that volume.

3. Storage. The system may allocate a storage volume to either a temporary or permanent data set as long as VOL=PRIVATE or SER or REF are not coded. (The S/95 scratch disks, G1SCR0 - G1SCRA, are storage volumes.)

4. Scratch. A removable tape volume can be assigned the use attribute of scratch or private. The use attribute of scratch is assigned when the PRIVATE subparameter is not coded, a non-specific volume request is made, and the data set is temporary. The use attribute of private is assigned when the PRIVATE subparameter is coded, a specific volume request is made, or the data set is nontemporary. A scratch volume is demounted when its unit is required by another volume. If a data set on a scratch volume is PASSed, the scratch volume does not become PRIVATE in the receiving step unless PRIVATE is coded in the receiving step.

   If an attempt is made to KEEP a scratch volume, the disposition is changed to PASS, which may result in the volume being re-assigned to another job before being demounted.

Note: All DD cards that define tape data sets should include a DSNAME. Otherwise the system assigns a long temporary data set name which gets printed at the operator's console as part of the MOUNT and KEEP messages.

17.1.3 RECORD FORMATS

Three record formats are available:

1. F - Fixed Length Records
2. V - Variable Length Records
3. U - Unformatted Records

17.1.3.1 Spanning

Spanned records were originally a FORTRAN feature but are now supported by data management. They allow the programmer to process records which are longer than the physical blocksize. Spanning uses the record descriptor field of variable records to describe a segment of the record. Spanning may be used with either blocked or unblocked variable records, by coding an "S" in the RECFM subparameter, i.e., VS or VBS. In FORTRAN, spanning is associated with unformatted I/O. Prior to OS release 17, the FORTRAN I/O routines handled spanning. In OS release 17, support of this feature was removed from the FORTRAN I/O routines and placed in the OS data management routines. Records which have been written with older versions of FORTRAN (i.e., the executable load module was link edited with a release 16 or older FORTLIB) can be retrieved by later versions by coding an "S" in the RECFM. Since the older data management routines ignore the "S" in the RECFM, cataloged procedures containing the "S" may be used by load modules of both types.

17.1.3.2 Blocking and Buffering

Blocking is the grouping of multiple logical records into a single physical record. This reduces the number of record gaps required to separate the physical records, thus allowing more logical records to be contained on a volume. By reducing the number of accesses required, blocking saves CPU cycles needed for I/O control, avoids some WAITS, and actually saves the programmer I/O charge time since the charge is determined by the number of accesses made. Because the physical records are longer, each transmission is longer (although total transmission time is reduced) and every buffer must be larger; this requires more core.

All I/O equipment requires record gaps to separate physical records. On 9-track tape, this gap is six-tenths of an inch of erased tape (at 1600 bpi, this would store 960 bytes of data). On disk, the record gap is supplied by control characters written out with the record in unblocked format; 40 80-byte unblocked records (total 3200 bytes) can be written on a 2314 track. At full track blocking, 91 80-byte records (total 7280 bytes) can be written. On remote devices the record gap may cause line turnaround or require acknowledgement. The point is that each of these items requires resources of time or space which must be subtracted from the total resources available to the system.

Blocksizes for direct-access data sets should be chosen so that they fit the track. Refer to Table 3.6-2 to determine how many blocks will fit on each device. Two 3520-byte blocks (each containing 44 80-byte records) will fit on a 2314 track. Any increase in the blocksize (over 3520 bytes) would be regressive until the blocksize increased to 7040 bytes. There is no track limit to the blocksize for tape files, but the desirability of being able to shift from one medium to another should be considered. Tape blocks longer than $2^{14}-1$ must be handled in a special way (LRECL=X) and should be used only when justified.

Fixed and variable records may be blocked. In fixed-block format, the blocksize is a multiple of the record length. Variable records are preceded by a 4-byte record descriptor word. The LRECL is four greater than the longest record that may be written or read by the program. Variable blocks are preceded by a block descriptor word. The blocksize is at least four greater than the LRECL (i.e., eight bytes greater than the longest record) unless the records are spanned. Unblocked variable records contain both the block and the record descriptor words.

The BUFNO subparameter of the DCB parameter allows the user to specify the number of buffers to be used for a data set. The system default is two buffers per data set.

17.1.3.3 Choosing A Format

In choosing a format, the user should examine the logical records. If the records are relatively the same length, an easy, fixed format is appropriate. If not, the user should attempt to determine the length of both the longest record and the median record. If these records are nearly the same, then the fixed format may still be most efficient.

Because of the record descriptor word and the overhead in processing it, a fixed-length format has an advantage over a variable format if the records are of nearly equal size. If the records are not of equal size, the time and space wasted in padding and stripping may more than offset the inherent advantages of fixed format. If the length of the longest record greatly exceeds the length of the median record, a variable format is more efficient than a fixed format. If the length of the longest record causes the blocksize to exceed a maximum for the storage device, or if the buffer length is too long for the available core, then (variable) spanned records should be used.

Unformatted records are the equivalent of variable unblocked records without the record descriptor word. They have the advantage of saving the four bytes used and the time required for checking; however, there is no check to see that a full record has been read.

If the user is writing in ALC and using variable format, either the data must be aligned on a word boundary, or must be processed by instructions which do not require full word alignment; otherwise, specification errors will occur.

17.1.4 TO QUEUE OR NOT TO QUEUE

For the usual data management applications, the Queued Sequential-Access Method (QSAM) is recommended. QSAM relieves the programmer of the responsibility for blocking and buffering. The system code is more efficient than that of the average coder. QSAM has a look-ahead for filling and emptying buffers, thus reducing I/O wait time.

BSAM gives the programmer more control, which he should exercise, if necessary. The access methods applicable only to direct-access devices are discussed in subsection 17.2. The EXCP and XDAP methods are mentioned in subsection 17.1.9 and in The System Programmer's Guide (GC28-6550).

17.1.5 EFFICIENT USE OF CHANNELS AND ACCESS MECHANISMS (SEP AND AFF)

The path used in transferring data between an I/O device and core storage is called a "channel". In balancing and optimizing I/O operations, channel separation, channel affinity, unit affinity, and unit separation provide useful controls. Channel separation, by definition, includes unit separation. Separation and unit affinity requests are made in terms of previously defined DD statements. Channel affinity requests are made in terms of previously coded channel separation requests.

The use of separate channels for I/O operations allows for greater overlap and results in faster execution when concurrent operations are required. The use of separate units for disk operations also results in faster executions by reducing arm movement.

17.1.5.1 Channel Separation

In coding, channel separation or channel affinity is a parameter of the DD statement:

```
SEP=(ddname)

     or

AFF=(ddname)
```

where ddname refers to the name of one or more previous DD statements in the same job step.

When coding the DD cards, the user should define the most used file first, the second most used file next, and so on. An attempt may be made to put each file on a separate channel. However, with large jobs this may prove to be cumbersome, if not impossible, since there are a limited number of channels available. Consequently, subsequent files may be defined with affinity to previous separate requests. Affinity requests are a shorthand way of indicating that the previous separation is required. Channel affinity does not mean that a unit will be allocated to the same channel as that assigned to the unit in the file to which it requested affinity; rather, it means that the unit will be separated from the same files as its affinate files.

The Sort procedure provides an example of channel separation. The original input is output to a work file; in turn, it becomes input which is output to another work file, and so on. Since I/O overlap is important, the user codes, along with the other parameters:

```
//SORTIN          DD   etc.
//SORTWK01        DD   SEP=SORTIN
//SORTWK02        DD   SEP=SORTWK01
//SORTWK03        DD   AFF=SORTWK01
//SORTWK04        DD   AFF=SORTWK02
```

SORTWK03 could be coded as SEP=SORTIN.

Many compilers split their intermediate work areas into several data sets and then print and punch their output. Since a punched deck and an object module are produced, there are three output data sets. In this case, the codes are:

```
//SYSIN           DD   etc.
//SYSPRINT        DD   SYSOUT=A
//SYSUT1          DD   SEP=(SYSIN,SYSPRINT),etc.
//SYSUT2          DD   SEP=(SYSIN,SYSPRINT,SYSUT1),etc.
//SYSPUNCH        DD   SEP=(SYSPRINT,SYSUT1,SYSUT2),etc.
//SYSGO           DD   SEP=(SYSPRINT,SYSUT1,SYSUT2,SYSPUNCH),etc.
```

The use of multiple ddnames in channel separation requests on smaller machines (with fewer channels) forces the operator to reply 'NOSEP' to those requests which cannot be honored. In this case, the compiler example should be coded as follows:

```
//SYSIN           DD   etc.
//SYSPRINT        DD   SYSOUT=A
//SYSUT1          DD   SEP=(SYSIN,SYSPRINT),etc.
//SYSUT2          DD   AFF=SYSUT1,etc.
//SYSPUNCH        DD   SEP=SYSPRINT,etc.
//SYSGO           DD   SEP=(SYSPRINT,SYSPUNCH)
```

This works as long as three channels are available.

Channel separation is not possible when using the data cell; the use of unit separation, though possible, seriously degrades performance, as the bins are considered separate units.

17.1.5.2 Unit Separation

Unit separation or affinity is a subparameter of the UNIT parameter:

```
UNIT=AFF=ddname

     or

UNIT=(2314,SEP=ddname)
```

A channel or unit may have affinity to only one other ddname. Channel or unit separation may be applied to up to 8 ddname names of previous DD cards in the same job step.

Unit separation is useful for direct-access data sets which may be used concurrently. If both data sets were on the same unit, the access mechanism would be moving between the data sets as they were being accessed. If the data sets were on separate units, the arm contention is eliminated, although there may be other arm contention arising from programs being run concurrently. Since only one data set may be open on a tape, unit separation is implied for the tapes, unless affinity is requested.

In tape operations, unit affinity can be used to obtain deferred mounting. Unit affinity can only be used for devices with removable volumes. In requesting unit affinity on the 2321 data cell, the data sets will have the same unit, but may be assigned different bins. Unit affinity, by definition, includes channel affinity.

17.1.6 DATA SET PROTECTION

After completely checking out a program, it is frustrating to discover that a key data set has been lost with no back-up. The user should learn from the experience of others to protect valuable data sets.

17.1.6.1 Physical Protection

Except for card decks and listings, the average programmer does not usually have custody of his data sets. Certain precautions should be noted for those occasions when he does handle volumes.

- Cards are subject to warping and swelling when exposed to moisture or high humidity. They may also be nicked or worn by rubber bands or by handling. These conditions cause feed checks or jams. Cards should be protected by end boards (stiff cardboard), double rubber bands, boxes, trays, and cases. Even when boxes or trays are used, rubber bands should also be used; however, they should not be stretched so tightly as to damage the cards.

- Certain punches (2780 and 1442) use the same path for reading and punching. If a deck is readied when the device is selected for punching, that deck will be punched.

- Multipart listings tend to pick up smudges from handling. Certain types of paper are damaged by heat.

- Magnetic recordings consist of tapes, disks, and data cells. Their information content may be destroyed or damaged by proximity to equipment or electrical lines which emit magnetic fields. Heat and mishandling may warp the surfaces so that read heads cannot function properly. Dirt also prevents the proper functioning of heads.

- Disk packs and tapes have been destroyed in the trunks of cars that were parked in the summer sun. When transporting tapes from one building (or air-conditioned room) to another, the tape should be allowed at least three hours to reach equilibrium (24 hours, to be safe).

- File protect rings should be used. IBM tape drives have an interlock which disables the write circuits when a tape does not have a ring in place. Since FORTRAN opens tapes for INOUT, the operators stick a ring in the tape. This bypasses the protection. To prevent this, override the LABEL parameter with IN for input or OUT for output, as appropriate. See paragraph 5.6.6.7.

17.1.6.2 Logical Protection

OS has several software features for data set protection. One is the PASSWORD option. Another is the expiration date in the data set label. Both of these features require operator acknowledgement to overwrite a data set. Neither provides much protection if stand-alone or non-OS programs are used on the computer. Neither provides much protection for tape files when Bypass Label Processing (BLP) is allowed.

The PASSWORD option allows the programmer to specify write or read-and-write protection for a data set. To open a protected data set, the operator must reply with the password. Unless the programmer operates the system himself, he must make the password known to the operator. The PASSWORD data set must be built by a systems programmer. For most applications, the PASSWORD option requires more effort to use than the protection it affords.

For labeled data sets (all direct-access and standard label (SL) tapes) an expiration date may be given (see paragraph 5.6.6.7). The form LABEL=EXPDT=yyddd defines it absolutely and is preferable for data sets whose utility will cease on a given date. For most users, the form LABEL=RETPD=nnnn is preferable. The system adds the number of days specified to the current (system) date; this becomes the expiration date. Expiration dates are recommended only for Read Only files. If a file is to be updated, the operator must indicate each time a data set with an unexpired expiration date is opened for writing. If he does this often, he not only uses his time, but the reply becomes a habit. This defeats the protection. The PURGE parameter of the IEHPROGM and IEHDASDR utilities override the expiration date.

17.1.6.3 Backup

See Section 9 (Utilities) for examples of creating back-up copies of data sets.

17.1.7 ERROR OPTIONS

The EROPT subparameter of the DCB parameter allows the user to specify the action to be taken if an erroneous block is encountered and there is no SYNAD routine specified (see Supervisor and Data Management Services, GC28-6646). The actions allowed are:

1. ACC - accept the erroneous block
2. SKP - skip the erroneous block
3. ABE - ABEND task on erroneous block

The default for the EROPT subparameter is ABE.

The EROPT subparameter is coded as:

$$\text{DCB=EROPT=} \begin{Bmatrix} \text{ACC} \\ \text{SKP} \\ \text{ABE} \end{Bmatrix}$$

Suggested uses for these options are: to ABEND where a missing or erroneous record would destroy the value of a run, e.g., for compiler input, to ACCEPT, e.g., on a printed output file, or to SKIP, e.g., on the update of an ISAM or direct-access file where the missing transaction could be processed later.

Where potential errors can be anticipated and corrective action taken, an error exit (&SYNAD in the DCB in ALC, ERR in the READ in FORTRAN) should be specified. This allows the programmer to take appropriate action at the time an error occurs.

17.1.8 GENERATION NUMBERS

When a cataloged data set is updated regularly, and previous versions of the data set are to be kept, the creation of a generation data group is indicated. Refer to the Job Control Language User's Guide (GC28-6703) and the Job Control Language Reference (GC28-6704), for information on creating and retrieving generation data sets.

17.1.9 GATHER WRITE AND SCATTER READ

On some computers (1400 series), a block of data was read and written from contiguous addresses. On other computers (701X series), by coding channel commands a programmer could read a block of data and place it in non-contiguous areas or write a block of data from non-contiguous core. This feature is especially valuable when the user wants large physical blocks, without allocating a large buffer or expending CPU cycles to move edited data to a buffer.

FORTRAN and PL/I coders apparently have this facility by using lists and strings. ALC coders may obtain these advantages by coding EXCP or XDAP macro instructions. These techniques are described in the System Programmer's Guide, GC28-6550. It should be noted that it is much more difficult to write and debug EXCP and XDAP macros than to use standard access methods.

17.1.10 DISPOSITION PARAMETER; PRIVATE, SHARED, MOD DATA SETS

The Disposition (DISP) parameter reflects the status of a data set with respect to the current job step, and the disposition of the data set after the job step is completed, both normally and abnormally. If a data set has been created prior to the job step, it will have a status of OLD, SHR, or MOD, depending on the processing to be done. The SHR option allows other jobs to use the data set concurrently. If a data set does not already exist, the NEW option should be used. When a NEW data set is to be passed, the last receiving job step must specify a disposition other than PASS, or the data set will be DELETED at the end of the job. The second and third subparameters specify the disposition of the data set, depending on normal and abnormal termination, respectively. The available options are KEEP, DELETE, CATLG, or UNCATLG. PASS is available as a disposition only for normal termination of the job step.

The default values for the DISP parameter are NEW, DELETE. Note that if OLD or SHR is coded, the default dispositions are KEEP,KEEP. The default disposition for abnormal termination is to take the normal termination disposition.

When MOD is used, if the data set does not exist, it is treated as NEW; if it already exists, it is treated as OLD. Any records written are added after the last previously written record in the data set.

If a job step is updating a data set that has a status of SHR, it should ENQueue and DEQueue to obtain sole possession of the data set while it is being updated. Normally, a data set to be updated should have a status of OLD rather than SHR. However, the OS scheduler will ENQueue an OLD data set for the duration of the job, not just the one step in which OLD is encoded. For a long running job, and especially one in which a data set may occasionally be written but is normally read, it is possible to execute more than one such job concurrently by using a status of SHR and ENQueueing and DEQueueing under program control.

17.2 DIRECT-ACCESS CONSIDERATIONS

This section expands on some items applicable only to data sets on direct-access devices.

17.2.1 DATA SET ORGANIZATION (DSORG SUBPARAMETER)

The data set organizations available with direct-access devices are: Sequential (PS), Partitioned (PO), Direct (DA), and Indexed Sequential (IS).

Sequential Organization is the only one that gives complete device independence, as tapes and unit record devices only accommodate sequential data sets. In sequential organized data sets the information is processed in the order in which it appears on the device. Sequential organization gives the user the choice of queued or basic access methods.

A partitioned data set (PDS) is composed of individual members, whose names are contained in a directory index that is part of the data set. Each member has sequential organization. The chief advantages of a partioned data set are that the members can be accessed and updated individually, and that space allocation may be made for the whole data set, with a variable number of members.

The address of a data set on a direct-access device is obtained from the Data Set Control Block (DSCB) which is an entry in the Volume Table of Contents (VTOC). The address of a member of a partitioned data set is obtained from the data set directory by a BLDL macro. It is placed in the directory by a STOW macro. When a data set is scratched, its name is removed from the VTOC. The addresses of the tracks it occupies are returned to the free space queue which is in a special DSCB in the VTOC. The data set is not obliterated but its pointer is removed, making it inaccessible. When a member of a partitioned data set is deleted or replaced, its pointer is removed or altered. However, there is no free space queue in the directory. Additions or updates take place at the end. The tracks formerly used by updates or deleted members become unusable. The data set must be compressed. (See Section 9, Utilities, for a discussion of compressing a PDS.)

A direct-access data set allows the programmer to access individual records as required, rather than having to access all records preceding the one of interest. For example, to access the 50th record in a direct-access data set, the user need only specify the 50th record in the I/O statement. To access the 50th record in a sequential data set, records 1 through 49

must already have been accessed. (Refer to paragraph 17.2.3 for a discussion of FORTRAN direct-access data sets.) FORTRAN programmers may access members of partitioned data sets by coding the member name in the DSNAME parameter. In this case, the supervisor routines issue the BPAM macros when allocating and the program treats the member as a sequential data set. The LABEL parameter must be set to IN or OUT, not defaulted.

The directory of a partitioned data set corresponds to the index of an indexed sequential data set. While the directory is part of a partitioned data set, the index is separate from an indexed sequential data set. The index is usually maintained on the highest speed direct-access device available. As in the direct-access method of organization, records may be accessed selectively. Unlike the direct-access method, there need not be a one-to-one correspondence between the existing and potential keys. Records may be deleted and inserted at any point in the file. Overflow areas are provided.

The DSORG subparameter is used to specify the data set organization in the DD statement, i.e.:

$$\text{DCB=DSORG=} \left\{ \begin{array}{c} \text{PS} \\ \text{PO} \\ \text{DA} \\ \text{IS} \end{array} \right\}$$

17.2.2 SPACE DETERMINATION AND SPECIFICATION (SPACE PARAMETER)

The amount of space necessary to contain a given amount of data is primarily a function of the blocksize. For example, with logical records of 80 characters using a blocksize of 80, a track on the 2314 disk will hold 40 records; when the blocksize is equal to 7280, a track disk will hold 91 such records. Table 3.6-2 gives direct-access device characteristics that will aid in determining how many records will fit in a track.

In allocating space for a partitioned data set, allowance must be made for the directory entries. The directory space is allocated in 256-byte blocks (28 per 2314 track). Each block holds from three to seven directory entries, with an average of five entries per block for load and object modules. Source and procedure module directory entries are shorter, and up to 12 will fit in each block, depending on the presence and amount of optional data. An entry is required for each alias, as well as for each member name. Any part of the directory track left over will be allocated to the first member if the blocksize is small enough to fit in the remaining space.

If full blocking is used for the data set, the programmer may wish to allocate entire tracks for directory records, as shown in the following table:

DEVICE	DIRECTORY BLOCKS/TRACK
2301	45
2303	12
2314	17

When allocating space to data sets, the allocation may be made in terms of blocks, tracks, or cylinders. Block allocation provides device independence since the space allocated remains the same; whether it is being allocated to a disk, drum, or data cell. By using the ROUND subparameter (when allocating in blocks) an integral number of cylinders is allocated. Data sets requiring a large number of tracks should be specified with the cylinder option (CYL), as the allocation of tracks may cross cylinder boundaries, thereby degrading performance. FORTRAN DA (direct access) data sets must be allocated in blocks to agree with the DEFINE file statement.

All allocations are made in contiguous units where possible; however, if the contiguous space requested is not available, the available contiguous space is allocated and the remaining primary allocation is added as one or more secondary (overflow) allocations. This has the disadvantage of reducing the total number of secondary allocation requests (15) that can be made. The CONTIG option is available to force a contiguous allocation of tracks or cylinders.

The RLSE subparameter, which has not been reliable under previous releases, works under Release 19 and its use is encouraged. When your data set is closed unused space is given back to the system and becomes available for use by other data.

