File No. S360-26
Order No. GC33-4000-3

OS

IBM Systems Reference Library

OS ALGOL Programmer's Guide

Program Numbers: 360S-AL-531 (ALGOL Compiler)
360S-LM-532 (ALGOL Library)

OS Release 21

This publication describes how to compile, link-edit, and execute a program written in the IBM Operating System Algorithmic Language (ALGOL). It includes an introduction to the operating system and a description of the information listings that can be produced, the job control language, and the subroutine library. The publication also contains information about, and a description of, the OS ALGOL F Independent Component Release.
The intended audience for this publication includes application programmers, system programmers, and IBM system engineers.

Fourth Edition (January 1972)

This is a major revision of, and obsoletes, GC33 - 4000 - 2 and Technical Newsletter GN33 - 8091. Changes to the text and to illustrations are indicated by a vertical line to the left of the change.

This edition applies to release 21 of the IBM O] erating System and to all subsequent modifications until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the specifications herein. Before using this publication in connection with the operation of IBM systems, consult the latest SRL Newsletter, Order No. GN20 - 0360, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Program Product Center, 43 Schwabstrasse, 7000 Stuttgart, Germany. Comments become the property of IBM.

Preface

This publication is intended for use by Application Programmers, System Programmers and IBM Systems Engineers. A knowledge of ALGOL is assumed, and the reader is expected to be familiar with the prerequisite publication

| OS ALGOL Language, Order No. GC28-6615.

In Section 2, the description 'IBM-Supplied Cataloged Procedures' provides sufficient information to process and execute an ALGOL program that can use the IBM-supplied cataloged procedures without modification.

The rest of Section 2, together with information in Section 1 and the Appendixes, will be required for programs that cannot use the IBM-supplied cataloged procedures without modification.

The description of information listings in Section 3 and the list of diagnostic messages given in 'Appendix F' will be helpful in interpreting system output, especially for debugging.

An extensive index has been provided to assist the reader in using the manual for reference purposes.

This publication contains most of the information required by the Applications Programmer. The following publications are referred to within the text for information beyond the scope of this publication.

OS Assembler Language, Order No. GC28-6514

OS Loader and Linkage Editor, Order No. GC28-6538

OS JCL Reference, Order No. GC28-6704

OS Operator's Procedures, Order No. GC28-6692

OS Operator's Reference, Order No. GC28-6691

OS Utilities, Order No. GC28-6586

OS FORTRAN IV Library, Order No. GC28-6596

OS Messages and Codes, Order No. GC28-6631

OS Supervisor Services and Macro Instructions, Order No. GC28-6646

OS Data Management Macro Instructions, Order No. GC26-3794

OS Sysgen, Order No. GC28-6554

OS Data Management for System Programmers, Order No. GC28-6550.

Contents

Figures

LINKAGE EDITOR TABLE

Maintenance
The table of linkage editors has been updated for completeness.

MESSAGE IEX011I

Maintenance
The explanation of message IEX011I has been rewritten for clarity.

TITLE CHANGES

Maintenance
The names of reference publications have been changed to reflect their current titles.

Section 1: Introduction

The primary constituent of a System/360 data processing operation is a job. This basically, is the work the user requires the computer to do. To carry out a job, a computer needs two types of information -- a program and data.

- A program is a sequence of instructions which specify the actions to be performed by the computer. These instructions are written in a symbolic language and are translated into machine language by a processing program contained in the operating system before they are performed.
- Data is the information to be processed by the program. The source program is regarded as data while it is being processed by operating system programs to make it suitable for execution.

Discussions of the source program and the operating system appear below, followed by the machine configuration necessary to compile and execute an ALGOL job.

Source Program

For jobs discussed in this publication, the source program will be written primarily in System/360 Operating System ALGOL (Algorithmic Language). This is defined in OS ALGOL Language. In addition the programmer must observe the restrictions, caused by internal capacity limitations, listed in Section 4.