ABSTR (not shown in above format) should not be used unless the user has consulted the PAC, Building 3, Room 133A.

If the program being executed runs more efficiently with a larger space allocation (i.e., SORT/MERGE), but the user is unsure of the maximum contiguous space available, the user should code MXIG. This will allocate the largest block of contiguous space on the volume.

The SPACE parameter need only be specified when creating data sets on direct-access devices.

The SPACE parameter is coded:

$$\text{SPACE=}\left(\begin{Bmatrix}\text{TRK,}\\ \text{CYL,}\\ \text{nnnnn}\end{Bmatrix},(\text{xxxxx}[,\text{yyyy}]\ [,\text{dddd}])\ ,\text{RLSE},\left[\begin{matrix}\text{CONTIG}\\ \text{MXIG}\\ \text{ALX}\end{matrix}\right][,\text{ROUND}]\right)$$

where:

nnnnn — is the average block length in bytes

xxxxx — is the number of units required (either tracks, cylinders, or blocks).

yyyy — is the number of secondary (overflow) units required

dddd — is the number of directory blocks required - for partitioned data sets only

For further information concerning the SPACE parameter refer to the IBM manuals Job Control Language User's Guide, GC28-6703 and Job Control Language Reference, GC28-6704.

17.2.3 USE OF FORTRAN DA FACILITIES

Direct-access data sets are available to FORTRAN users; however, records must be transmitted only by FORTRAN direct-access I/O statements. Direct-access data sets must reside on direct-access volumes. The DEFINE FILE statement must be used, and the FORMAT statement must conform to the DEFINE FILE statement. For example, the following DEFINE FILE statement in a FORTRAN program:

```
DEFINE FILE 8(100,50,E,NEXT)
```

could be used with a FORMAT statement:

```
FORMAT (I2,4F12.7)
```

It is significant that the FORMAT statement must not specify a greater number of bytes than the maximum record size in the DEFINE FILE statement. On the DD statement, ddname must be:

```
FTxxF001
```

where xx corresponds to the reference number in the DEFINE FILE statement.

The corresponding DD statement could be

```
//FT08F001     DD   DSNAME=name,DISP=(NEW,KEEP),
//                  LABEL=(,SL),UNIT=DISK,
//                  VOLUME=(PRIVATE,RETAIN),
//                  SPACE=(50,100,,CONTIG),
//                  DCB=(RECFM=F,BLKSIZE=50)
```

The significant factor above is the relationship of the SPACE and DCB parameters to the DEFINE FILE statement. The SPACE parameter on the DD statement must be specified in blocks, and care must be taken that the units conform to the characteristics described in the DEFINE FILE statement.

17.2.3.1 DCB Assumptions for FORTRAN Load Module Execution

For compilation, the LRECL value for the following data sets is fixed and cannot be altered by the programmer:

Data Set	LRECL Value
SYSPRINT	120(G), 137(H)
SYSIN	80
SYSPUNCH	80
SYSLIN	80

The SYSPRINT, SYSIN, and SYSPUNCH compiler data sets can contain blocked records.

The BLKSIZE value must be an integral multiple of the corresponding LRECL value. The maximum BLKSIZE value is limited only by the type of input/output device (see Table 17.2-1).

For load module execution, specifications depend on record type. For F-type records, the BLKSIZE value must be an integral multiple of the LRECL value; for V-type records, BLKSIZE must be specified as 4+n x LRECL (where n is the number of records in the block); for U-type records, no blocking is permitted. Note that the BLKSIZE and LRECL range is limited only by the type of device used to directly write the data set. Load module DCB parameter default values are shown in Table 17.2-2.

Refer to FORTRAN IV Language (GC28-6515), and FORTRAN IV Programmer's Guide (GC28-6817) for further information.

Table 17.2-1. BLKSIZE Ranges: Device Considerations

DEVICE TYPE	BLKSIZE RANGES	
	F AND U RECORD TYPE	V RECORD TYPE
CARD READER	$1 \leq x \leq 80$	$9 \leq x \leq 80$
CARD PUNCH	$1 \leq x \leq 81$	$9 \leq x \leq 89$
PRINTER:		
120 SPACES	$1 \leq x \leq 121$	$9 \leq x \leq 129$
132 SPACES	$1 \leq x \leq 133$	$9 \leq x \leq 141$
144 SPACES	$1 \leq x \leq 145$	$9 \leq x \leq 153$
MAGNETIC TAPE	$18 \leq x \leq 32,760$	
DIRECT ACCESS:	WITHOUT TRACK OVERFLOW[1]	WITH TRACK OVERFLOW[1]
2301	$1 \leq x \leq 20,483$	$1 \leq x \leq 32,760$
2302	$1 \leq x \leq 4984$	$1 \leq x \leq 32,760$
2303	$1 \leq x \leq 4892$	$1 \leq x \leq 32,760$
2311	$1 \leq x \leq 3625$	$1 \leq x \leq 32,760$
2314	$1 \leq x \leq 7294$	$1 \leq x \leq 32,760$

1 If RECFM = V, the minimum block size is 9.

Table 17.2-2. Load Module DCB Parameter Default Values

		SEQUENTIAL DATA SETS		DIRECT-ACCESS DATA SETS	
DATA SET REFERENCE NUMBER	DDNAME	DEFAULT BLKSIZE[1]	DEFAULT RECFM[2]	DEFAULT RECFM	DEFAULT LRECL OR BLKSIZE
1	FT01Fyyy	800	U	F	THE VALUE SPECIFIED AS THE MAXIMUM SIZE OF A RECORD IN THE DEFINE FILE STATEMENT.
2	FT02Fyyy	800	U	F	
3	FT03Fyyy	800	U	FA[3]	
4	FT04Fyyy	800	U	F	
5	FT05Fyyy	80	F	F	
6	FT06Fyyy	133	UA[3]	F	
7	FT07Fyyy	80	F	F	
8	FT08Fyyy	800	U	F	
•	•	•	•	•	
•	•	•	•	•	
•	•	•	•	•	
99	FT99Fyyy	800	U	F	

1 If the records have no FORMAT control, the default LRECL is 4 less than BLKSIZE, where the default BLKSIZE is as specified in this table. For direct-access data sets, blocksize is usually limited by track capacity, unless track overflow has been specified.

2 If the records have no FORMAT control, the default RECFM is V (F if it is direct access).

3 The first character in the record is for carriage control.

17.2.4 TRACK OVERFLOW

The Track Overflow option is obtained by coding the character "T" in the RECFM subparameter. Its effect is to continue the writing of a block over a track boundary to the next track. It will not continue over a cylinder boundary or to a new extent (allocation). The spanning option(s) can be used to write records longer than a convenient blocksize, (see paragraph 17.1.2). Track Overflow is standard on the 2314, but for the 2303 and 2321, it requires a hardware feature and a SYSGEN option. For these devices it is supported on the 360/95, 360/75, and 360/65. The use of this feature for non-temporary data sets is not recommended and should be discussed with the system programmer responsible (see paragraphs 3.2.2, 3.3.2, and 3.4.2).

17.2.5 MULTI-UNIT FILES

Occasionally, a file will require more space than is available on one volume. In this case, multiple volumes must be allocated to the file. If specific volume references are made, all the serial numbers must be specified.

Multivolume files can be allocated in these ways:

```
UNIT=(device,volcount)
VOL=SER=(#1,#2,...,#n)
VOL=(PRIVATE,,,n)
```

More volumes in the VOLUME parameter than units in the UNIT parameter may be specified, but these must be coded PRIVATE. Otherwise, the volumes mounted first cannot be de-mounted during the job step. If the number of units equals the number of volumes, the user need not code PRIVATE. If specific serial numbers are coded (i.e.,[VOL=SER= ser#1,ser#2,...,ser#n]), and the data set is cataloged, all the serial numbers specified are cataloged, whether used or not. Otherwise, only those volumes actually used are cataloged. The SPACE parameter must have a secondary allocation for multi-volume data sets. BPAM and FORTRAN direct-access data sets are limited to one volume.

It is easier to retrieve a multi-volume file if it is cataloged. In this case, only the DSN and DISP parameters need to be coded. The UNIT parameter may be used to allocate multiple units for parallel processing.

17.3 TAPE CONSIDERATIONS

Tapes are, by their nature, less reliable than direct-access devices; their use should be carefully considered. Tape volumes only support sequential data sets. Since tapes may have errors, care should be taken in the programming to allow for and cope with these errors. For M&DO computers, the highest available density is 800 bpi for 7-track and 1600 bpi for 9-track.

17.3.1 9-TRACK TAPES

Nine-track tapes should be used when tapes are to be accessed from an IBM 360. Density of 1600 bpi should be specified (DEN=3). Density should always be specified, as the defaults may be different in various procedures. The blocksize should be as large as possible, but not greater than 32000 bytes. The spanned record format may be used for very long records. Record format should be F or V, as format U cannot be blocked. RECFM=FB or VB or VBS should be used.

17.3.2 7-TRACK TAPES

Seven-track tapes are used when machine independence is required, as when transferring data for use on an IBM 7094 or Univac computer. In general, the density should be the highest reliable density available on the computer receiving the generated tapes. The TRTCH option may be used to specify data conversion (C), even parity (E), translation (T), or even parity and translation (ET). Care must be taken that the labels, if used, have the same parity as the data. Again, the programming should include handling for errors. Only sequential data set organization is possible on tape volumes.

The Convert/Deblock routines, described in subsection 20.3, will prepare binary input from a 7-track tape by expanding the 6-bit byte to an 8-bit byte (deblocking); the convert routines will convert the buffers returned by the deblock routines to the S/360 structure.

17.3.3 INTERNAL TAPE LABELS

Standard labels should be used whenever possible, both to avoid operational errors, and to save time (and possible errors) in JCL. When standard labels are used, the DCB subparameters, specified when the data set was created, are incorporated in the label. Refer to paragraph 5.6.6.7 for coding of the LABEL parameter.

17.3.4 MULTIFILE REELS, MULTIREEL FILES

When multifile reels are used, care must be taken to specify the same density and parity for all files. Standard labels should be used insofar as possible. Because multifile tape volumes cannot be updated, they are primarily useful only for archives or libraries. The relative file position must be specified in the LABEL parameter (see paragraph 5.6.6).

MULTIREEL files can be created in at least the following four ways:

- UNIT=(device,volcount,DEFER)
- VOL=SER=(ser#1,ser#2,...,ser#n)
- VOL=(PRIVATE,,,volcount)
- VOL=(,,,volcount)

If standard labels are used, multireel files can be retrieved as a single data set. If not, each reel must be treated as a concatenated data set.

If the data set is cataloged, it may be retrieved without specifying the volume parameter. The UNIT parameter is necessary only if parallel processing is desired. If only one unit is allocated, the volumes are mounted in sequence on that unit. If more than one unit is allocated, the next volume will be processed while the prior volume is unloading. If "P" is specified for the number of units, as many units as volumes are allocated.

SECTION 18

MACHINE INDEPENDENCE

18.1 COMMON CONFIGURATION SUBSET

In order to achieve a certain degree of machine independence among M&DO 360 computers, the subset of devices common to the 360/95, 75, and 65 should be used. The program should also fit into the core requirements of the 360/65, which has the least main storage. Table 18.1-1 shows the common configuration subset and the storage devices available on each of the computers.

18.2 PHYSICAL TRANSFER OF DATA SETS

Programs using data sets located on direct-access storage devices on one computer could have the data sets unloaded to tape and restored to another computer, as needed. Smaller data sets may be in card form. It is not advisable to move a 2316 disk pack between computers unless the transfer involves many data sets and the user has ascertained that:

1. A drive will be available on which to mount the pack.

2. No users will request the pack while it has been removed.

When transferring the volume to another M&DO S/360, the operator must be notified that the volume is available, so that the job is not cancelled when the volume is requested. To transfer a data set from a public or non-removable volume, the utility IEHMOVE can be used (see Section 9, User Utilities).

When transferring data sets or programs to a computer other than a 360, the user should be aware of possible limitations imposed by that computer, i.e., the conversion of EBCDIC source statements or data to BCD or other character sets; the absence of 9-track tape drives, and therefore the conversion of 9-track tapes to 7-track, with a possible limitation of 200 or 556 BPI; the difference in word lengths (32 bits for the 360 computers, 36 bits for the 7094); and differences between the languages acceptable to the compilers (see Section 6 for a discussion of FORTRAN language differences).

Section 20, Conversion Aids, describes routines available for conversion of data sets from the IBM 7094.

Table 18.1-1. Configuration Summary

	Common Subset	65	75	95
Core	1536K	1536K	2048K	5120K
2314 Disk Units	2	2	2	3
Drum	1	1	2	2
Data Cell	0	0	1	1
7-Track Tape	2	2	4	4
9-Track Tape	4	4	4	8
PRINTER	2	2	2	5
CARD Reader	1	1	2	2
PUNCH	1	1	1	2
2250/2260 Displays	2	6	2	10

18.3 DIFFERENCES IN RUN PRIORITY DETERMINATION AND SET-UP RESTRICTIONS

18.3.1 JOB STREAM MANAGER (JSM)

The Job Stream Manager (JSM) is available on the 360/95, Orbit 360/75 and the 360/65. The function of JSM is primarily that of placing jobs into job classes based upon the resources required by each job. Initiators are then started to job classes in such a mixture that all of the resources of the machine (e.g., tape drives) are allocated to actively running jobs, without having jobs waiting in the initiation stage because they require resources that are currently allocated to other jobs. Waits of this kind prevent additional jobs from entering initiation and, consequently, make it most difficult for the machine to process the maximum of 15 jobs concurrently. In performing its function, JSM also includes the assignment of internal priorities based upon estimated running time.

JSM is not a standard feature of OS/360. It was acquired from another installation and further modified locally to adapt it to the GSFC environment. It has been our experience that each new release of OS/360 required extensive modifications to JSM to permit it to be carried over to the new release. Also, IBM has included the System Management Facility (SMF) in recent releases of OS/360 (JSM predates SMF).

18.3.2 GSFC JOB STREAM MANAGER

In order to minimize the maintenance problem, as well as to extend JSM to include features not present in the original version, JSM has been rewritten. The new version of JSM interfaces with OS/360 through "SMF exits," which IBM is committed to maintain as standard system features.

The new JSM is capable of assigning up to 36 job classes, although only 26 are used at present. There have been configuration changes (particularly, the number and type of tape drives) since JSM was originally installed. The new JSM has been carefully contrived to accommodate, in optimal fashion, the current tape configuration. The new JSM also takes the memory requirement of each job into account in classifying jobs, whereas the original JSM did not. JSM will use the maximum REGION required by any step of a given job, whether explicitly stated in the JOB card or an EXEC card, or implied as a default value through the use of cataloged procedures. The various job classes and the conditions under which jobs are assigned to these classes for the new JSM are shown in Table 18.3-1. The assignment of priorities has been somewhat divorced from the new JSM, and this function is now part of the Reader/Interpreter. Priority assignments have also been reviewed, and the prevailing situation is shown in Table 18.3-3.

As with the original JSM, initiators are started to certain job classes and, in the event that a given class is depleted of jobs, there are defaults so that an initiator does not unnecessarily remain idle. Also, since there can be a maximum of only 15 initiators, and there are (presently) 26 classes to which jobs may be assigned, there are certain job classes which can be reached only through this default action. Table 18.3-2 lists the primary and default classes to which initiators are started, provided the system is brought up with the console command provided for that purpose. The table is to be read from left to right. That is, on the first line, the initiator first looks at class M. If there are no jobs in class M, it next defaults to class D, then to O, etc. The figures to the right in the table summarize the 7- and 9-track tape requirements, respectively, for the primary and first default class for each initiator.

Jobs within a class are run on a priority basis, i.e., the job having the shortest total estimated run time will be run first. The relative position of jobs within a class waiting to be executed will change as new jobs are introduced into the class. The run time estimate is supplied by the programmer on the job card where estimated CPU and I/O time is inserted. A job that exceeds the shorter of the two estimates will "time out" and be cancelled by the system with a 322 ABEND. Therefore, although one may desire to insert a low run time to obtain a higher priority, it is important that the estimate be fairly accurate to avoid the necessity of rerunning the job. See subsection 5.3.1, GSFC Job Card Format.

Set-up restrictions on the 95 are that jobs requiring more than 15 minutes of computer time or more than 700K of storage will not be run during the day shift. Also, there are differences in the procedure libraries on the models 95 and 75: (1) the program referred to by a name, such as FORTRAN, refers to a FORTRANG procedure on the 95, and to FORTRANH on the 75, and (2) the default parameters can be different for the various procedures.

Table 18.3-1. S 360/95 Job Stream Manager Class Assignments

CLASS	RESOURCE RESTRICTIONS 9-Track	7-Track	CORE
A	0	0	400K
B	0	0	700K
C	1	0	400K
D	1	0	700K
E	2	0	400K
F	2	0	700K
G	3	0	400K
H	3	0	700K
I	0	1	400K
J	0	1	700K
K	0	2	400K
L	0	2	700K
M	1	1	400K
N	1	1	700K
O	2	1	400K
P	2	1	700K
Q	3	1	400K
R	3	1	700K
S	3	2	400K
T	3	2	700K
U	1	2	400K
V	1	2	700K
W	SYSTEMS		
X	2	2	400K
Y	2	2	700K
Z	(NONE OF THE ABOVE & ALL OVER 700K)		

Table 18.3-2. Initiator's Order of Searching Job Class Queues

INITIATORS	PRIMARY CLASS		SECONDARY CLASS	
	7-TRACK	9-TRACK	7-TRACK	9-TRACK
MDOFQUXS	1	1	0	1
IBKABZ	1	0	0	0
KIJMUR	2	0	1	0
DCAEBF	0	1	0	1
CDAEBF	0	1	0	1
CIJDG	0	1	1	0
EDFGKLA	0	2	0	1
EDABIJ	0	2	0	1
ACIJHX	0	0	0	1
ABFKLY	0	0	0	0
BEGOP	0	0	0	2
CABNQ	0	1	0	0
AIBJUV	0	0	1	0

Table 18.3-3. Priority Within a Job Class Queue

PRIORITY	TIME (Minutes)
11	≤1/2
10	≤1
9	≤2
8	≤3
7	≤5
6	≤8
5	≤12
4	≤15
3	≤20
2	≤25
1	>25

Note: The 1/2 minute time estimate is not available on the 360/75's and 360/65 in building 14.

18.4 RUN TIME ESTIMATES FOR DIFFERENT IBM 360 MODELS AND TIMING DIFFERENCES BETWEEN LCS AND MAIN MEMORY

Although the IBM models 65 and 75 both have the same memory cycle time, the model 75 CPU has about 1.5 times the processing speed of the model 65 because the model 75's memory is interleaved four ways, instead of two ways, as on the model 65. The model 95 CPU cycle time is 3.3 times faster than the model 65. In practice, however, it is from 3 to 9 times faster than the model 75, making it 4.5 to 13.5 times faster than the model 65 in computing.

The primary reason that run time estimates vary is the factor introduced by the presence of Large Core Storage (LCS) in combination with main memory. Since the programmer has no control over the type of core in which his program will be loaded, the running time varies, depending on how much of the slower storage is used by the program. Occasionally, the user may notice that (because his program ran in the faster core) the program execution time has decreased. However, do not decrease the time requirements on the job card since the next run may run in the slower core and the job will time out.

Table 18.4-1 shows the timing differences between LCS and main memory on the models 65, 75, and 95.

Table 18.4-1 Comparative Memory Timing

Model	LCS Cycle Time	Main Memory Cycle Time
65	8μsec	0.75μsec
75	8μsec	0.75μsec
95	0.12μsec (THIN FILM)	0.75μsec (CORE MEMORY)

18.5 DIFFERENCES BETWEEN GSFC SOFTWARE AND OTHER INSTALLATIONS

When moving programs from one computer to another, it is advisable to consult the PAC. Useful information includes the JCL to be executed, the Stage I SYSGEN listing, a listing of the catalog, and a print of the Procedure Library (PROCLIB) of the target computer. Other helpful information includes a listing of the PARMLIB and the directory of the LINKLIB.

The Stage I SYSGEN listing will indicate the processor and system defaults, the hardware configuration, and the unit names in use. The catalog listing will be useful in determining which data sets will be directly available, and which will require volume references. The PROCLIB listing is necessary to determine if the same cataloged procedures are available

and if there are differences in them such as default options, data definition parameters, etc., which must be overridden in order to be able to use them. The PARMLIB (PRESRES member) will show which volumes may be mounted. The directory of the LINKLIB will show which programs and versions of processors are available.

Areas of difference between computer centers will be found in:

- Job statement format, including use of priority and job class information
- Procedure names for standard processors
- JCL and compiler defaults
- Generic and derived unit names
- Job libraries and compiler run time libraries
- OS release -- this may not be the same as the one in use at GSFC

18.5.1 JOB STATEMENT AND ACCOUNTING DIFFERENCES

Subsection 5.3.1 describes the GSFC job card format. If another installation does not require accounting information, the GSFC job card will pass. However, if the target installation has a job card format using either the name, accounting, or programmer's ID field, the job card will have to be changed.

GSFC does not use the CLASS, PRTY, or TIME fields on the job card. For more efficient use of core the REGION field should be specified on the EXEC card and not on the job card.

Other installations may require the use of the TIME parameter on the EXEC card. See paragraph 18.5.3 for further information.

18.5.2 PROCEDURE NAMES

The IBM-supplied procedures are not in all of the procedure libraries of the M&DO computers. Even if the same name exists, there may be some differences. For standard functions (compile and link), these differences will be small and can usually be overcome with override cards.

For execution on another computer system, application oriented procedures would have to be included as part of the job being submitted. If symbolic parameters are not used (or are removed), the procedure can be entered in the job stream. If the application is to be run frequently on the other computer, the procedure should be entered into that system's procedure library. Systems using OS Release 19 or later may enter procedures, including symbolic parameters, in the job stream. See paragraph 18.5.5 for another way to accomplish this.

18.5.3 DEFAULTS

The IBM Programmer's Guides to Link Edit, FORTRAN, PL/I, ALC, etc., list certain options as defaults. These default values may be overridden in the procedures that execute them. Many of these default values may be set at SYSGEN time, and at GSFC, several of them have been set to values different than those shown in the IBM documentation. This may be true at other installations. To be safe, all applicable parameters should be stated explicitly.

The reader procedure (see subsection 11.7) is an important source of default information for system operation. Among the default values which are supplied are:

PARAMETER	IBM SUPPLIED VALUE	OVERRIDE
PRIORITY	01	PRTY=nn[1]
TIME	30 minutes	TIME=nnnn[1]
SYSOUT PRIMARY	50 tracks	SPACE=(units,(primary, secondary))[2]
SECONDARY	10 tracks	SPACE=(units,(primary, secondary))[2]
REGION	50K	REGION=nnnk

18.5.4 GENERIC AND DERIVED UNIT NAMES

Users have been cautioned elsewhere as to the liabilities in using specific unit addresses. Most IBM 360 systems have a card reader at 00C and a printer at 00E. These are nearly always used for SYSIN and SYSOUT, and therefore are not available to applications programs. For other devices, it is the exception for two devices to have the same specific address on different systems.

Derived unit names are a SYSGEN option. Some, such as DISK, are almost standard. Others, such as 2400-9, may only be used at a particular installation. Even when two installations have the same derived unit names defined, they will often include different units in the collection. At one installation, DISK may refer to 2314s, at another to 2311s. On the other hand, jobs which refer to UNIT=DISK will be acceptable if SPACE is allocated in blocks and the DCB parameters are acceptable.

Generic unit names will define the same device on any 360 to which the device is attached. For example, 2400-4 is always a dual density tape drive. For a list of the derived names in use at GSFC, refer to paragraph 19.2.2.

[1]On the M&DO computers TIME and PRTY are determined by the Job Stream Manager (JSM).

[2]On each DD defining a SYSOUT data set.

18.5.5 JOB AND MODULE LIBRARIES

When moving an application program to another computer system, if the JCL makes any specific volume references (VOL=SER=xxxx or VOL=REF=xx), either these volumes must be moved or the references changed to coincide with those of the new computer system. If the application is to be executed, the executable load module and permanent data base (if any) will be required. If the application program is to be maintained as well, then source modules and procedures may also be required.

Paragraph 9.2.1 on the IEHMOVE utility provides an example of moving a load module. Since IEHMOVE can copy both sequential and partitioned data sets, all data sets necessary to execute an application program can usually be copied in one operation. Another possibility is that all necessary data sets be located on one or more removable volumes which can be transported between computers or dumped and restored, using IEHDASDR.

In the DODS system, for example, the permanent data base is located on one disk pack. The system data sets are on another disk pack. To move the DODS application to another computer, one would transport these two disk packs and make two modifications to the operating system. These modifications connect the DODS procedure library and the DODS catalog to the operating system.

Certain procedures and OS features may be added by an installation. The formatted dump is a GSFC feature. MAPDISK or ADDTOLIB may not be available at another installation.

18.5.6 OS RELEASE DIFFERENCES

At this time, IBM issues a new release of OS every nine months. The acceptance of a new release for the M&DO 360 computers depends upon such factors as:

1. Impact upon the user, e.g., what changes have to be made to existing programs?

2. Improvements to be gained in systems performance.

3. Improvements to be gained in software features (additional facilities, corrections, or improvements to existing facilities).

Before a new OS release becomes the standard release, the M&DO systems programmers go through a rather complex three-phase checkout procedure to insure that the release has little, if any, detrimental effect upon the user. Other installations may update to every or every other release; they may freeze at a given release; or they may obtain an advanced system from IBM on a pre-release basis.

Since new features are added to OS at each release, and old features may be modified or dropped, OS differences between installations must be taken into account. Installations also modify OS features. Some of these modifications are transparent, i.e., the user does not notice them. Others, such as the elimination of the RLSE or BPL parameters, may cause a JCL error or an inappropriate action if the user is unaware of the change.

18.5.7 OS OPTION DIFFERENCES

In addition to configuration and procedural differences, another installation may not be using OS/MVT as the control program. The other possibilities in an OS System are OS/MFT and OS/PCP.

Areas of concern in converting to a non-MVT OS are:

- Region size - MFT has regions of fixed size, called partitions. Most installations using MFT or PCP do not have more than 512K bytes of core (depending on the S/360 model). PCP does not have regions or partitions, since it only runs one job step at a time.

- Multitasking - PCP does not support multitasking; MFT and MVT both provide multitasking facilities, but differ in usage. The associated macro instructions, for example, have different formats and different effects, depending on whether MVT or MFT is being used. For details on these differences, refer to the IBM manuals Concepts and Facilities (GC28-6535) and OS/360 Planning for Multiprogramming with a Fixed Number of Tasks MFT (GC27-6939).

- SYSIN data sets - PCP only allows one input stream data set per job step, which must be the last data set in the job step and must be followed by a /* card.

In general, higher level language programs will be more successfully transferred to an MFT or PCP system; the less complex the program, the better chance it has of being successfully transferred. For assistance, the user should consult the PAC (Building 3, Room 133A, extension 6768) and the system programmers at the new installation.

SECTION 19

GSFC STANDARDS

19.1 PROCESSORS AND PROCEDURES

In September, 1968, the GSFC Standards Group implemented a set of standard cataloged procedures on the T&DS (now M&DO) computers. These procedures use the standard IBM processors (such as compilers, assemblers, and Linkage Editor). The procedures are not IBM-supplied procedures but are unique to the GSFC computer facility. However, standard IBM rules for using cataloged procedures and for overriding DD cards within a procedure are applicable to the GSFC standard procedures. The rules are described in the IBM manuals Job Control Language User's Guide (GC28-6703) and Job Control Language Reference (GC28-6704).

19.1.1 GSFC STANDARD RULES

The following rules were adopted to define the standards for GSFC procedures:

- The standard name for a language processor consists of a name and and level (such as FORTRANH, FORTRANG, and PL1F).
- The general name for a processor is the one that is most desirable for a particular computer (i.e., FORTRAN refers to FORTRANG on the model 95 and to FORTRANH on the model 75).
- Stepnames for compile, link, and execute steps have the names SOURCE, LINK, and GO, respectively.
- The procedure name and stepname for most small procedures are the same.
- All DD cards in a procedure are in alphabetic order, except in a case, for example, when it is to be overridden with DD * by the user.
- All procedures use GSFC standard unit names.

- For all FORTRAN programs, the following are the GSFC standard data sets:

	Reference Number	DD Name
Card input	5	FT05F001
Printed output	6	FT06F001
Punched output	7	FT07F001

19.2 UNIT NAMES

There are three forms of unit names - specific, generic, and derived. Specific names refer to individual devices such as an actual address. Generic names refer to all devices of the same type. Generic and specific unit names are automatically produced at system generation for the I/O devices and their addresses, respectively. Derived unit names are specified by the installation at system generation to group or subdivide classes of units. Refer to Section 17 for a more complete discussion of the unit parameter.

19.2.1 GENERIC UNIT NAMES

A generic unit name should be used for old data sets and also for new data sets which are to be cataloged or placed on a specific volume. Generic unit type numbers are automatically established at system generation. Each unit type number has the format of either nnnn or nnnn-x, where nnnn is the basic unit type and -x is used to indicate a variation of the basic unit (e.g., 2400 for any 9-track tape drive or 2400-4 for a 9-track, dual-density tape drive).

The following generic names are available on the GSFC computers where the corresponding unit is attached:

Unit Type	Unit
2400	2400 series 9-Track Magnetic Tape Drive that can be allocated to a data set written or to be written in 800 bpi when the dual-density feature is not installed on the drive or in 1600 bpi when the dual-density feature is not installed on the drive.
2400-1	2400 series Magnetic Tape Drive with 7-Track Compatibility and without Data Conversion.
2400-3	2400 series 9-Track Magnetic Tape Drive that can be allocated to a data set written or to be written in 1600 bpi density.
2400-4	2400 series 9-Track Magnetic Tape Drive having an 800 and 1600 bits-per-inch (density) capability.

Unit Type	Unit
2301	2301 Drum Storage Unit
2303	2303 Drum Storage Unit
2314	2314 Disk Storage Facility
2321	any bin mounted on a 2321 data cell drive.
2250-1	2250 Display Unit, Model 1
2260-1	2260 Model 1 Display Station (Local Attachment)

A more detailed description of the IBM unit names may be found in the IBM manuals Job Control Language User's Guide (GC28-6703) and Job Control Language Reference (GC28-6704).

19.2.2 DERIVED UNIT NAMES

Derived unit names are defined by system programmers when the system is generated. They are used to spread nonspecific allocation requests over as large a number of devices as possible. Due to a system limitation, derived unit names should not be used when cataloging procedures. They should not be used with a specific volume request, because the system may allocate a unit that cannot support the volume. They also should not be used in utility control statements. In all other cases, derived unit names may be used when allocating new or temporary data sets.

The recommended derived unit names are listed below and should be used where applicable:

9TRACK, 2400-9	-	Any 9-track magnetic tape drive
7TRACK, 2400-7	-	Any 7-track magnetic tape drive
DISK	-	2314 direct-access storage facility
CELL	-	Any cell mounted on the 2321 data cell drive
DRUM	-	Designates the 2301 or 2303 drum storage unit attached to the particular computer
2250	-	Designates the 2250-1 or 2250-3 display units attached to the particular computer
2260	-	Designates the 2260-1 or 2260-2 display units attached to the particular computer

19.2.3 SPECIFIC UNIT NAMES

Generic and derived unit names indicate only a device type which has certain capabilities and functional characteristics. To specify a particular unit, a three-character device address is required. An example of this address is UNIT=0D5 for channel 0, control unit D, unit 5. To request a specific bin on a specific 2321, code UNIT=address/bin, where bin is a numeric digit (0-9).

Specific unit names should only be specified for the initialization option of a IEHDASDR utility or for hardware checking. Improper use of this parameter, such as specifying a unit which is not available, causes a wait for the device and increases the chance of cancellation. For direct-access devices, and the data cell in particular, if the volume requested is not mounted at the time of request or is mounted on another device, cancellation will normally result.

19.3 GSFC STANDARD CATALOGED PROCEDURES

The procedures discussed in the following paragraphs are the most commonly used procedures in SYS1.PROCLIB. It is important that the user be familiar with what they are, what they do, and how to use them. The procedures are periodically reviewed for accuracy and efficiency and are updated when changes are necessary due to changes in the software, configuration, etc.

The procedures listed in this section are taken from SYS1.PROCLIB on the 360/95 and are provided for illustrative purposes only. The user should always refer to a current listing of the PROCLIB before using a cataloged procedure.

The PAC in Building 3, Room 133A maintains current listings of the procedure libraries in the M&DO computers. When using two different systems, the appropriate procedures must be compared to insure that any differences will not affect execution of the program.

A permanent data set name should be used for data sets on tape; otherwise, the Operating System will create a unique name of up to 44 characters. This 44-character name takes approximately 2.5 to 3 seconds to be printed on the operator's keyboard, as opposed to .5 seconds for the standard eight-character data set name.

The following rules apply when overriding, adding, or nullifying parameters in a cataloged procedure step. The overrides are not permanent. They remain in effect only during the current execution of the program and pertain only to the program doing the overriding. The cataloged procedure in the system PROCLIB remains unchanged.

- The DD statements must have the name field of the form procstepname.ddname, e.g., GO.SYSIN.
- Overriding DD statements must be in the same order in the input stream as the corresponding DD statements in the cataloged procedure. Otherwise, the statements which are out of sequence will be added to the end of the procedure step and the first (original) DD statement will be used.
- DD statements to be added must follow overriding DD statements for the step; the ddname used must be different from the other ddnames in the procedure step.

19.3.1 COMPILER PROCEDURES

The four major GSFC standard compiler procedures and their applications are:

- ASEMBLRF -- executes the F level Assembler
- FORTRANG -- executes the G level FORTRAN compiler

- FORTRANH -- executes the H level FORTRAN compiler
- PL1F -- executes the PL/I F level compiler

Each of these procedures executes the IBM-supplied processor for the language indicated.

Refer to subsection 6.2 for a description of the Language Processors associated with the GSFC standard compiler procedures.

The minimum coding required to execute each of these procedures is:

```
//stepname      EXEC       procedurename
//SOURCE.SYSIN DD          *
      (source deck(s))
```

where:

- procedurename is the name of the cataloged procedure to be executed.
- The input is in card form in the input stream.

19.3.1.1 Input

The input to each of the compiler procedures is defined by the SYSIN DD card. Input may be in card form in the input stream, card images on tape or disk, or as a source module passed from a previous step.

An example of input on tape is:

```
//stepname      EXEC       procname
//SOURCE.SYSIN DD          UNIT=9TRACK,VOL=SER=xxxxxx,
//                         LABEL=(,SL,,IN),DISP=OLD,DSN=yyyyyy
```

For input on disk, the SOURCE.SYSIN statement must be changed to:

```
//SOURCE.SYSIN DD   DSN=name,UNIT=DISK,VOL=SER=xxxxxx,DISP=OLD
```

For a source module passed from a previous step, the following statement is used:

```
//SOURCE.SYSIN DD   *.STEP1.OUTPUT,DISP=OLD
```

where:

STEP1 is the name of the previous step

OUTPUT is the ddname of the statement in the previous step which defines the data set being passed

For a cataloged data set on disk, the following statement is coded:

```
//SOURCE.SYSIN DD   DSN=name,DISP=OLD
```

It is often necessary to accept input from more than one source as shown in the following example:

```
//stepname      EXEC      procname
//SOURCE.SYSIN DD         UNIT=9TRACK,VOL=SER=xxxxxx,
//                        LABEL=(,SL),DISP=OLD,DSN=yyyyyy
//              DD        *
      (source deck(s))
```

This example concatenates input from a 9-track labeled tape and from source decks in the input stream.

19.3.1.2 Output

The standard output of each of the compiler procedures is a temporary data set on disk called &&OBJMOD which has a disposition of DISP=(,PASS). This data set is normally passed to the LINK step of the LINK or LINKGO procedures; it may also be punched, written on tape or disk, or placed in a library.

For object module output on tape, the following coding is used:

```
//stepname           EXEC      procname
//SOURCE.SYSLIN      DD        LABEL=(,SL,,OUT),
//                             UNIT=9TRACK,VOL=SER=xxxxxx,
//                             DISP=NEW,DSN=yyyyyy
//SOURCE.SYSIN       DD        *
      (source deck(s))
```

To output the module as a permanent data set on disk, a permanent name and volume number are provided:

```
//SOURCE.SYSLIN      DD        DSN=name,VOL=SER=xxxxxx,
//                             UNIT=2314,DISP=(NEW,KEEP)
```

For the Assembler, the ddname is SYSGO rather than SYSLIN; the override ddname is SOURCE.SYSGO.

19.3.1.3 ASEMBLRF

ASEMBLRF is the procedure for executing the S/360 F level Assembler as follows:

```
MEMBER NAME  ASSEMBLY
ALIASES   ASEMBLRF
ALIASES   ASSEMBLR
//DEFAULT PROC NBLK=20
//SOURCE  EXEC      PGM=IEUASM,PARM=LOAD,REGION=100K
//SYSGO         DD  DSN=&&OBJMOD,SPACE=(3200,(&NBLK,10),,,ROUND),UNIT=DISK,
//                  DISP=(MOD,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSLIB        DD  DSNAME=SYS1.MACLIB,DISP=SHR
//SYSPRINT      DD  SYSOUT=A,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=7260),
//                  SPACE=(CYL,(1,1)),UNIT=(DISK,SEP=SYSGO)
//SYSPUNCH      DD  DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),
//                  SPACE=(TRK,(3,3))
//* INSERT        //SOURCE.SYSPUNCH DD DSN=&&DECK,SYSOUT=B  FOR OBJECT DECK
//SYSUT1        DD  UNIT=(DISK,SEP=(SYSGO,SYSPRINT)),SPACE=(CYL,(2,1))
//SYSUT2        DD  UNIT=(DISK,SEP=(SYSGO,SYSPRINT,SYSUT1)),
//                  SPACE=(CYL,(2,1))
//SYSUT3        DD  UNIT=(DISK,SEP=(SYSGO,SYSPRINT,SYSUT1,SYSUT2)),
//    .             SPACE=(CYL,(2,1))
```

To get a punched object deck, the following JCL is used:

```
//stepname          EXEC      ASEMBLRF,PARM=DECK
//SOURCE.SYSPUNCH   DD        DSNAME=&&DECK,SYSOUT=B
//SOURCE.SYSIN      DD        *
     (source deck)
```

Note that the instructions for getting a punched object deck are inserted as a comment in the cataloged procedure. If the user wants a deck and output on disk, the PARM field should be coded as PARM='LOAD,DECK'.

Note: The use of the PARM field on the EXEC card nullifies any information existing in the PARM field of the cataloged procedure. Therefore, LOAD must be coded in the PARM field for the normal execution of the user's assembler program.

The DSNAME parameter is required to override the DUMMY parameter in the SYSPUNCH DD statement. In OS Release 18, this causes a message "IEF6551 DSNAME INVALID WHEN SYSOUT SPECIFIED" to be printed. This is a warning and may be ignored in this case.

Only one source program can be assembled by each execution of this procedure.

There are several minor differences in the assembler procedures on the M&DO computers; none of them should normally affect the user. On the model 65, ASEMBLRF is not a valid name or alias; ASSEMBLY or ASSEMBLR must be used to execute this procedure.

19.3.1.4 FORTRANG

FORTRANG executes the FORTRAN G level compiler as follows:

```
MEMBER NAME    FORTRANG
ALIASES   FORTRAN
//DEFAULT PROC      NBLK=20
//SOURCE  EXEC      PGM=IEYFORT,REGION=200K
//SYSLIN       DD   DSN=&&OBJMOD,UNIT=DISK,SPACE=(3200,(&NBLK,4),,,ROUND),
//                  DISP=(MOD,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT     DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=120,BLKSIZE=7200),
//                  SPACE=(CYL,(2,1))
//SYSPUNCH     DD   DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),
//                  SPACE=(TRK,(10,5))
```

The output of the following compilation is a punched object module deck and the temporary data set &&OBJMOD:

```
//stepname          EXEC      FORTRANG
//SOURCE.SYSPUNCH   DD        DSN=&&DECK,SYSOUT=B
//SOURCE.SYSIN      DD        *
     (source deck(s))
```

On the model 95, the alias FORTRAN invokes the FORTRANG procedure; on models 65 and 75, the alias FORTRAN invokes the FORTRANH compiler.

19.3.1.5 FORTRANH

FORTRANH is the procedure to execute the FORTRAN H level compiler as follows:

```
MEMBER NAME    FORTRANH
//DEFAULT PROC      NBLK=20
//SOURCE  EXEC      PGM=IEKAA00,REGION=300K
//SYSLIN       DD   DSN=&&OBJMOD,UNIT=DISK,SPACE=(3200,(&NBLK,10),,,ROUND),
//                  DISP=(MOD,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT     DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                  SPACE=(CYL,(2,1))
```

```
//SYSPUNCH      DD   DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),
//                   SPACE=(TRK,(10,5))
//SYSUT1        DD   UNIT=DISK,DCB=(RECFM=FB,LRECL=105,BLKSIZE=3465),
//                   SPACE=(3465,(15,15))
//SYSUT2        DD   UNIT=DISK,SPACE=(4096,(2,1)),DCB=BLKSIZE=4096
```

To get a punched object deck and the temporary data set &&OBJMOD, the following JCL is used:

```
//stepname           EXEC      FORTRANH,PARM=DECK
//SOURCE.SYSPUNCH    DD        DSN=&&DECK,SYSOUT=B
//SOURCE.SYSIN       DD        *
     (source deck(s))
```

If only an output deck is desired, PARM='DECK,NOLOAD' is coded.

The alias FORTRAN invokes the FORTRANH compiler on the model 75 and model 65; it invokes the G level compiler on the model 95.

The region size for FORTRAN H is 300k on the models 95 and 75, and 275K on the model 65.

19.3.1.6 PL/I

PL1 is the procedure to execute the PL/I F level compiler as follows:

```
MEMBER NAME    PL1
ALIASES   PLI
ALIASES   PL1F
//SOURCE  EXEC      PGM=IEMAA,REGION=250K
//SYSLIN        DD  DSN=&&OBJMOD,DISP=(MOD,PASS),UNIT=DISK,
//                  SPACE=(CYL,(1,1)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSPRINT      DD  SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)
//SYSPUNCH      DD  DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),
//                  SPACE=(TRK,(3,3))
//* INSERT        //SOURCE.SYSPUNCH DD DSN=&&D,SYSOUT=B  FOR PUNCHED DECK
//SYSUT1        DD  UNIT=DISK,SPACE=(400,(400,80),,,ROUND),DCB=BLKSIZE=400
//SYSUT3        DD  UNIT=DISK,SPACE=(400,(400,80),,,ROUND),DCB=BLKSIZE=400
```

This procedure currently uses version 4.3 of the PL/I compiler on the models 95 and 75.

To get a punched deck, the following coding is used:

```
//stepname           EXEC      PL1,PARM=DECK
//SOURCE.SYSPUNCH    DD        DSN=&DECK,SYSOUT=B
//SOURCE.SYSIN       DD        *
     (source deck(s))
```

The standard procedure name for PL/I is defined to be PL1F, but the name actually used on the M&DO computers is PL1.

19.3.2 LINK-EDIT AND EXECUTE

There are four GSFC standard procedures used to link-edit and to execute a previously compiled program.

LINK and GO are two single-step procedures used to link-edit and to execute a previously compiled program, respectively.

LINKGO is a two-step procedure which incorporates the two functions of LINK and GO.

LOADER is a one-step procedure which is used to load-and-go with previously compiled programs. It lacks some of the capabilities of the Linkage Editor; however, it is much more efficient when the full capabilities of the Linkage Editor are not needed.

Refer to subsection 6.3 for a description of the Linkage-Editor and Loader programs associated with the LINK-EDIT procedures.

19.3.2.1 LINK

LINK is a single-step procedure which executes the 128k Linkage Editor to process previously compiled programs as follows:

```
MEMBER NAME    LINK
//DEFAULT PROC      NBLK=50
//LINK    EXEC      PGM=IEWL,PARM=(MAP,LIST),COND=(5,LT),REGION=300K
//LOADLIB      DD   DSNAME=SYS2.LOADLIB,DISP=SHR
//NEWLIN       DD   DUMMY
//SYSLIB       DD  DSNAME=SYS2.DUMMY,DISP=SHR
//             DD  DSNAME=SYS1.FORTLIB,DISP=SHR
//             DD  DSNAME=SYS2.GSFCLIB,DISP=SHR
//             DD  DSNAME=SYS2.LOADLIB,DISP=SHR
//             DD  DSNAME=SYS1.PL1LIB,DISP=SHR
//             DD   DSNAME=SYS1.SSPAK,DISP=SHR
//SYSLMOD      DD   DSN=&&LODMOD(GSFC),DISP=(NEW,PASS),UNIT=DISK,
//                  SPACE=(3072,(&NBLK,40,1))
```

```
//SYSPRINT      DD    SYSOUT=A,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=1210),
//                    SPACE=(TRK,(10,5))
//SYSUT1        DD    UNIT=DISK,SPACE=(1024,(100,20)),DCB=BLKSIZE=1024
//TAPELIB       DD    DUMMY,VOL=SER=TAPEIN,UNIT=(9TRACK,,DEFER),LABEL=(,BLP)
//                    DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSLIN        DD    DSN=&&OBJMOD,DISP=(OLD,DELETE),DCB=RECFM=FB
//              DD    DDNAME=OBJECT
```

The LINK procedure is normally executed following one of the compiler procedures. The output of the compiler (&&OBJMOD) is passed as input to the Linkage Editor and is defined by the SYSLIN DD statement.

The output of the LINK procedure is a member (GSFC) of the temporary partitioned data set &&LODMOD. This PDS has a disposition of DISP=(NEW,PASS) and is passed to the next step.

The automatic call library (SYSLIB) contains the libraries which are most commonly required by GSFC programmers. Additional libraries may be concatenated to SYSLIB by coding:

```
//LINK.SYSLIB         DD    DSN=MYLIB,DISP=SHR
```

This statement overrides the SYS2.DUMMY which is provided for that purpose. If the private library is not cataloged, the UNIT and VOL DD parameters must be supplied.

If more than one library is to be added via LINK.SYSLIB, care must be taken so that desired libraries are not deleted. For example, to add two libraries without overriding SYS1.FORTLIB, SYS2.GSFCLIB, and SYS2.LOADLIB, the following procedure would be used:

```
//LINK.SYSLIB   DD    DSN=MYLIB1,DISP=SHR
//              DD
//              DD
//              DD
//              DD    DSN=MYLIB2,DISP=SHR
```

The LINK procedure is executed by the statement:

```
//stepname            EXEC      LINK
```

Input to the Linkage-Editor may include previously compiled object decks (as card input) and the passed data set &&OBJMOD, as shown in the following example:

```
//stepname            EXEC      LINK
//LINK.OBJECT         DD        *
     (object deck(s))
```

Or object decks only:

```
//stepname          EXEC      LINK
//LINK.SYSLIN       DD        *
     (object deck(s))
```

The TAPELIB and NEWLIN DD cards are included in the procedure to facilitate the input of object modules on tape or disk. The parameters on the TAPELIB card have been coded to define tape input. When using NEWLIN, the user must code all of the necessary DD parameters. The following example illustrates the use of TAPELIB and the Linkage-Editor control cards required to include the data set identified by TAPELIB:

```
//stepname          EXEC      LINK
//LINK.TAPELIB      DD        DSN=name,LABEL=(,SL),VOL=SER=xxxxxx
//LINK.SYSLIN       DD        *
  INCLUDE TAPELIB
  ENTRY MAIN
```

Object modules on tape may be entered directly through SYSLIN by using the following override card:

```
//stepname          EXEC      LINK
//LINK.SYSLIN       DD        UNIT=9TRACK,VOL=SER=xxxxxx,
//                            LABEL=(,SL),DISP=(OLD,KEEP),DSN=yyyyy
```

In this example, the Linkage-Editor control cards are not required.

There are several differences between the LINK procedures on the three M&DO computers:

- PGM=IEWLF44 on the model 75 executes the 128k Linkage Editor.
- PARM=NCAL is specified on models 75 and 65, but not on the model 95.
- Other differences occur in the SYSPRINT blocksize and the data sets provided in SYSLIB.
- The alias LKED which is provided on the model 75 is not recommended; if this alias is used on the model 95, it will execute the IBM-supplied Linkage Editor procedure.

19.3.2.2 GO

GO is the single-step procedure which logically follows the LINK procedure. If the GO procedure is to be used immediately following the LINK procedure, the LINKGO procedure (see paragraph 19.3.2.3) is recommended. The LINK and GO are used separately when some processing is to be executed between the LINK and GO steps, or if multiple executions of a program are required. A listing of this procedure follows:

```
MEMBER NAME    GO
//GO      EXEC      PGM=*.&STEP..LINK.SYSLMOD,COND=(5,LT),REGION=100K
//FT05F001     DD   DDNAME=DATA5
//FT06F001     DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                  SPACE=(CYL,(1,1))
//FT07F001     DD   DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),
//                  SPACE=(TRK,(1,20))
//*INSERT          //GO.FT07F001 DD DSN=&&DECK,SYSOUT=B FOR PUNCHED OUTPUT
//SYSPRINT     DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629),
//                  SPACE=(TRK,(1,20))
```

The EXECUTE statement in the GO procedure points to the load module created by the LINK procedure as the program to be executed. The user must provide the program stepname which executed the LINK procedure, as shown in the following example:

```
//LINKIT       EXEC      LINK
//GOGO         EXEC      GO,STEP=LINKIT
//GO.DATA5     DD        *
     (data)
```

The program stepname LINKIT replaces the symbolic name &STEP in the PGM= statement in the GO procedure. The program to be executed is then defined by PGM=*.LINKIT.LINK.SYSLMOD.

The input to the program is a card deck in the input stream.

The GO procedure is written for the FORTRAN programmer. The names of the DD statements for input, print, and punch are the standard FORTRAN ddnames.

19.3.2.3 LINKGO

LINKGO is a two-step procedure which combines the functions of the LINK and GO procedures. It is used when the user wishes to link-edit and immediately execute the resulting load module. A listing of this procedure follows:

```
MEMBER NAME    LINKGO
//DEFAULT PROC      NBLK=50
//LINK    EXEC      PGM=IEWL,PARM=(MAP,LIST),COND=(5,LT),REGION=300K
//LOADLIB      DD   DSNAME=SYS2.LOADLIB,DISP=SHR
//NEWLIN       DD   DUMMY
//SYSLIB       DD  DSNAME=SYS2.DUMMY,DISP=SHR
//             DD  DSNAME=SYS2.DUMMY,DISP=SHR
//             DD  DSNAME=SYS1.FORTLIB,DISP=SHR
//             DD  DSNAME=SYS2.GSFCLIB,DISP=SHR
//             DD  DSNAME=SYS1.PL1LIB,DISP=SHR
//             DD  DSNAME=SYS1.TELCMLIB,DISP=SHR
//             DD  DSNAME=SYS2.LOADLIB,DISP=SHR
//             DD  DSNAME=SYS1.SSPAK,DISP=SHR
//             DD   DSNAME=USER.SUB502.FUSLLIB,DISP=SHR
//                  UNIT=DISK,VOL=SER=G1SYS1
//SYSLMOD      DD   DSN=&&LODMOD(GSFC),DISP=(NEW,PASS),UNIT=DISK,
//                  SPACE=(3072,(&NBLK,40,1))
//SYSPRINT     DD   SYSOUT=A,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=1210),
//                  SPACE=(TRK,(10,5))
//SYSUT1       DD   UNIT=DISK,SPACE=(1024,(100,20)),DCB=BLKSIZE=1024
//TAPELIB      DD   DUMMY,VOL=SER=TAPEIN,UNIT=(9TRACK,,DEFER),LABEL=(,BLP),
//                  DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSLIN       DD   DSN=&&OBJMOD,DISP=(OLD,DELETE),DCB=RECFM=FB
//             DD   DDNAME=OBJECT
//GO      EXEC      PGM=*.LINK.SYSLMOD,COND=(5,LT),REGION=100K
//FT05F001     DD   DDNAME=DATA5
//FT06F001     DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                  SPACE=(CYL,(1,1))
//FT07F001     DD   DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),
//                  SPACE=(TRK,(1,20))
//* INSERT          //GO.FT07F001 DD DSN=&&DECK,SYSOUT=B  FOR PUNCHED OUTPUT
//SYSPRINT     DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629),
//                  SPACE=(TRK,(1,20))
```

The minimum control cards required to execute the LINKGO procedure are:

```
//stepname      EXEC      LINKGO
//GO.DATA5      DD        *
     (data)
```

In this example:

- The input to the Linkage Editor is the passed data set &&OBJMOD.
- No additional libraries are required for the automatic call library (SYSLIB).
- The data is in the input stream.

Any changes or additions to the DD statements in the LINKGO procedure require additional JCL statements. The user wishing to make any changes or additions should read the comments in paragraphs 19.3 and 19.3.2.1.

Note that the EXECUTE statement of the GO step executes the load module created by the LINK step. The user does not have to code a program stepname as he does when using the separate LINK and GO procedures.

The alias LKEDG for LINKGO on the model 75 executes the wrong procedure if it is inadvertantly used on the model 95.

The LINKGO procedure on the model 95 has several SYSLIB data sets not provided on models 75 and 65. The user should be sure that all required routines are available.

19.3.2.3 LOADER

The LOADER procedure, used to execute the Loader program, is an efficient method of combining the functions of LINK and GO into one job step. It can substantially reduce the execution time of the linkage functions. Use of this procedure is recommended when the Linkage Editor control statements (such as INCLUDE, NAME, XREF, and OVERLAY) are not required; only the MAP, LET, NCAL, and SIZE options are supported. See paragraph 6.3.2 for a description of the Loader processing functions and limitations, and the IBM manual Linkage Editor and Loader (GC28-6538) for a complete description of the Loader. A listing of this procedure follows:

```
MEMBER NAME    LOADER
ALIASES   LOADR
//GO       EXEC      PGM=LOADER,PARM=(MAP,CALL),COND=(4,LT),REGION=240K
//SYSLIB         DD   DSN=SYS2.DUMMY,DISP=SHR
//               DD   DSN=SYS2.DUMMY,DISP=SHR
//               DD   DSN=SYS1.FORTLIB,DISP=SHR
//               DD   DSN=SYS2.GSFCLIB,DISP=SHR
//               DD   DSN=SYS1.PL1LIB,DISP=SHR
//               DD   DSN=SYS1.TELCMLIB,DISP=SHR
//               DD   DSN=SYS2.LOADLIB,DISP=SHR
//               DD   DSN=SYS1.SSPAK,DISP=SHR
//SYSLOUT        DD   SYSOUT=A
//SYSLIN         DD   DSN=&&OBJMOD,DISP=(OLD,DELETE)
//               DD   DDNAME=OBJECT
//FT05F001       DD   DDNAME=DATA5
//FT06F001       DD   SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                    SPACE=(CYL,(1,1))
//FT07F001       DD   DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),
//                    SPACE=(TRK,(1,20))
```

```
//* INSERT          //GO.FT07F001 DD DSN=&&DECK,SYSOUT=B FOR PUNCHED OUTPUT
//SYSPRINT     DD    SYSOUT=A,DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629),
//                   SPACE=(TRK,(1,20))
```

The minimum requirements for the loader step of a compile and execute are:

```
//stepname          EXEC      LOADER
//GO.DATA5          DD        *
     (data)
```

where the input data set (&&OBJMOD) is passed from a previous compile step and the data is in the input stream.

In addition to the data set passed by a previous compile, the Loader accepts object modules and/or load modules from either tape, disk, or the input stream. Data to be processed by the executable load module created by the Loader may be input by tape, disk, or card.

The input to the Loader in the following example is &&OBJMOD and object modules in the input stream; the input to the executable load module created by the Loader is data cards in the input stream:

```
//stepname          EXEC      LOADER
//GO.OBJECT         DD        *
     (object module(s))
//GO.DATA5          DD        *
     (data)
```

A frequent practice at GSFC is to load-and-go with object decks. The input to the problem program is data cards in the input stream:

```
//stepname          EXEC      LOADER
//GO.SYSLIN         DD        *
     (object deck(s))
//GO.DATA5          DD        *
     (data)
```

In the following example, the input to the Loader is &&OBJMOD and load modules are on disk; the input to the executable program is on tape:

```
//stepname          EXEC      LOADER
//GO.OBJECT         DD        DSN=loadpds,UNIT=2314,
//                            VOL=SER=xxxxxx,DISP=OLD
//                            DCB=BLKSIZE=3702
//GO.DATA5          DD        UNIT=9TRACK,VOL=SER=yyyyyy,
//                            LABEL=(,SL),DISP=OLD,DSN=zzzzzz
```

The user can also use override cards to specify additional data sets to be concatenated with SYSLIB and to request punched output by overriding FT07F001.

As in the LINK and LINKGO procedures, there are differences between the LOADER procedures on the M&DO computers. The SYSLIB DD statement on the model 95 contains several data sets not available on models 75 and 65.

On the model 65, the user must supply the DCB parameter for FT06F001 and FT07F001.

19.3.3 SORT

The SORT procedure executes the IBM-supplied Sort/Merge program. The procedure contains the SYSLIN and SYSLMOD DD statements which allow the programmer to specify user exits which require the Linkage Editor.

For normal Sort/Merge usage, the user must supply his own sort work areas, SYSIN DD card, and Sort/Merge control statements. A listing of this procedure follows:

```
MEMBER NAME    SORT
//SORT    EXEC      PGM=IERRCOOO,REGION=200K
//SYSOUT        DD   SYSOUT=A
//SYSPRINT      DD   DUMMY
//SYSLMOD       DD   UNIT=SYSDA,SPACE=(3600,(20,20,1))
//SYSLIN        DD   UNIT=SYSDA,SPACE=(80(10,10))
//SORTLIB       DD   DSNAME=SYS1.SORTLIB,DISP=SHR
//SYSUT1        DD   UNIT=(SYSDA,SEP=(SORTLIB,SYSLMOD,SYSLIN)),
//                   SPACE=(1000,(60,20))
//*  USER SUPPLIES "//SORT.SYSIN DD *" AND CARD INPUT AS NEEDED
```

For maximum efficiency:

- Use channel separation when possible.
- Assign SORTWK areas of equal size.
- The tracks of a work area must be contiguous.
- Do not use the DATACELL for sort work areas.
- The size of a work area is dependent on the record size, number of records, and number of work areas assigned. Refer to the IBM manual SORT/MERGE (GC28-6543) for the formula to determine work area size. A good rule of thumb is to use a total area of 1.25 times the size of the input and to divide this into three or more work areas as required.

- On the 2314 disk, all necessary devices should be assigned; however, only one work area per device should be used. For all but very large sorts, the user should limit work areas to four devices in order to prevent tying-up the system. If contiguous space is not available, the user must decrease the size of his work areas and increase the number. Because the space available for each 2314 varies continuously, no set of work areas is guaranteed to be sufficient every time.

The following example represents the general format of the coding required to execute the SORT procedure. Note that the SORTWK cards are not preceded by the procedure stepname (SORT. SORTWK01) as is usually the case when adding to or overriding a procedure. This exception is allowed only for single-step procedures, but is seldom used except with the sort DD cards. The SYSIN DD statement has been coded SORT. SYSIN, although the prefix SORT is not required here:

```
//stepname          EXEC     SORT
//SORTIN            DD       DSN=input,UNIT=2314,VOL=SER=xxxxxx,
//                           DISP=(OLD,KEEP),DCB=(RECFM=FB,
//                           LRECL=80,BLKSIZE=3200)
//SORTOUT           DD       DSN=output,DISP=(NEW,PASS),
//                           UNIT=2314,
//                           DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),
//                           SEP=SORTIN,SPACE=(TRK,(350),,CONTIG)
//SORTWK01          DD       UNIT=2314,SEP=SORTIN,
//                           SPACE=(TRK,(100),,CONTIG)
//SORTWK02          DD       UNIT=2314,AFF=SORTOUT,
//                           SPACE=(TRK,(100),,CONTIG)
//SORTWK03          DD       UNIT=2314,SEP=SORTIN,
//                           SPACE=(TRK,(100),,CONTIG)
//SORTWK04          DD       UNIT=2314,AFF=SORTOUT,
//                           SPACE=(TRK,(100),,CONTIG)
//SORT.SYSIN        DD       *
     (sort control cards)
```

Although this example is coded for work areas on disk, tape units may also be used for input, output, and work areas for sorting.

In this example, the SORTOUT data set is coded to be passed to a following step. If a permanent copy of the data set is required, it may be put on tape or disk. If placed on a private disk, the user should code the volume serial number and request that the volume be placed on a different channel from SORTIN. If SORTOUT is to be kept on a scratch pack, the user should check with the PAC to determine the channel location of each scratch pack.

The SORT procedures on models 75 and 65 do not contain the SYSPRINT, SYSLMOD, and SYSUT1 DD cards, and thus cannot handle user modifications. They are essentially different procedures under the same name.

The alias for SORT on the model 75 is SORTD. However, use of this alias on the model 95 executes a different Sort procedure--SORT has no alias on the model 95. On the model 65, SORTD is the member name and SORT is an alias.

See paragraph 6.3.3 for more information on the SORT procedure, and the IBM manual SORT/MERGE (GC28-6543) for a complete discussion of the requirements and facilities of Sort/Merge.

19.3.4 PRNTPROC

The PRNTPROC procedure executes the OS/360 utility program IEBPTPCH to print the contents of SYS1.PROCLIB. Each member is printed beginning on a new page. The input to PRNTPROC is the procedure PRTPROC2 which contains the utility control cards for IEBPTPCH. A listing of this procedure follows:

```
MEMBER NAME    PRNTPROC
//PRNTPROC   EXEC   PGM=IEBPTPCH
//SYSPRINT     DD   DUMMY
//SYSUT1       DD   DSN=SYS1.PROCLIB,DISP=SHR
//SYSUT2       DD   SYSOUT=A,SPACE=(CYL,(3,1))
//SYSIN        DD   DSN=SYS1.PROCLIB(PRTPROC2),DISP=SHR

MEMBER NAME    PRTPROC2
     PRINT     TYPORG=PO,MAXFLDS=1
     TITLE     ITEM=('** 360/95 SYS1.PROCLIB **',45)
     RECORD    FIELD=(80,1,,20)
```

To print SYS1.PROCLIB, the user needs only to code:

```
//stepname     EXEC      PRNTPROC
```

On the model 75, only selected members of SYS1.PROCLIB are printed; the complete PROCLIB on the model 65 is printed.

The utility control statements are stored in PRTPROC on the model 65 and in PRTPROC2 on the model 75.

19.3.5 ADDTOLIB

ADDTOLIB is a two-step procedure for adding object modules to a user's private library which may be used as a private automatic call library. ADDTOLIB executes the program LIBRYGN2 to search the input object decks,

to find the CSECT and entry point names, to generate alias and entry point names, and to generate alias cards and a name card for the Linkage Editor. The LINK step executes the program SAVELIBS to link the object modules into the user's library. A listing of this procedure follows:

```
MEMBER NAME    ADDTOLIB
//DEFAULT PROC      NBLK=2000
//LIBNAME EXEC      PGM=LIBRYGN2,PARM='=',COND=(5,LT),REGION=150K
//SYSLIB        DD  DSNAME=SYS2.DUMMY,DISP=SHR
//SYSOUT        DD  DSNAME=&LIBMOD,SPACE=(80,(&NBLK,2000),,,ROUND),
//                  DISP=(,PASS),
//                  UNIT=DISK,DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSPRINT      DD  SYSOUT=A,DCB=(RECFM=FB,LRECL=81,BLKSIZE=3240)
//SYSPUNCH      DD  DUMMY,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)
//SYSIN         DD  DSNAME=&OBJMOD,DISP=(OLD,PASS),DCB=RECFM=FB
//              DD  DDNAME=OBJECT
//LINK    EXEC      PGM=SAVELIBS,PARM=(MAP,LIST,NCAL),COND=(5,LT),
//                  REGION=300K
//LOADLIB       DD  DSNAME=SYS2.LOADLIB,DISP=SHR
//NEWLIN        DD  DUMMY
//SYSLMOD       DD  DSNAME=NULLFILE
//SYSPRINT      DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=3509)
//SYSUT1        DD  UNIT=DISK,SPACE=(TRK,(10,5))
//SYSLIN        DD  DSNAME=&LIBMOD,DISP=(OLD,PASS)
```

The input to the LIBRYGN2 program of the LIBNAME step is the temporary data set &OBJMOD which is created in a previous compile step. Object decks may be used in addition to, or in place of, &OBJMOD. The output of this step is the temporary data set &LIBMOD which contains the object modules and the generated alias and name cards.

&LIBMOD is input to the SAVELIBS program of the LINK step. The output is a nonexecutable load module which is stored in the data set named by the user in the SYSLMOD DD statement.

The following cards are required to execute the ADDTOLIB procedure:

```
//stepname          EXEC    ADDTOLIB
//LINK.SYSLMOD      DD      DSN=G1.userid.Lxxxxxx,
//                          DISP=SHR
```

where:

G1 = the code for the model 95 (G3 is the code for the model 75, and G2 is used for the model 65).

userid = the programmers ID.

Lxxxxxx = the user-supplied name of the data set being saved.

If object decks are used in place of &OBJMOD, the following is coded:

```
//stepname          EXEC      ADDTOLIB
//LIBNAME.SYSIN     DD        *
     (object deck(s))
//LINK.SYSLMOD      DD        DSN=Gl.userid.Lxxxxxx,DISP=SHR
```

The name card generated by the LIBRYGN2 program contains the CSECT name unless there is an entry point with the same address as the CSECT name, or unless the CSECT name contains one of the exclude list characters; the first nonexcluded name is put on the name card.

Alias cards are generated for the first five nonexecutable entry points which have not been used on the name card. To change the exclude list, use the following:

```
//stepname      EXEC      ADDTOLIB,PARM.LIBNAME='charstring'
```

where charstring is the set of characters which is used to exclude all names containing any one or more of the set. A specified character string overrides any defaults used, such as the # or = symbols.

Note: Only one member with a given name can be in a library at any time. Thus, FORTRAN main programs, which are all called MAIN by the compiler, must be renamed by recompiling, using:

```
//stepname      EXEC      FORTRAN,PARM='NAME=name'
```

There are differences between ADDTOLIB on the model 95 and model 75 in the SPACE and BLKSIZE parameters of the SYSOUT DD statement. SYSOUT has BLKSIZE=80 on the model 95, and BLKSIZE=3200 on the model 75; the difference in these should not cause any problems. ADDTOLIB is not available on the model 65.

Contact Mrs. Pat Barnes in the GSFC Program Library, Building 3, extension 6796, for further documentation on ADDTOLIB.

For assistance in using ADDTOLIB, contact Mr. Lee Foster, Building 1, Room 254, extension 6748.

19.3.6 SAVEPROG

The SAVEPROG procedure executes the SAVELIBS program to link edit programs into SYS2.LOADLIB. The input to SAVELIBS is the temporary data set &OBJMOD

created in a previous compile step and a NAME control card which uniquely identifies the member being placed in LOADLIB. The name is composed of the user's five-character registered ID and three alphanumeric characters selected by the user. The output of SAVEPROG is a nonexecutable load module stored as a member of SYS2.LOADLIB. A listing of this procedure follows:

```
MEMBER NAME    SAVE
ALIASES   SAVEPROG
//LINK    EXEC      PGM=SAVELIBS,PARM=(MAP,LIST,NCAL),COND=(5,LT),REGION=300K
//LOADLIB      DD   DSNAME=SYS2.LOADLIB,DISP=SHR
//NEWLIN       DD   DUMMY
//SYSLMOD      DD   DSNAME=SYS2.LOADLIB,DISP=SHR
//SYSPRINT     DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=3509)
//SYSUT1       DD   UNIT=DISK,SPACE=(TRK,(10,5))
//TAPELIB      DD   DUMMY,VOL=SER=TAPEIN,UNIT=(9TRACK,,DEFER),LABEL=(,BLP),
//                  DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSLIN       DD   DSNAME=&OBJMOD,DISP=(OLD,DELETE),DCB=RECFM=FB
//             DD   DDNAME=OBJECT
```

Input to SAVEPROG may be the passed data set &OBJMOD, object decks, or object modules on tape. At this point, the reader should refer to paragraph 19.3.2.1 to review the LINK examples and to compare the LINK and SAVEPROG procedures.

The input process is the same for both procedures; the output of LINK is an executable load module stored in &&LODMOD, while the output of SAVEPROG is a nonexecutable load module saved in SYS2.LOADLIB.

The following example illustrates the use of SAVEPROG after a FORTRAN compile:

```
//stepname          EXEC      FORTRAN
//SOURCE.SYSIN      DD        *
     (source deck(s))
//stepname          EXEC      SAVEPROG
//LINK.OBJECT       DD        *
     NAME      userid xxx(R)
```

There are differences between the SAVE procedures on the three M&DO computers in member and alias names, region size, SYSLIB DD statement, the program to be executed, and the execution parameters.

The region size is 300k for the model 95 and 200k for the models 75 and 65.

The model 65 SAVE procedure executes the 128k F level Linkage Editor. (IEWLF128) contains a SYSLIB DD statement and specifies PARM=(XREF,LET, NCAL) on the execute card.

The SAVE procedure on models 95 and 75 executes the program SAVELIBS, has no SYSLIB DD statement, and specifies PARM=(MAP,LIST,NCAL) on the EXEC card.

Contact Mrs. Pat Barnes in the GSFC Program Library, Building 3, extension 6796, for further documentation on SAVEPROG.

For assistance in using SAVEPROG, see the PAC in Building 3.

19.3.7 BB

BB is the three-step procedure to execute the Boole and Babbage Problem Program Analyzer which is used to measure and analyze Problem Program Efficiency (PPE). Descriptions of the Boole and Babbage System and the cataloged procedure BB are presented in subsection 7.2.

19.3.8 FAPCON

FAPCON is the procedure used to execute the FAPCON program which converts S/360 FORTRAN IV programs in EBCDIC or BCD from single precision to double precision. A description of this procedure and its usage is presented in subsection 20.2.

For further documentation, contact Mrs. Pat Barnes, in the GSFC Program Library, Building 3, extension 6796.

19.3.9 FORMAC

FORMAC is a cataloged procedure used for executing the FORMAC processor, an extension of the PL/I compiler which provides the capability to perform formal algebraic manipulation of variables and expressions within a PL/I program.

See subsection 7.3 for a description of the FORMAC program and the cataloged procedure, FORMAC, used for executing this program.

Users desiring more detailed information about FORMAC should contact Mrs. Pat Barnes in the GSFC Program Library, Building 3, extension 6796.

19.3.10 GPSS V

GPSS V is a procedure for execution of the General-Purpose Simulation System, applicable where the model deals with discrete items, events, and times.

Input to GPSS is a formatted data deck representing a special block diagram coded by the user.

Refer to subsection 7.5 for a description of the GPSS program and the cataloged procedure, GPSS, used for executing this program.

For more complete information on GPSS, contact Mrs. Pat Barnes in the GSFC Program Library, Building 3, extension 6796.

19.3.11 RELEASE

- Description. Use of this procedure in conjunction with a JOB card entry TYPRUN=HOLD allows a programmer to run a group of jobs in a specified sequence, such as when all programs in the group of jobs must share some nonsharable resource (same tape volume, DSNAME, etc.). To avoid having the OS initiators select more than one of this group of jobs simultaneously and thus cause allocation problems, the RELEASE procedure should be used.

- Control Cards, Parameters, etc., To Be Used With This Cataloged Procedure.

  1. On the JOB card for each job, except the first, punch TYPRUN=HOLD (preceded by a comma) after all other entries.

  2. At the end of each job, except the last, place this card at the end of the deck to release the next job from the hold queue:

     ```
     //LASTSTEP EXEC RELEASE,COND=EVEN,PARM=xxx
     ```

  where xxx represents the last three characters of the jobname of the next job to be run, which is the job to be released from the hold queue by the preceding job. Each job in the sequence must have the same user ID.

An example of the deck setup for three successive jobs to be run with RELEASE would be:

First job:

```
//usrid001 JOB (........).....
//..(required JCL cards).....
     (program)
/*
//LASTSTEP EXEC RELEASE,COND=EVEN,PARM=002
```

Second job:

```
//usrid002 JOB (.....)....,TYPRUN=HOLD
//...(required JCL cards).....
     (program)
/*
//LASTSTEP EXEC RELEASE,COND=EVEN,PARM=003
```

Third (last) job:

```
//usrid003 JOB (.....)....,TYPRUN=HOLD
//...(required JCL cards).....
     (program)
/*
```

19.3.12 JOB SUBMISSION SLIP WITH RELEASE

The user who submits multiple jobs with the RELEASE procedure must submit the jobs as a group, with a comment on each submission slip calling the dispatcher's attention to use of this procedure. (CAUTION: It is a wise precaution to mention the use of this method to the dispatcher or operator at the time of job submission. Otherwise, the results may be unpredictable.)

19.3.13 ASSEMBLER G

Assembler G is available on the 360/95, 360/75 and the 360/65. It may be called by executing the cataloged procedure ASMG. Some advantages of Assembler G are:

1. It runs in batch mode: that is, one can assemble more than one CSECT per step.

2. It is completely compatible with Assembler F but faster.

3. It can assemble and execute in the same step, provided there are no external references (V-type ADCON) in the program to be assembled.

4. It has a higher SYSUT1,2,3 blocking than Assembler F, therefore, less I/O charge.

5. One can define his own instruction set if he so desires.

EXAMPLE

```
//JOB CARD
//Stepl       EXEC    ASMG
//SYSIN         DD    *
One         CSECT
              .
              .

              .
              .
            END
Two         CSECT
              .
              .

              .
              .
            END
.
.
.
N           CSECT
              .
              .

              .
              .
            END
```

The GSFC Standards Committee has decided that NODECK be the default option. Therefore, to obtain an object deck code PARM=(DECK=DECK) on your EXEC card.

```
//Stepl   EXEC      ASMG,PARM=(DECK=DECK)
```

SECTION 20

CONVERSION AIDS

Conversion aids are used to make programs or data created for one computer system acceptable to another computer system, or to convert source programs written in one language to another language.

These aids are particularly useful in converting data from the 7090/94, which uses a 36-bit word, to a form acceptable to the OS/360, which has a 32-bit word. Other programs assist in converting 7090/94 FORTRAN to S/360 FORTRAN IV, single-precision FORTRAN programs to double-precision, and FORTRAN IV source statements to PL/I. Some of these programs also edit and renumber FORTRAN source programs.

The various programs are highly restricted by the type of input that is acceptable and by the extent of conversion they will perform. They usually indicate those statements which must be hand converted. For more complete documentation, the user should contact Mrs. Pat Barnes (extension 6796) in the GSFC Computer Program Library in Building 3, Room 133.

20.1 DATA STATEMENT SIFT PROGRAM

SIFT is a conversion aid for installations converting from 7090/94 FORTRAN to OS/360 FORTRAN IV. SIFT's main function is to convert a DATA statement containing an implied DO into a DATA statement(s) acceptable to OS/360 FORTRAN IV. The program also flags any DATA statements which contain hollerith or actual initialization values.

SIFT is available on the IBM model 95 in SYS1.LINKLIB.

20.1.1 INPUT/OUTPUT

The input to SIFT is a 7090/94 BCD FORTRAN source deck. The output contains a listing and a resequenced BCD deck.

The listing produced by SIFT shows the records read and generated by SIFT. Each input record has a number appearing to the left of it; each record generated or reproduced by SIFT has a number appearing to the right of it. Thus, each record which is reproduced exactly has numbers appearing on both sides of it.

DATA statements with an implied DO have the input record listed first, followed by the generated statements. DATA statements flagged for hand conversion have ***HAND CONVERT*** appearing to the right of the output sequence number.

20.1.2 RESTRICTIONS

The value of SIFT is limited by the following restrictions:

1. The input deck must be free of errors.
2. A data statement is limited to 200 comment cards.
3. Variables within an implied DO may appear only in the innermost DO.
4. Each subscript of a variable within an implied DO must be an integer variable controlled by the implied DO, or it must be an integer constant.
5. The DATA statement must not contain redundant parentheses.
6. Array names which appear in the same list as an implied DO are treated as scalars.
7. Groups which contain hollerith or actual data are flagged for hand conversion, without expanding any implied DOs.

20.1.3 REFERENCES

Further information pertaining to SIFT may be obtained by referring to program number G00073, available at the GSFC Computer Program Library.

20.2 FAPCON

FAPCON is used to convert S/360 FORTRAN IV source programs punched in EBCDIC or BCD from single-precision to double-precision format. This program converts all real variables, function calls, constants, and FORMAT statements. The REAL*8 specification is also replaced.

A FAPCON procedure is available on the IBM model 95.

20.2.1 INPUT/OUTPUT

The input is FORTRAN IV source subprograms in card image form. They may be punched in BCD or EBCDIC code.

The output contains a listing of the modified FORTRAN IV subprogram and a modified tape or deck, as requested. Each modified statement is prefixed in the left-hand margin by the number of changes made to it. Any unusual conditions encountered are noted by diagnostic messages.

20.2.2 PROCESSING CAPABILITIES

Most FORTRAN statements are not changed. Those which are changed are:

1. IMPLICIT
2. Arithmetic and logical statements
3. Arithmetic IF
4. Logical IF
5. Call
6. Statement functions

All IMPLICIT statements are deleted. The following statement is added as the first statement of a MAIN program or as the second statement of a subprogram:

```
IMPLICIT REAL*8 (A-H,O-Z,$)
```

The following rules govern the changes which are made to the remainder of the items in the preceding list:

1. All real constants are changed to double precision.
2. All real library function names are changed to their corresponding double-precision function names, except those which are dimensioned, those which appear in an EXTERNAL statement, and those which are used as scalar variables.

3. The portion of an arithmetic IF statement within the outermost parentheses is treated as an arithmetic statement.

4. The portion of a logical IF statement within the outermost parentheses is treated as a logical statement. The portion to the right of the rightmost parenthesis is treated as a separate FORTRAN statement.

The specification statements EXPLICIT, DIMENSION, and COMMON are not changed, but their variable names are examined for possible dimensioning of FORTRAN-supplied function names. It is the programmer's responsibility to assure proper boundary alignment for the DIMENSION, COMMON, and EQUIVALENCE statements.

EXTERNAL subprogram statements are not changed, but the names declared are examined for possible identity to FORTRAN-supplied function names.

20.2.3 RESTRICTIONS

The FAPCON program imposes the following restrictions on the user:

1. A FORTRAN statement may not exceed 20 cards. A warning message is printed for any statement exceeding that limit.

2. No more than 50 comment cards may appear consecutively. If the number exceeds 50, those after the 50th are deleted.

3. Explicit length specification of FORTRAN-supplied REAL library function names are ignored; such names are changed to their double-precision equivalents, unless the statement is dimensioned. For example, REAL*4 SIN is changed to DSIN; REAL*4 SIN(4) is not changed.

4. Each FORTRAN subprogram must end with an END statement, or the succeeding subprogram will be treated as an extension of the previous one.

5. Only one sequential data set (or file) may be processed by FAPCON in any one pass. In order to process a tape containing n files, FAPCON would have to be executed n times.

20.2.4 JCL

A listing of the JCL example and cataloged procedure to execute the FAPCON program follows:

```
MEMBER NAME     FAPCON
//SOURCE  EXEC      PGM=FAPCON,REGION=250K
//COMPILE       DD  DSNAME=&NEW,DISP=(,PASS),UNIT=DISK,SPACE=(CYL,(5,1)),
//                  DCB=(RECFM=F,BLKSIZE=80)
//PUNCH         DD  DUMMY
//SYSPRINT      DD  SYSOUT=A
//SYSIN         DD  DUMMY
```

To execute this procedure for punched input and output, the user must code the following:

```
//stepname      EXEC      FAPCON,PARM=xx
//SOURCE.SYSPUNCH   DD    DSN=&DECK,SYSOUT=B
//SOURCE.SYSIN      DD    *
     (source deck)
```

Where:

xx in the EXEC card must be either DD or EE; DD will cause all E format specifications to be changed to D; EE will leave all E format specifications unchanged.

The source module is also output as a temporary data set, &NEW, which is passed to the next step which is normally a compile step.

The user may also specify input and output on either tape or disk by overriding SYSIN and SYSPUNCH with the appropriate DD parameters.

20.2.5 REFERENCES

Complete documentation may be found in program number G00199, in the GSFC Program Library.

20.3 DEBLOCK/CNVRT PACKAGE

The DEBLOCK/CNVRT package is used to convert 7-track tapes with 36-bit data words to a format compatible with the S/360 32-bit word.

The four DEBLOCK routines read any 7-track tape and expand the 36-bit word by appending two high-order zeros to each group of six-bits, creating an eight-bit byte used by the S/360.

The FORTRAN programmer cannot directly access the record retrieved. The CNVRT routine uses one of 10 possible conversions to convert the data retrieved by the DEBLOCK routines into the S/360 structure.

A call to any of the DEBLOCK routines returns the address of the logical record retrieved; the assembly language programmer then uses this address to process the data.

When working with binary data, the CMPRS routine strips the appended zeros from each byte and reconstructs the binary structure of the 7090/94 word.

20.3.1 DEBLOCK SUBROUTINES -- DBFOR, DBDCS, DBFDCS

The DEBLOCK subroutines DBFOR, DBDCS, and DBFDCS read 7-track tapes having 7094 records with FORTRAN, Direct Couple System (DCS), or FORTRAN-DCS control words, respectively. The logical record is stripped of control words (maximum of four); each six consecutive bits of the 7094 word have two high-order zero bits appended to them, and each is located in continuous bytes in the S/360. The address of the retrieved record is returned in the second parameter of the CALL statement and is available to the CNVRT routine or to an assembly language program. These routines read single files only.

20.3.2 DEBLOCK SUBROUTINE -- DBGEN

The DBGEN subroutine reads any 7-track tape and retrieves a physical record. It is assumed that there are no control words and that the tape is unblocked. As in the other DEBLOCK subroutines, the six-bit groups are expanded to eight bits and stored in the IBM 360. The address of the first byte stored is returned in the first parameter of the CALL statement. DBGEN reads multi-file tapes.

20.3.3 SUBROUTINE CNVRT

This subroutine converts the data returned by one of the DEBLOCK routines into the S/360 structure. The possible data conversions are:

1. Single-precision 7090 floating point to either single-precision or double-precision S/360 floating point.

2. Double-precision 7090 floating point to either single-precision or double-precision S/360 floating point.

3. 36-bit 7090 fixed point to S/360 single-word fixed point, single-precision floating point, or double-precision floating point.

4. 7090 BCD to S/360 EBCDIC.

5. 7090 decrement to S/360 floating point.

6. Move 36 bits without conversion to the leftmost 36 bits of a S/360 double word.

7. Move specified numbers of 7090 words into the data area as a contiguous bit stream.

The required data conversion is requested by a value in one of the parameters of the CALL statement.

20.3.4 SUBROUTINE CMPRS

After a 7090/94 tape has been read in and expanded by one of the DEBLOCK routines, the CMPRS routine may be used to remove the two high-order zero bits and compress the 7090 word to its original structure. The user may specify in the CALL parameters whether he wants six bytes compressed to four and one-half-bytes (36 bits) or 12 bytes compressed to nine bytes (72 bits). If more than one word is to be compressed, each four and one-half-byte group starts on a byte boundary and is separated by a one-half byte from the next four and one-half-byte group.

20.3.5 JCL TO USE DEBLOCK/CNVRT

The programmer must supply a DD card to identify his tape to the DEBLOCK subroutine being used. The DD name depends on the subroutine being used:

Subroutine	ddname
DBGEN	GENTAP
DBFOR	FORTAP
DBDCS	DCSTAP
DBFDCS	DBTAPE

8 9

Each DEBLOCK subroutine requires the DSNAME, UNIT, VOL, LABEL, and DISP parameters in the DD statement, but the DCB requirements differ as shown:

```
DBGEN:   DCB=(LRECL,BLKSIZE,DEN,TRTCH)
DBFOR:   DCB=(LRECL,BLKSIZE,DEN)
DBDCS:   DCB=(DEN)
DBFDCS:   DCB=(DEN)
```

The following is an example of JCL used to identify an input tape to the DBGEN subroutine:

```
//STEP1         EXEC          LINKGO
//GO.GENTAP     DD            DSNAME=name,UNIT=7track,
//                            VOL=SER=xxxxxx,LABEL=(,BLP),DISP=OLD,
//                            DCB=(LRECL=606,BLKSIZE=606,DEN=1,TRTCH=C)
```

Note: The values of LRECL and BLKSIZE are equal to the physical FORTRAN record (including the FORTRAN control word) times six.

This is a 7-track unlabeled tape with a density of 556 bpi. It is read with the data conversion feature on (TRTCH=C).

20.3.6 REFERENCES

Users desiring further documentation for DEBLOCK/CNVRT subroutines should contact Mrs. Pat Barnes (extension 6796) in the GSFC Program Library in Building 3.

20.4 DATCON

The DATCON package is a set of assembler language routines used to write 7090 and 1107 format tapes, complete with FORTRAN and 1107 control words. DATCON is available on the M&DO 360/65, 75, and 95, and may be called from FORTRAN.

20.4.1 CALL STATEMENTS FOR DATCON

There are four forms of the CALL statement to DATCON:

1. The first CALL is issued only once in the program and is issued before any other CALL. It sets up the program to convert for either 7090 or 1107.

2. The second CALL is issued once for each tape generated by DATCON. It must be issued before the third or fourth CALL for that tape. This CALL tells the DATCON routine the FORTRAN logical unit number (between 1 and 30, inclusive) on which the tape is to be written. It also tells the number of 36-bit words to be put in one record on the output tape.

3. The third CALL is issued once for each data conversion. It may be issued one or more times before the fourth CALL is issued. This CALL contains the parameters which define the type of conversion to be executed.

4. The fourth CALL is issued when a record is ready to be physically written on the 7-track output tape. This CALL specifies the FORTRAN logical unit number and whether the data to be written is a record, end-of-file mark, or end-of-volume marker.

20.4.2 JCL FOR DATCON

The output file must be identified in the GO step by a DD statement with the format:

```
//GO.FTxxF001      DD      DSNAME=name,
//                         UNIT=7TRACK,VOL=SER=yyyyyy,
//                         LABEL=(,BLP),DISP=(NEW,KEEP),
//                         DCB=(BLKSIZE=zzz,DEN=1)
```

Where:

xx=FORTRAN logical unit number, as specified in CALL statement.
zzz=A value greater than or equal to the largest output record.
yyyyyy=Volume serial number of the output tape.

20.4.3 REFERENCES

Further information pertaining to DATCON may be obtained by referring to program number GA00042, available in the GSFC Computer Program Library in Building 3.

20.5 OTHER AIDS

The conversion aids described in this subsection are not currently available on any of the M&DO computers. They may be obtained upon request from the GSFC Computer Program Library.

20.5.1 TIDY

TIDY is a FORTRAN program that renumbers and edits other FORTRAN source programs whose statement numbering has become unwieldy and whose readability has deteriorated. This deterioration results from many revisions, patches, and corrections that are typical of reworked programs. TIDY processes programs routine-by-routine and punches new versions of the programs, with the following characteristics:

1. All statements increase in consecutive order.

2. Only statements referred to by other statements retain statement numbers.

3. All statement number references are updated to conform to the numbering scheme.

4. All FORMAT statements are collected and appear at the end of each routine.

5. All FORMAT and CONTINUE statements that are not referenced are deleted.

6. Blanks are inserted in the FORTRAN statements to improve readability; excessive blanks in the statements are deleted.

7. Comments are processed to delete excessive blank comments and to eliminate comments from the FORTRAN statement number and continuation fields.

8. All cards are labeled with a unique letter-number combination. TIDY is entirely written in ASA FORTRAN, and accepts and processes all ASA FORTRAN statements, as well as some IBM and CDC dialect statements.

20.5.1.1 References

For further information, see program 000512 in the GSFC Computer Program Library.

20.5.2 PK ALTR

PK ALTR is a conversion aid for 7090/94 users who will be running FAP or IBMAP coded applications on the S/360 machine. Given a FAP or IBMAP source deck, PK ALTR produces an equivalent program in S/360 assembly language, each source statement producing an average of one or two S/360 instructions. All source statements except I/O, macro definition, and XEC are translated. PK ALTR flags these exceptions, as well as any potential ambiguities or mistranslations, for the user to correct manually. PK ALTR is written in FAP and runs under the FMS monitor, requiring a 32K 7090 with eight tape drives.

20.5.2.1 References

Users desiring further documentation for PK ALTR subroutines should contact Mrs. Pat Barnes (extension 6796) in the GSFC Program Library in Building 3.

20.5.3 FORTLCP

FORTLCP is used to convert FORTRAN IV statements into PL/I statements having the same meaning and effect.

20.5.3.1 Capabilities

FORTLCP does the following:

1. Detects and flags those FORTRAN IV statments that have no PL/I equivalents, or which may result in ambiguous translations.

2. Produces an output listing of the PL/I program, and optionally, the original FORTRAN statements.

3. Produces, when specified by the user, the converted program on cards, tape, or disk.

4. Makes optional editing features available.

20.5.3.2 Restrictions

FORTLCP has the following restrictions:

1. Hexadecimal and octal constants are not converted.

2. Assigned variables cannot be used for any purpose other than for the assigned GO TO statement.

3. Transfers back into a DO loop are not allowed.

4. An implied DO in a DATA statement is not converted.

5. Initial values assigned to variables in EQUIVALENCE statements should be adjusted by the user.

6. Cannot simulate the effect of some FORTRAN subprograms.

7. Use of the following FORTRAN IMPLICIT statement causes a 322 ABEND:

```
IMPLICIT REAL*8 (A-H,O-Z)
```

20.5.3.3 References

Users desiring further documentation for FORTLCP subroutines should contact Mrs. Pat Barnes (extension 6796) in the GSFC Program Library in Building 3.

20.6 DACUT9

DACUT9 is a set of System/360 Assembler Language subroutines which write 7094 FORTRAN unformatted (binary) tapes, complete with FORTRAN control words. The subroutines are FORTRAN callable. The DACUT9 subroutine package uses about 4k bytes of memory plus work areas the size of the records being written. At least one 7-track tape unit with the convert feature is needed.

20.6.1 INDIVIDUAL SUBROUTINES

A discussion of the individual DACUT9 subroutines follows:

GOBUILD: This routine sets up registers for the BUILD routine.

BITSTR: This routine takes care of all bit string conversions. It handles all the addressing of input and output areas.

CHAR: This routine translates character strings to 7090 code by means of a translate table. The translated characters are then moved to the output area.

INTEGER 2
INTEGER 4
INTEGER: These routines handle all possibilities of integer conversion: S/360 short or long form for the 7090. The resulting numbers are in the proper format, with complementing where necessary and the proper sign affixed.

BUILD: This routine obtains the output area for the writing of records, and builds the DCB's required. The entry (four System/360 words) in the table of logical units being written on is filled in with the initial values. The address of the table is returned to the calling routine. The DCB is built and opened. The output area provided by this routine is about 30% larger than asked for to allow for the FORTRAN control words.

WRITE: This routine handles all the "housekeeping" involved in writing of records on tape. It affixes the FORTRAN control words where required (these control words are not used with bit strings), branches to the OUTAPE routine to actually write the tape, and initializes all values in the table entry after completing the write.

OUTAPE: This routine, called from WRITE, handles the actual process of writing of record on 7-track tape. It places the proper information in the Data Event Control Block (DECB), writes and checks the record.

SEARCH: This routine, given a tape number, returns the address of the table element corresponding to that tape. The routine is used by almost all the other routines.

ABEND: This routine places in the Abend macro the code corresponding to the type of error mode. This code appears on the core dump as USER = xxxx, where xxxx is the numeric code for the error.

F36F90: This routine handles conversion of floating point numbers to 7090 format. Either single- or double-precision numbers may be input to the routine, and the result can be 7090 short or long form.

FLOAT: This routine sets up linkage and handles housekeeping for the floating-point routine (F36F90).

20.6.2 REFERENCES

The document DACUT9 is available in the GSFC Program Library (extension 6796) in Building 3, Room 133. This document contains a discussion of the DACUT9 CALL statements and parameters, error codes, a sample program, and examples of the DD cards required to use DACUT9.

SECTION 21

DEBUGGING FACILITIES

21.1 INTERPRETING SYSTEM MESSAGES

The general format of a system message is as follows:

AAAnnnB

where:

AAA is a three-letter prefix that identifies the module or processor (or set of processors) giving the message

nnn identifies the specific message; the first digit of nnn often identifies the program having the error

B is an indicator that describes the type of message being given:

I = information only

A = action required (i.e., an operator message)

W = wait, processing stopped until action taken

D = decision required by operator

E = eventual action required by operator

The modules and processors that correspond to the three-letter prefix, plus the first-digit specifications (where applicable) are shown in Table 21.1-1.

The full text of the messages corresponding to the codes are contained in the IBM manual, Messages and Codes, GC28-6631. The text may list several possible reasons for the ABEND, or, its meaning may seem unclear. If such difficulties are encountered, contact the PAC, extension 6768, Building 3, Room 133A.

ABENDs are usually accompanied by system completion codes, discussed in Section 11; they are also listed in Messages and Codes.

Table 21.1-1. System Message Prefixes

IEA		Supervisor Messages
IBC		Independent Utility Messages
IEB		Data Set Utility Messages
	IEB0nn	IEBEDIT
	IEB1nn	IEBCOPY
	IEB2nn	IEBCOMPR
	IEB3nn	IEBGENER
	IEB4nn	IEBPTPCH
	IEB5nn	IEBUPDAT
	IEB6nn	IEBISAM
	IEB7nn	IEBDG
	IEB8nn	IEBUPDTE
IEC		Data Management Messages
	IEC0nn	End of Volume
	IEC1nn	Open
	IEC2nn	Close
	IEC3nn	Catalog Management
	IEC4nn	Checkpoint/Restart
	IEC6nn	Direct-Access Device Space Management
	IEC7nn	Tape Label Creation
	IEC8nn	BTAM/QTAM

Table 21.1-1. (Cont'd)

IEE		Master Scheduler Messages
IEF		Job Scheduler Messages
IEG		TESTRAN Messages
IEH		System Utility Messages
	IEH1nn	IEHLIST
	IEH2nn	IEHPROGM
	IEH3nn, IEH4nn	IEHMOVE
	IEH5nn	IEHUCSLD
	IEH6nn	IEHINITT
	IEH7nn	IEHIOSUP
	IEH8nn	IEHDASDR
IEI		System Generation Messages
IEJ		FORTRAN IV E Messages
IEK		FORTRAN IV F Messages
IEM		PL/1 F Messages
IEP		COBOL E Messages
IEQ		COBOL F Messages
IER		SORT/MERGE Messages
IES		RPG Messages
IET		ALC E Messages
IEU		ALC F Messages
IEW		Loader/Link Edit

Table 21.1-1. (Cont'd)

IEX	ALGOL
IEY	FORTRAN IV G Messages
IHB	Supervisor and Data Management Assembler Expansion Messages
IHC	FORTRAN IV Object Program Messages
IHD	COBOL E Object Program Messages
IHE	PL/I Object Program Messages
IHI	ALGOL Object Program Messages
IHJ	Check/Restart Messages
IHK	RJE Messages
IMx	Service Aids Messages

21.2 IMPRECISE INTERRUPTS ON THE 360/95 AND WHAT TO DO NEXT

Imprecise interrupts occur on the model 95 because several instructions can be in execution at once. When an additional instruction enters execution (even though previous ones may not have finished), the instruction counter is updated. This means that when one of the instructions causes a program exception interrupt, the system cannot tell which of the several instructions in execution caused the exception.

The cause of the interrupt is found in the old PSW double word (location hex 28 of a dump), in digits five through seven, when bits 27-31 and 32-33 = 0.

PSW Digits 5-7	Related Completion Code	Interrupt Type
800	0C4	Protection - possible cause, trying to store into a location outside of the program
400	0C5	Addressing - addressing a location outside of the jobs region
200	0C6	Specification - possible cause, boundary alignment
100	0C7	Data - applies to decimal operations, therefore inapplicable on the 95
080	0C8	Fixed-point overflow
040	0C9	Fixed-point divide check
020	0CC	Floating-point exponent overflow
010	0CD	Floating-point exponent underflow
008	0CE	Significance
004	0CF	Floating-point divide check

A combination of the above codes may occur, e.g., 030 would mean that both underflow and overflow occurred. Sometimes, digits five through seven contain information which is meaningless.

The preceding table applies only when digit seven is even, digit eight is zero, and digit nine is less than four.

Suppose that an old PSW is:

F	F	5	5	0	0	4	0	2	2	1	3	0	8	E	6
				5th	6th			7th	8th			9th			

The above table can be used since digit seven = 4 (even), digit eight is zero, and digit nine is two (less than four). The related condition code is OCF, a divide check. If there is no STAE traceback available to find the FORTRAN statement causing the divide check, or an assembler listing of the FORTRAN program, some of the following reasons for OC1-OCF errors should be considered:

a. The arguments in CALL and subroutine statements do not agree in number and type.

b. A subscript was used which was out of range of the defining dimension statement.

c. Variables have not been initialized.

d. Transfer was made to the middle of the DO loop.

e. CALL statements of form CALL SUB (A,1,2) where SUB changes arg 2 and arg 3.

f. There is an incorrect or missing DD card.

g. An unformatted READ or WRITE exists, with units referenced by formatted I/O statements (unformatted I/O must use RECFM=VS or VBS on DD card).

h. There are missing or misnamed JCL statements.

These reasons are not matched with specific interrupt types because each of the errors can precipitate various interrupts, depending on the exact situation.

If none of these reasons seems to apply, the PAC Center should be consulted, Building 3, Room 133A, extension 6768.

21.3 ERROR TRACEBACK

When the FORTRAN Extended Error Handling facilities (see paragraph 21.5.2) are not used, the built-in error handling facility of the FORTRAN compiler is in effect. When an error occurs, the compiler takes the standard fixup, described in Appendix D of the FORTRAN G&H Programmer's Guide, (GC28-6817), "Execution Error Messages."

In addition to the standard fixup, the error handler also provides a traceback to enable the programmer to find the FORTRAN statement associated with the error. The chain of subroutines is shown in the traceback, in reverse order. Next to the subroutine name there is a number that refers to the ISN of the statement in the calling routine. In Table 21.3-1, ISN20 in MAIN calls COMP, and ISN17 in COMP calls FNDTSK, which calls IBCOM. Registers 14, 15, 0, and 1 are displayed to the right of each routine. Register 14 gives the address to which the routine would have returned; register 15 shows the entry point of the routine. R1 contains a pointer to a list of addresses indicating the variables passed to the called program. The entry point of the main program is shown in hex below the summary. Next is a message describing the action taken and the results. The error code is on the last line; looking up this code in the FORTRAN G&H Programmer's Guide will show what the standard fixup is for this error. For I/O errors, the record having the error is printed below the text (not shown). If the ERR option was specified in the I/O statement, control is returned there, after the fixup.

Table 21.3-1. Example of Error Traceback

TRACEBACK ROUTINE	CALL FROM ISN	REG. 14	REG. 15	REG. 0	REG. 1
IBCOM		00033DAC	00037260	00000024	00033BDC
FNDTSK	0017	620346C6	00033A78	00045990	00034138
COMP	0020	42030B76	00034038	00000001	000309C8
MAIN		4005AABE	00030808	FF000018	0005A7F0

```
ENTRY POINT = 00030808

STANDARD FIXUP TAKEN , EXECUTION CONTINUING

IHC215I CONVERT - ILLEGAL DECIMAL CHARACTER *
```

21.4 DUMPS

An ABEND error is caused by an SVC issued by the processing program, problem program, or supervisor if it is unable to continue processing. Both ABEND and UDUMPs are caused by the ABEND SVC (appropriately numbered 13).

An ABEND is most often issued by the supervisor in response to a program interrupt. The interrupt handler in the NUCLEUS fields the interrupt, determines the cause, and decides if the error is recoverable. If it is not, an ABEND is issued for the task causing the interrupt.

The fact that the ABEND is issued by the interrupt handler means that the displayed PSW and REGS AT ENTRY TO ABEND do not refer to the user or processing program, but rather to the interrupt handler (where the user issues the ABEND, reference is to the user program).

The first four dumps listed below are in most MVT OS systems; the fifth is local to the M&DO 360/95, 360/75 (ORBIT), and 360/65.

1. SYSUDUMP (or just UDUMP) - a full core dump, listing all control blocks for the user's region (such as TCBs, RBs, DEBs, etc.), plus the user's region. A SYSUDUMP is only given when a //SYSUDUMP DD card is included in the JCL for the job step.

2. SYSABEND dump - same as SYSUDUMP, but in addition, it has dumps of the NUCLEUS and a TRACE TABLE of interrupts. Normally, the SYSUDUMP should be sufficient. A //SYSABEND DD card will provide a SYSABEND dump.

3. SNAP dump - produced by assembler language SNAP macro; job continues processing. The FORTRAN PDUMP is equivalent to the SNAP dump. See the Supervisor and Data Management Macro Instructions manual, GC28-6647, for details.

4. Stand-alone dump - used for system crashes to dump all of core. The dump is formatted by service aid utilities. No formatting of control blocks.

5. GSFCDUMP - obtained via CALL STAE, it prints an abbreviated formatted dump on the data set indicated by the //GSFCDUMP DD card. For details, see T&DS Computer Bulletin No. 9, or Frank Ross, extension 6796.

21.5 FORTRAN DEBUGGING AND ERROR HANDLING

21.5.1 FORTRAN DEBUGGING PACKAGE

The FORTRAN Debugging Package, available with FORTRANG, is a programming aid to facilitate error determination in FORTRAN source programs. The debug package provides the capability of tracing program flow, displaying values of variables, and checking subscripts for proper range. The debug control statements are TRACE ON, TRACE OFF, and DISPLAY variable list. These statements, along with several examples, are described in the FORTRAN Language publication. The debug statements are fairly easy to use and may be inserted anywhere within the program. A statement of the form AT statementnumber is used to indicate the place within the program where debugging is desired. The major limitation of the debug facility is that the debugging information cannot be conditionally controlled.

21.5.2 FORTRAN EXTENDED ERROR HANDLING

Certain errors in FORTRAN programs are monitored by the FORTRAN Error Monitor at execution time. For these errors, the FORTRAN Error Monitor prints a message indicating the type of error which occurred, traceback information containing the last sequence of subroutine calls, and contents of certain registers; in addition, for each type of error, the Error Monitor takes some specific action, e.g., terminating the job step, setting a value to zero, or continuing execution. For each particular error that is detected, the normal action taken by the Error Monitor is referred to as "standard corrective action." The purpose of the extended error handling facility is either to permit the user to specify certain user-desired actions to be taken upon error occurrence, in place of the standard corrective action, or to change the number of allowable errors of a given type.

A detailed explanation of the extended error facility is given in the FORTRAN G&H Programmer's Guide. The following is an example of one possible use of error handling:

> Suppose while executing a FORTRAN program, IHC210 errors are produced, indicating that a floating-point underflow has occurred. This is not a fatal error; normally, an unlimited number of occurrences is permitted for this type of error. If the programmer is unable to identify the variable producing this error, he may desire a core dump to check the values of certain variables at the time the error occurred. The programmer could cause this error to be fatal by inserting the following two statements in the program in which the error occurred:
>
> EXTERNAL MYDUMP
> CALL ERRSET (210,2,1,2,MYDUMP), where MYDUMP is a written subroutine, such as:

```
SUBROUTINE MYDUMP
CALL ABEND (99)
RETURN
END
```

The above insertion causes a transfer to the user subroutine MYDUMP after one occurrence of the HC210 error. Error traceback information will also be printed. The number of error occurrences permitted is user-specified by the use of the second parameter in the ERRSET call. Also, the fourth parameter can control the printing of the traceback.

There are many uses of the extended error facility. It is suggested that the user refer to the FORTRAN G&H Programmer's Guide for a more thorough explanation of the use of this facility.

21.6 TESTRAN

TESTRAN is an assembly language set of macro instructions that provide execution time debugging facilities to problem programs. Services are performed at specified points in the problem programs, as ordered by the TESTRAN statements. TESTRAN macro instructions and problem program instructions can be intermixed, grouped separately, or independently assembled (to be merged by the Linkage Editor).

TESTRAN statements provide the capability of dumping and tracing selected areas of the program, either unconditionally or on selected criteria. By using the various TESTRAN options, it is possible to use TESTRAN to debug overlay programs or dynamically serial or dynamically parallel programs. TESTRAN modules are neither re-enterable nor re-usable, so care should be taken that a module has been closed before it is re-opened.

The IBM manual, TESTRAN, GC28-6648, gives examples of the use of TESTRAN statements, TESTRAN syntax, and TESTRAN data set JCL.

21.7 PRINTING DATA SETS FOR DEBUGGING

21.7.1 CORE DUMPS

Several types of post-mortem dumps are available, as shown in this table:

Table 21.7-1. Post-mortem Dumps

DDNAME	CALL	REFERENCES	PUBLICATION
SYSABEND	ABEND	11.4 and 21.4	GC28-6670
SYSUDUMP	ABEND	11.4 and 21.4	GC28-6670
GSFCDUMP	STAE	21.4	T&DS Bulletin, #9
PL1DUMP	IEHDUMP		GC28-6594

These, and some others, are obtained by defining them as SYSOUT data sets. Alternatively, they can be written to intermediate devices and processed later by a utility or editing routine. The TESTRAN output (SYSTEST) is processed by an editing routine. Dumps are usually output to intermediate devices on systems which are print-bound; in this way, the printer is not tied up with a 200-300 page dump, unless it is absolutely necessary. At the same time, the dump is available, if needed, without having to rerun the job.

The following DD statement is coded in the job step:

```
//SYSUDUMP    DD   DSN=pgiddump,DISP=(,,KEEP),
//                 DCB=(RECFM=VBA,LRECL=137,BLKSIZE=7265),
//                 UNIT=DISK,SPACE=(TRK,(50,100))
```

To print the dump, it is necessary to execute the following job:

```
//JOBNAME     JOB      required jobcard information
//PRINT       EXEC     PGM=IEBGENER
//SYSUT1      DD       DSN=pgiddump,DISP=(OLD,DELETE,KEEP),
//                     UNIT=DISK,VOL=SER=volid
//SYSPRINT    DD       DUMMY
//SYSIN       DD       DUMMY
//SYSUT2      DD       SYSOUT=A
```

The volume identifier (volid), as shown in the example above, would be determined from the de-allocation messages of the first run.

21.7.2 DYNAMIC DEBUG OUTPUT

While writing a program, a programmer will often include debug aids. These may include SNAPSHOTS (ALC), PDUMPS (FORTRAN), CHECK lists (PL/I), or hand coding, such as writing out input or intermediate variables. The programmer may desire routing these debug outputs to data sets other than the normal outputs. Many FORTRAN programmers write all output to file six (FT06F001).

Several outputs will appear intermixed and more than likely cause confusion; however, by putting all output on one data set, the programmer gains the advantage of seeing the output in chronological order. Special precautions must be taken when multi-tasking (in ALC or PL/I). If two tasks both use the same data set, one task will overwrite the data of the other. Only one DCB may be opened at any time for a data set, or the results will be unpredictable.

21.7.3 INTERMEDIATE OUTPUTS

In order to debug a program, it is often desirable to look at some input or output data. The IEBPTPCH and IEBGENER utilities are useful. They are described in Section 9 and in the IBM manual, OS Utilities, GC28-6586. Among the options available are field editing (spacing), character format conversion (Hex), selective printing (every Nth record), and limit printing (stop after n records).

For JCL, system, and procedural problems, a trip or call to the PAC is advisable.

21.8 B37s, D37s, E37s

The system completion codes -- B37, D37, and E37 -- refer to situations involving (1) lack of space on a volume, (2) insufficient primary track allocation with no secondary allocation, or (3) inadequate number of volumes, or (4) exhaustion of all 16 extents. See the IBM manual, Messages and Codes, GC28-6631, for a detailed description of these errors.

The programmer's primary consideration when encountering one of these errors is to find out which data set has a space problem. To find the data set that ran out of space, the next to last SVRB in the formatted part of the dump (first or second page) must be found. Register 2 in this section contains the DCB address. This address (rightmost six hex digits) should be matched with a DCB address in the tables following the formatted TIOT. This address appears to the right of the DCB under each DDNAME in the tables. The DDNAME that has the same DCB address as that in Register 2 indicates the data set which ran out of space.

Once the offending data set has been identified, a check must be made to ascertain that a program error, such as a loop in an output routing, did not cause the overflow. If a program error has not caused the overflow, then a legitimate space problem exists. If the data set is not a SYSOUT data set, then the user(s) must have allocated it. It is therefore necessary to calculate again the amount of space required.

The system provides a default allocation (TRK,(20,100)) for SYSOUT data sets. If this is insufficient, more space may be allocated by including the SPACE parameter on the DD card defining the SYSOUT data set. A reverse application of the SPACE parameter is to use a D37 to limit the amount of output on a data set. Each track allocated provides space for about one page of output at full track blocking. (No secondary should be allocated when using this technique.) This technique leads to an interesting case when a job step ABENDS with two completion codes: the first is legitimate; the second is an X37 which occurs during ABEND processing.

If sufficient space has been allocated, several problems could have caused the overflow. For a D37, no secondary was allocated; therefore, a larger primary or secondary must be allocated. For a B37 or an E37, either all 16 extents were allocated, or the VTOC was full. Although it is unlikely for a full VTOC to be encountered on a scratch or system pack, due to free-space-fragmentation, 16 extents could have been used before all the requested space was allocated. (A diskmap or LISTVTOC of the volume involved would show the condition of its VTOC.) The user must either request that the job be run after scratch packs are scratched, or allocate more units in the UNIT parameter. (Partitioned data sets, however, cannot be on more than one unit.)

The "STAE" subroutine will give the ddname, unit address, volume, and formatted DEB of the data set encountering the B, D, or E37.

The scratch packs tend to fill up when the system is unstable. When the system is IPLed on a "warm start" basis, all jobs which were executing when the system went down are terminated. As part of the termination processing, their temporary data sets are de-allocated, i.e., deleted. Messages indicating which data sets have been deleted appear on the operator's console. However, if a "cold start" must be performed, the de-allocation does not take place. The data sets are scratched as part of the "cold start" procedure.

An excellent manual for debugging dumps is An Introduction To OS/360 MVT Control Logic and Debugging With MVT Core Dumps, IBM manual number Z77-9058, obtainable from Mrs. Pat Barnes, extension 6796.

21.9 TADPOLE

The Test and Debug Procedure for On-Line Execution (TADPOLE) is an extension of the FORTRAN Debugging Package. It allows a FORTRAN programmer to use a 2250 Display in debugging his program. In addition to the standard features of the FORTRAN debugging package (described above), TADPOLE provides capabilities to display and change variables, cause traps on program conditions or other criteria, or delete code. Information about using and coding programs for TADPOLE is contained in the TADPOLE User's Guide (ask Pat Barnes, extension 6796, or Randy Barth, Code 603.3).

21.10 SIGPAC

SigPac, a prototype software system which is now operational on the model 95, traces the numerical significance of calculations. FORTRANG source language programs are tested with actual user data, with user-specified accuracy assumptions for input data and program constants. The system provides selective segment significance execution; single or double precision, integer, or mixed arithmetic and local and remote batch operation. Significance tracing has been successful with up to 1400 statements in a program.

This initial version of the system produces statistical predictions of "most probable remaining significance" at any chosen point during execution of a user program. SigPac is available for test use within GSFC. Copies of the User Manual can be obtained from the GSFC Program Library, Building 3, Room 133.

SECTION 22

OVERLAY CONSIDERATIONS

22.1 INTRODUCTION

OS provides the programmer with the capability of overlaying one part of his program with another, thus reducing the amount of storage necessary to run the program. The structure of an overlay program is specified at the Link Edit step. At execution time, each overlay is loaded, using the Overlay Supervisor, as required by the program.

Frequently run programs, using over 400K of core storage, benefit if they are formatted in overlays. One factor in determining job classes is the region size; therefore, any reduction in region size (due to overlaying) may put the job in a class to which more initiators are started. This should provide a noticeably faster turnaround time, at little cost to the total CPU and I/O time for the job.

The following subsections present programming considerations when overlaying a program, the definitions of the elements of an overlay program, and the Linkage Editor control cards required to achieve an overlay structure. For assistance in overlaying currently running programs, or in designing a program for overlay, the PAC should be consulted in Room 133A, Building 3, extension 6768.

For further reference, see Linkage Editor and Loader, IBM (GC28-6538).

22.2 DEFINITIONS

SEGMENT: The smallest portion of the program that can be loaded as one logical unit at execution time.

NORMAL OR NON-OVERLAY PROGRAM: One segment containing a main routine (may contain many subroutines and/or common blocks).

TREE: The graphic representation of an overlay program, showing which portions (segments) are in core at any time.

ROOT SEGMENT: It remains in memory throughout the execution of the program, and should be used for control of the program functions. The root segment is part of every path. (See Figure 22.2-1 for an example of a tree.)

PATH: All segments in line between the segment currently executing and the root segment. In Figure 22.2-1, path is pictured by drawing a line upward from the segment being considered (REPLACE for example) through every higher segment (SMOOTH), until the root segment (CONTROL) is reached. When a segment is in memory, all segments in its path (above it) are in memory also. Thus, in the example, a call from CONTROL to REPLACE would automatically load SMOOTH into memory.

OVERLAY REGION: The contiguous area of main storage which contains one "tree" of a program. An overlay program may contain up to four regions, and thus four trees may be built. The term "region," used in overlay discussions, should not be confused with the same term used to specify the main storage required for a job step. Each of the four (maximum) regions of an overlay program run in the one storage region specified for the job step.

MULTIPLE REGIONS: More than one overlay region. Used when the program becomes so complex that it is desirable to be loading one path while executing another path. Multiple regions are also used when the root segment is large enough to benefit from being overlaid. When multiple regions are used, each region has its own tree, and is independent of other regions, except that segments have access to segments that are not in their path. Thus, a program tree that has identical sub-trees on more than one branch is a good candidate for multiple regions. Figure 22.2-2 illustrates a program structure that will benefit from multiple regions; Figure 22.2-3 shows the structure divided into two regions. Note in Figure 22.2-3, the routines READ and ERROR appear in the first tree more than once, and are put into the second region in the second tree.

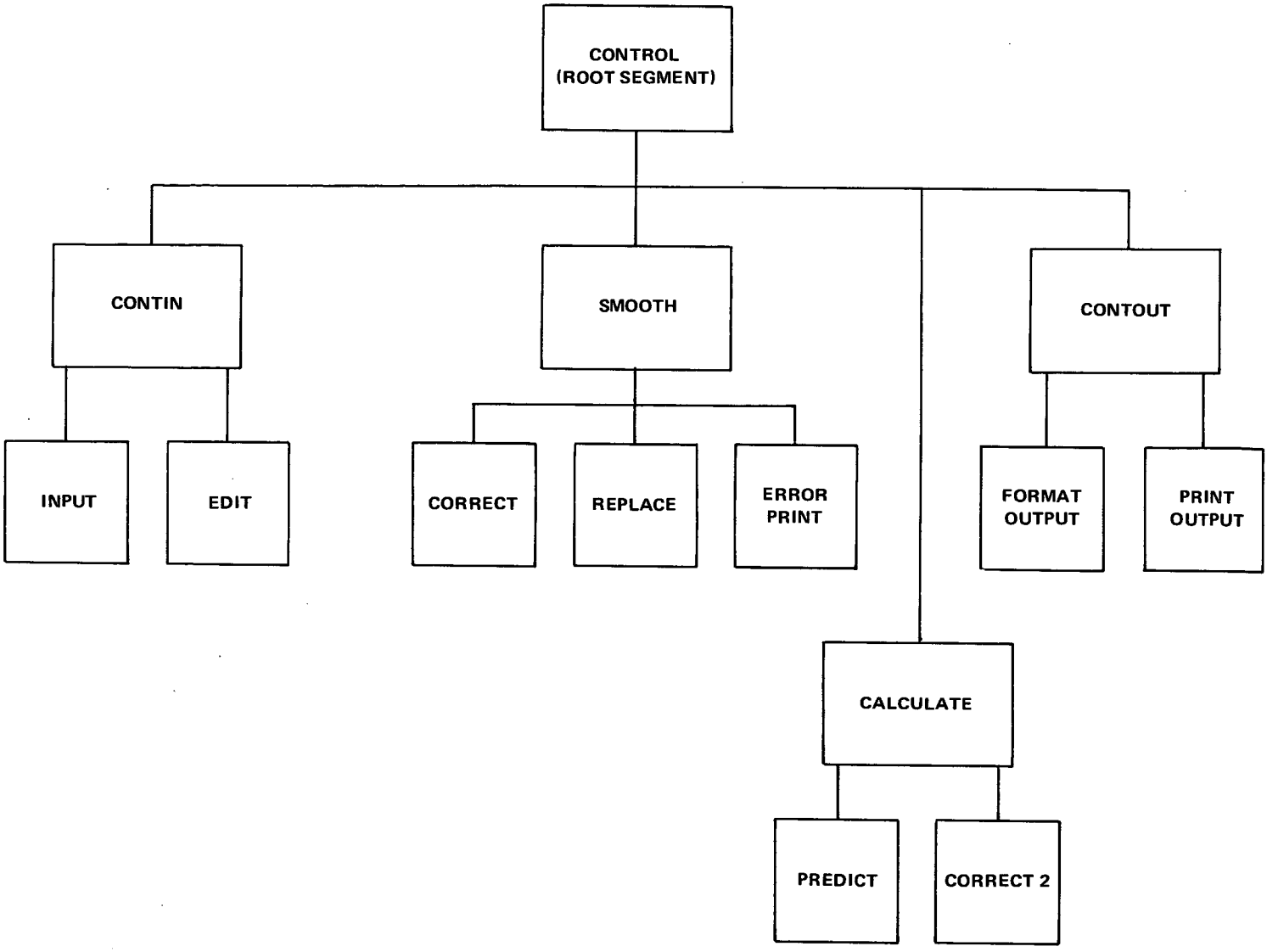


Figure 22.2-1. Example of Tree Structure

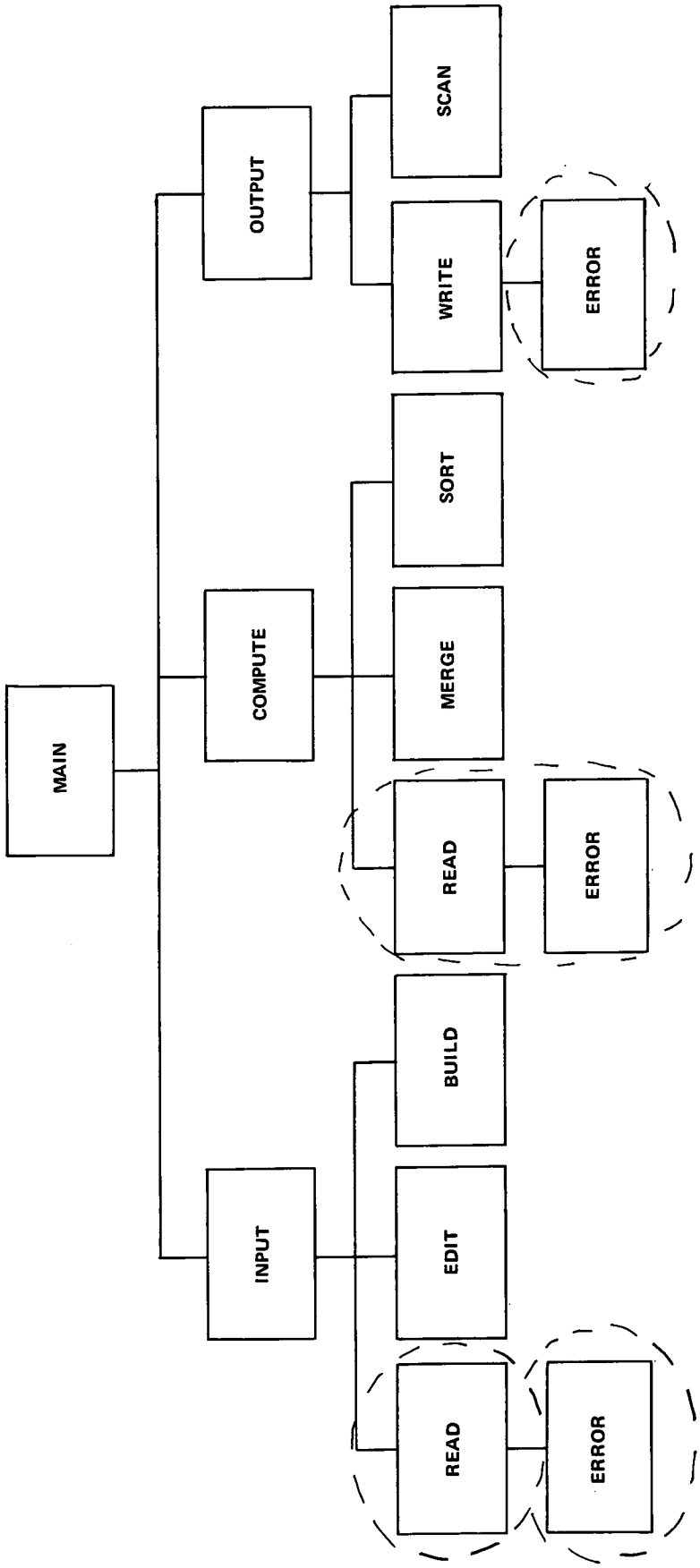


Figure 22.2-2. Tree Diagram

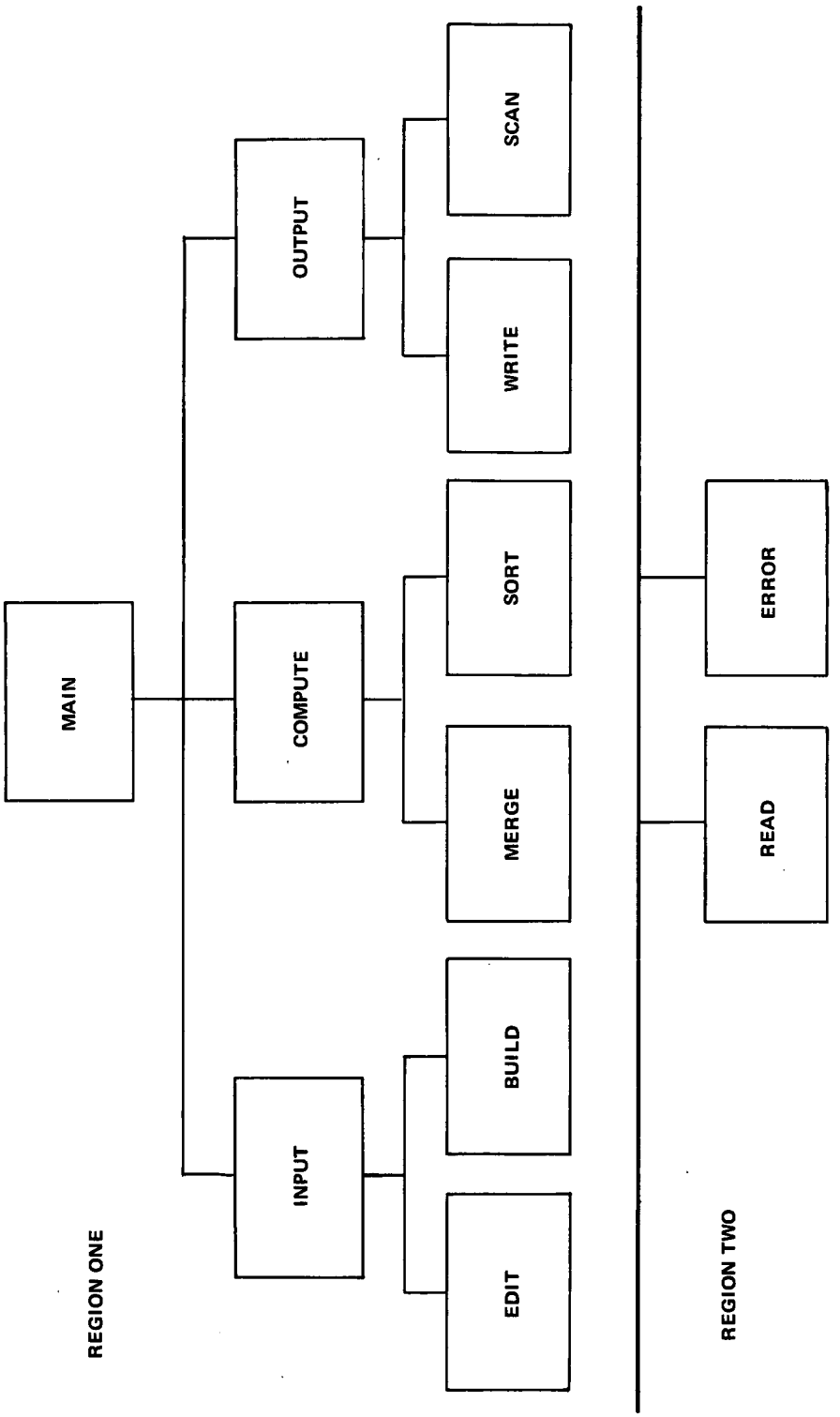


Figure 22.2-3. Tree Diagram

INCLUSIVE SEGMENTS: They can be in main storage at the same time. That is, the segments containing the routines CORRECT2 and CALCULATE are inclusive, since a path can be drawn to include them both.

EXCLUSIVE SEGMENTS: They are in the same region, but not in the same path. The segments containing EDIT and INPUT are exclusive, since one will always overlay the other in memory.

COMMON SEGMENT: This occurs where two separate paths join. The segment containing CONTIN is the common segment to the segments containing INPUT and EDIT.

22.3 PROGRAMMING CONSIDERATIONS

22.3.1 GENERAL

A good overlay structure corresponds to the logical flow of the program. An overlay tree that corresponds to the program must be constructed before a program call can be overlaid. Each major level should perform one logical function, and should not need to be frequently recalled after it has been overlayed. A call to an overlay segment should not be placed inside a tight loop.

Exclusive segment references are legal only if there is a reference to the exclusive segment in a common segment. In Figure 22.2-1, references between INPUT and CONTIN are inclusive; references between INPUT and EDIT are exclusive (but legal); and references between INPUT and PREDICT are exclusive and illegal (unless specific references are included in CONTROL).

22.3.2 COMMON ROUTINES AND DATA

Only one copy of a routine should appear in the tree. If it is necessary logically for a routine (or group of routines) to appear more than once, then either a separate overlay region should be used for those routines, or the routines should be moved up in the tree until they are high enough to be common to all branches in which they are to appear. If it is desirable for a routine to appear in several segments, the routine must be named differently each time it appears.

System service routines normally are included in the root segment. If their use is limited to some branches, they may be placed in lower segments -- especially special packages like mathematical or plotting routines. Any routine that issues an OPEN cannot be overlaid until a CLOSE is issued; in FORTRAN, this is handled by leaving IBCOM in the root segment.

Only the original copy of a segment (present at start of execution) is kept; each time an overlay segment is called, the routine reverts to the original code. This means that variable data to be passed between routines cannot be kept in the code that is overlaid. Instead, data to be passed should be kept in data sets, common blocks, or block data subprograms. Common blocks should be positioned so that they are located in the common segment of the segments that share the data. Figure 22.3-1 shows a portion of a tree with common blocks COM1 and COM2.

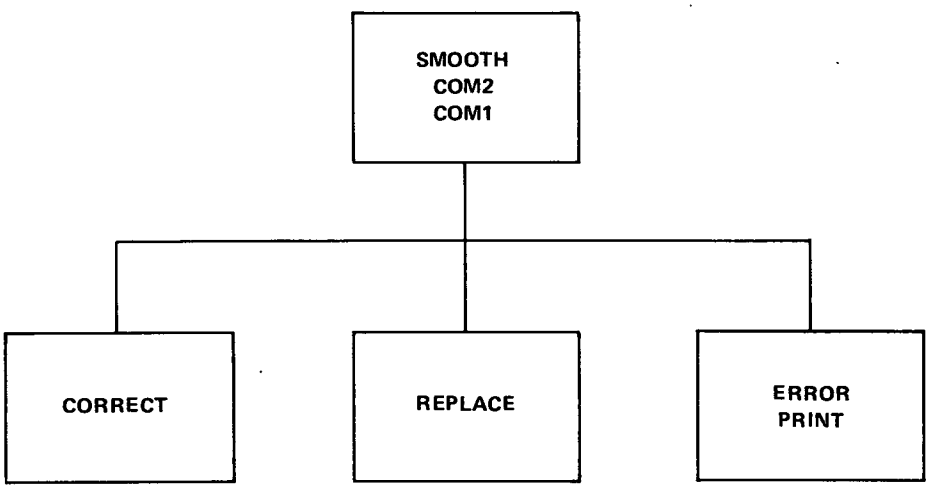


Figure 22.3-1. Example of a Tree Portion

The common block COM1 is used to communicate error data between CORRECT, REPLACE, and ERROR PRINT. This block must be in SMOOTH, since SMOOTH is the common segment (it is in all three paths) to the three routines. The common block COM2 is used only by CORRECT. It was not placed in the CORRECT (no pun) segment since CORRECT may be called in repeatedly, and the flags and counters must be preserved for each call.

Variable data initialized via data statements or block data subprograms should be placed in the next higher common segment (like COM1 above). (Data statements should be made into block data subprograms before being moved.)

Constant data can remain in the lowest segment needed, since the values do not change during execution.

It is important to recall the general rule that the length of each data statement in a block data subprogram must correspond to the length of the common block it is initializing. Note also that common blocks are more efficient than calling sequence arguments, because they occupy less storage and are passed more quickly than arguments.

22.3.3 OVERLAY TREES

Keeping in mind that the overlay structure should correspond to the logical structure of the program, it should be designed with the minimum practical number of repeat calls to a segment. For example, in Figure 22.3-2, after the INPUT segment has been called in, it should stay in until a set of input data has been processed.

When determining the tree for a program, each subprogram and labeled common set must have a unique name. A subprogram can only appear once in the tree. A labeled common set only appears in the tree in the highest segment that references it.

Before the Linkage Editor control cards are made up, a copy should be made of the tree listing all routines and common blocks in each segment.

Looking at the tree in Figure 22.3-2, the overlay levels are indicated by the horizontal line over the exclusive segments. These levels (two, in this diagram) should be given names to be used by the Link Editor.

The size of an overlay level can be determined roughly from the length of the longest segment at that level, since the load point of each segment at a given level is the same. In addition to a program's object code, there are tables within each segment that list other segments that can be called by an individual segment.

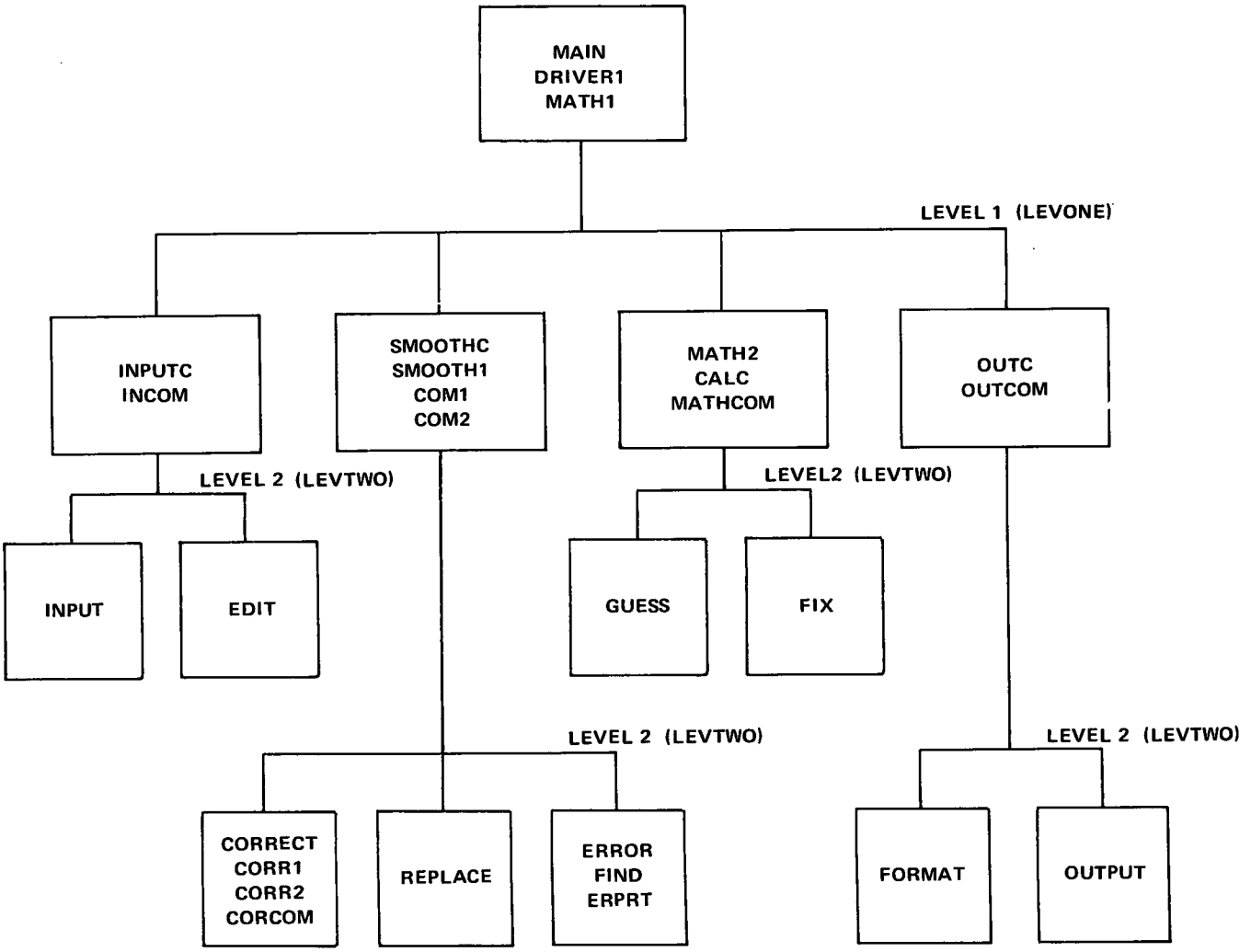


Figure 22.3-2. Tree Diagram

22.4 LINKAGE EDITOR CONTROL CARDS

Input cards to the Linkage Editor are placed after the //SYSLIN DD * card. INCLUDE or ENTRY cards should remain in their usual position in front of the overlay cards. The overlay cards are OVERLAY name (region) and INSERT Sub1,Sub2,....,Subn.

To determine the order of the OVERLAY and INSERT cards, the user refers to his tree, as directed in Table 22.3-2. Each level (1 and 2) is referenced on an OVERLAY card; the routines that make up a segment in the level follow on the INSERT card. There must be an INSERT card for each OVERLAY card. The root segment need not be considered an overlay level, since all routines present, but not named on INSERT cards, automatically go into the root.

Utilizing the tree diagram (Figure 22.3-2), the user must proceed to describe his tree structure by going from top to bottom on one branch (leftmost) and from left to right, finishing all of one branch before going to the next. The OVERLAY and INSERT cards for the example in Table 22.3-2 are given below:

```
OVERLAY   LEVONE
  INSERT    INPUTC,INCOM
OVERLAY   LEVTWO
  INSERT    INPUT
OVERLAY   LEVTWO
  INSERT    EDIT
OVERLAY   LEVONE
  INSERT    SMOOTHC,SMOOTH1,COM1,COM2
OVERLAY   LEVTWO
  INSERT    CORRECT,CORR1,CORR2,CORCOM
OVERLAY   LEVTWO
  INSERT    REPLACE
OVERLAY   LEVTWO
  INSERT    ERROR,FIND,ERPRT
OVERLAY   LEVONE
  INSERT    MATH2,CALC,MATHCOM
OVERLAY   LEVTWO
  INSERT    GUESS
OVERLAY   LEVTWO
  INSERT    FIX
OVERLAY   LEVONE
  INSERT    OUTC,OUTCOM
OVERLAY   LEVTWO
  INSERT    FORMAT
OVERLAY   LEVTWO
  INSERT    OUTPUT
```

All Link Editor input cards must have column 1 blank.

During program development, the parameters

```
PARM=(XREF,OVLY,XCAL)
```

are recommended since the cross reference table can be useful in optimizing overlay programs. In general, only PARM=OVLY is required: e.g.,

```
//BUILD   EXEC      LINKGO,PARM=OVLY
                        or
//BUILD   EXEC      LINKGO,PARM=(OVLY,XREF,XCAL).
```

The cross reference table can be used to determine the longest path; once the longest path is known, efforts can be made to reduce it, either by moving some routines into the root segment or by generating more overlay segments.

The XCAL parameter is necessary when exclusive references are made; normally, exclusive references cause the load module to be marked unexecutable. The XCAL option checks for valid exclusive references and only marks the load module unexecutable if there are invalid exclusive references. The LET option allows valid or invalid exclusive references; there will be unpredictable results if an invalid exclusive reference is executed.

The Boole and Babbage package can be used to determine how much time is spent in each path, and how often a given segment is called. This kind of information is essential in optimizing large overlay programs. See subsection 7.2 for information on the use of the Boole and Babbage system.

SECTION 23

REFERENCES

Key for Sources

IBM Standard Reference Library (SRL) manuals - GSFC Manual Library, Building 16 — A

GSFC Program Library documentation - Mrs. Pat Barnes, Program Librarian, Code 543.3, Building 3, 982-6796 — B

Newsletters, Bulletins - Contact Mr. Dave Kohnhorst, Code 601, extension 2133 to arrange for receipt of the GSFC Computer Newsletter. The M&DO Computer Bulletin (formerly T&DS Computer Bulletin) is circulated automatically to most M&DO computer users. — D

Subject	Document	Source
Current Local Information	GSFC Computer Newsletter published as needed by GSFC	D
	GSFC Microfilm Newsletter published as a subsection of the GSFC Newsletter	D
	M&DO Computer Bulletin prepared by the Computer Systems Branch on an as needed basis	D
IBM Bibliographies	GA22-6822: IBM System/360 Bibliography	A
	GA24-3089: IBM SRL Bibliography Supplement - Teleprocessing	A

Subject	Document	Source
IBM Master Index	IBM System/360 Operating System Systems Reference Library Master Index	A
GSFC Library Guides	Catalog of the GSFC Computer Program Library	B
	X-540-69-107: A Programmer's Guide to the Goddard Space Flight Center Computer Program Library	B
General System Documentation	GA22-6821: IBM System/360 Principles of Operation	A
	GC20-1619: Catalog of Programs for IBM System/360	A
	GC20-1646: A Programmer's Introduction to IBM System/360 Assembler Language	A
	GC20-1649: Introduction to IBM S/360 Direct Access Devices and Organization Methods	A
	GC20-1699: A Data Processing Glossary	A
	GC27-6909: IBM System/360 Operating System Graphic Programming Services for 2250 Display Unit	A
	GC27-6935: IBM System/360 Operating System Planning for Rollout/Rollin	A

Subject	Document	Source
General System Documentation	GC27-6942: IBM System/360 Operating System Introduction to Main Storage Hierarchy Support for IBM 2361 Models 1 and 2	A
	GC28-6534: IBM System/360 Operating System Introduction	A
	GC28-6535: IBM System/360 Operating System Concepts and Facilities	A
	GC28-6538: IBM System/360 Operating System Linkage Editor	A
	GC28-6540: IBM System/360 Operating System Operator's Guide	A
	GC28-6543: IBM System/360 Operating System Sort/Merge	A
	GC28-6550: IBM System/360 Operating System - System Programmer's Guide	A
	GC28-6554: IBM System/360 Operating System - System Generation	A
	GC28-6586: IBM System/360 Operating System Utilities	A
	GC28-6628: IBM System/360 Operating System - System Control Blocks	A
	GC28-6631: IBM System/360 Operating System Messages and Codes	A

Subject	Document	Source
General System Documentation	GC28-6646: IBM System/360 Operating System Supervisor and Data Management Services	A
	GC28-6647: IBM System/360 Operating System Supervisor and Data Management Macro Instructions	A
	GC28-6648: IBM System/360 Operating System TESTRAN	A
	GC28-6656: IBM System/360 Operating System Checkpoint/ Restart	A
	GC28-6662: IBM System/360 Operating System Sort/Merge Timing Estimates	A
	GC28-6670: IBM System/360 Operating System Programmer's Guide to Debugging	A
	GC28-6680: IBM System/360 Operating System Tape Labels	A
	GC28-6703: Job Control Language User's Guide	A
	GC28-6704: Job Control Language Reference	A
	GC28-6708: IBM System/360 Operating System Advanced Checkpoint/Restart Planning Guide	A
	GC28-6719: Service Aids (Release 19)	A

Subject	Document	Source
Assembly Language	GA22-6821: IBM System/360 Principles of Operation	A
	GC26-3756: IBM System/360 Operating System Assembler (F) Programmer's Guide	A
	GC28-6514: IBM System/360 Operating System Assembler Language	A
	GC28-6515: IBM System/360 FORTRAN IV Language	A
	GC28-6596: IBM System/360 FORTRAN IV Library Subprograms	A
	GC28-6817: IBM System/360 Operating System FORTRAN IV (G and H) Programmer's Guide	A
	GC28-6818: IBM System/360 FORTRAN IV Library: Mathematical and Service Subprograms	A
	GH20-0166: System/360 Scientific Subroutine Package Version III: Application Description	A
	GH20-0205: System/360 Scientific Subroutine Package Version III: Programmer's Manual	A
	ASA FORTRAN (ANSI, X3.9 - 1966)	B
PL/I	E20-0312: Preface to PL/I Programming in Scientific Computing	A
	GC28-6590: IBM System/360 Operating System PL/I Subroutine Library Computational Subroutines	A

Subject	Document	Source
PL/I	GC28-6594: IBM System/360 Operating System PL/I (F) Programmer's Guide	A
	GC28-6808: A Programming Language/One Primer	A
	GC28-8201: IBM System/360 PL/I Reference Manual	A
	GC33-2002: IBM System/360 Conversion Aids: FORTRAN IV-to-PL/I Language Conversion Program for IBM System/360 Operating System	A
	GH20-0544: System/360 Scientific Subroutine Package (PL/I) Application Description Manual	A
	GH20-0586: System/360 Scientific Subroutine Package, PL/I Program Description and Operations Manual	A
	GY33-6003: IBM System/360 PL/I Language Specifications	A
	SC20-1637: PL/I Guide for FORTRAN Users	A
	SC20-1651: A Guide to PL/I for Commercial Programmers	A
	SC20-1689: Introduction to the Compile-Time Facilities of PL/I	A
	360D-03.3.004: IBM Contributed Program Library, PL/I FORMAC Interpreter	B

Subject	Document	Source
Report Program Generator (RPG)	GC24-3337: IBM System/360 Operating System Report Program Generator Language	A
APL	GH20-0906: The APL/360 User's Manual	A
	GH20-0689: The APL/360 Primer	A
Remote Terminal Systems	CRBE User's Guide	B
	RITS User's Manual	B
	GA24-3125: IBM 1050, Operators Guide	A
	GA27-3005: IBM 2780 Data Transmission Terminal - Component Description	A
	GC30-2006: IBM System/360 Operating System Remote Job Entry	A
	GC30-2007: IBM System/360 Introduction to Teleprocessing	A
Graphics	TADPOLE User's Guide	B
	2260 Subroutine Package, by Frank Ross, Systems Programmer, Computer Systems Branch	B
	GC27-6912: IBM System/360 Operating System Graphic Programming Services for IBM 2260 Display Station (Local Attachment)	A

Subject	Document	Source
Graphics	GC27-6932: IBM System/360 Operating System Graphic Programming Services for FORTRAN IV	A
	GC27-6933: IBM System/360 Operating System User's Guide for Job Control from the IBM 2250 Display Unit	B
Plotters	CalComp Digital Recorder User's Manual, prepared by Computer Sciences Corporation, January 1967, with Update Packages A, B, C and D	B
	Gerber Plotter Subroutine Package for OS 360 User's Guide, prepared by Vitro Services, Inc., under NASA contract NAS5-9241.	B
	GSFC Newsletter No: 13, September 16, 1968	D
	Laboratory for Theoretical Studies, System 360, Bulletin No. 5 from Mr. P. Smidinger, September 26, 1966, subject "PRPLOT, A 360 Printer Plotting Program" with attachment from the University of Michigan Computing Center, dated March 1, 1961, on UMPLOT subroutine.	B

Subject	Document	Source
Plotters	Memorandum from Mr. D. Y. Sumida, Mathematics and Computing Branch, subject: "CPLOT", dated September 1, 1967, with earlier manual titled CPLOT CalComp Plotter Routine, Direct-Couple System for FORTRAN II, FORTRAN IV, Theoretical Division	B
	Programmer's Reference Manual for the Integrated Graphics Software Systems, Volume 1, Applications Programmer's Guide: Document 9500360, prepared by Stromberg Datagraphics, Inc. (formerly Stromberg-Carlson) and the Rand Corporation	B
	Universal SD-4060 System and Software Manual, Stromberg Datagraphics, Inc., June 22, 1970.	B
	SCOPLT, prepared by CalComp Systems Software	B
Conversion Aids	DACUT9	B
	DATCON (GSFC Program #A00042)	B

Subject	Document	Source
Conversion Aids	FAPCON (GSFC Program #G00199)	B
	SIFT (GSFC Program #G00073)	B
	Subroutine Package for Conversion of 7090 7-track Tapes to Formats Compatible to S/360	B
	User's Guide to Deblock, Conversion and Clock Subroutines	B
	User's Guide to Subroutine UNPACK	B
Automatic Flowcharting	GSFC documentation and various documents from Applied Data Research, Inc., on flowcharting programs written for both S/360 computers and for others - CDC, UNIVAC, SDS, and DDP.	B
Hardware, Operating Instructions	Computation Division ADP Equipment Guide	B
	A22-6884: IBM System/360 Model 65 Functional Characteristics	A
	A22-6889: IBM System/360 Model 75 Functional Characteristics	A
	A22-6907: IBM System/360 Model 91 Functional Characteristics	A
	GA22-6810: IBM System/360 System Summary	A
	GA22-6866: IBM System/360 Component Description 2400 Series Magnetic Tape Units	A

Subject	Document	Source
Hardware, Operating Instructions	GA22-6895: IBM System/360 Component Descriptions - 2301 Drum Storage and 2820 Storage Control	A
	GA24-3073: IBM 1403 Printer Component Description	A
	GA24-3231: IBM System/360 Model 30, Functional Characteristics	A
	GA26-3599: IBM System/360 Component Descriptions - 2314 Direct Access Storage Facility and 2844 Auxiliary Storage Control	A
	GA26-5889: IBM System/360 Model 20, System Summary	A
	GA26-5988: IBM System/360 Component Descriptions - DASD for 2841	A
	GA27-2700: IBM System/360 Component Description 2250 Display Unit Model 1	A
	GA27-3005: IBM 2780 Data Transmission Terminal - Component Description	A
	GC28-6540: IBM System/360 Operating System Operator's Guide	A
GPSS	GH20-0186: General Purpose Simulation System/360 - Application Description	A
	GH20-0304: General Purpose Simulation System 1360 Introduction - User's Manual	A

Subject	Document	Source
GPSS	GH20-0311: General Purpose Simulation System/360 - Operator's Manual	A
	GH20-0326: General Purpose Simulation System/360 - User's Manual	A
Miscellaneous	OSSLIP (GSFC #G00231)	B
	Re-entrant On-Line Assembler (ROLA2260) User's Guide, prepared by Computer Sciences Corporation, 1970 (GSFC Program G00310, REA360)	B
	Systems Measurement Software (SMS/360) User's Guide for PPE, Boole and Babbage, 1969	B
	Utility to Update Source and Object Files on Tape, by Charles R. Newman GSFC Code 551. (GSFC #D00145 and GSFC Document X-551-69-409)	B
Pocket Reference Cards	GX20-1703: IBM System/360 Reference Data Card	A
	GX20-1710: IBM 2314 Direct Access Storage Facility Capacity and Transmission Time Reference Card	A
	GX20-1733: IBM System/360 Operating System Data Management Macro Instructions	A
	GX20-1738: System/360 Reference Data Assembler Language Supervisor Macro Instructions	A

Subject	Document	Source
Pocket Reference Cards	GX20-1739: System/360 Reference Data Linkage Editor	A
	GX20-1760: Examples of Control Cards for System/360 Operating System	A

INDEX