An ALGOL source program may be written in free form on any 80-column coding sheet. The program text is contained in columns 1 to 72. Columns 73 to 80 can be used by the programmer for program identification. To avoid confusion with job control statements (see 'Operating System'), the character sequences // and /* must not be used in columns 1 and 2. It is possible to avoid these combinations since these sequences are syntactically incorrect outside strings and when they occur within string quotes (' '). Two character sets are available for punching the source program into a card deck (see 'Appendix C').

For operations that require more precise control over the computer than can be provided by ALGOL, subprograms written in Assembler language can be included in the ALGOL program (see Section 4). Assembler language subprograms can also be used as a link to other languages, such as PL/1, COBOL and FORTRAN. The Assembler language is defined in OS Assembler Language.

Operating System

The System/360 Operating System is a set of IBM-supplied control and processing programs (supplemented if necessary by user-written programs) that assist the programmer to use the computer efficiently. The operating system selected for a particular installation is generated during the initial setting-up of the computer, by a process known as system generation.

JOB CONTROL

Operating system instructions (known as job control statements) must be added to the source program to control its handling within the operating system and to specify the data management facilities required.

These statements do not need to be specified until the program is ready to be executed. This means that the program can be prepared independently of installation considerations.

Eight types of statements are available, which, in conjunction with associated parameters, can supply all information required by the operating system for job control. To save programming effort, commonly used sequences of control statements can be stored by the system for subsequent recall by identifying names. These sequences are known as cataloged procedures.

JOB is the first statement of each job. It indicates that a new job is beginning and, consequently, that the previous job has ended. A job can be divided into a number of job steps, which can be inter-related to improve processing efficiency. For example, the execution of one job step can be made dependent on the result of a previous one. This is an important feature of the operating system, and users are recommended to exploit it as fully as possible.

EXEC (Execution) is the first statement in each job step. It specifies the program or cataloged procedure to be executed, and must be included even if the job consists of only one job step.

DD (Data Definition) is the statement used to describe a data set and to specify associated data control block information. It also specifies input/output (I/O) device assignment. One or more DD statements are usually required for each job step.

In addition to the above JCL statements, the command statement is used to place operator commands into the input stream, the null statement indicates the end of the last job in the input stream, and the delimiter statement separates data from subsequent control statements when sequential scheduling is used. The command statement, when used, must immediately precede a JOB, EXEC or null statement.

The job control statements required for an ALGOL source program are described in Section 2. For a complete discussion of job control language, see OS JCL Reference.

CONTROL PROGRAM

The control program is the primary program within the operating system. It is divided into a number of functions. Those affecting the applications programmer are described in the following text.

Job Scheduling

A job scheduler is included as part of the control program to control the flow of jobs and allocate the I/O devices required. Two forms of job scheduling are available.

With sequential scheduling the jobs are carried out in the order they are presented in the input stream to the computer.

With priority scheduling a summary of the input job stream is stored on a direct access device and jobs are carried out in order of priority (as specified in the JOB control statement). Any hold-up in the execution of a program, due , for example, to a delay in mounting a volume, will cause the job scheduler to select the next job available (in order of priority) and then return to the higher priority job when it is ready.

Supervisor

The supervisor is a set of subroutines, included in the control program, for transferring control of the central processing unit of the computer from one program to another and co-ordinating I/O operations. Initialization and termination of all programs described in this publication are achieved using the standard method given in OS Supervisor Services and Macro Instructions.

Data Management

(This sub-section is a summary of data management facilities. Full details are given in OS Data Management Services.

Data Sets: Data is usually stored on I/O devices and is only brought into main storage for processing. It is organized into data sets. These are collections of records that are logically related (for example, a set of test readings).

System/360 Operating System allows a data set to be identified and accessed by symbolic name only, without any reference to its location on the storage device. To do this, the operating system builds a catalog of data set locations against names. This catalog resides on one or more direct access volumes. A volume is one complete physical unit of storage such as a tape reel or a disk pack. It may contain a number of data sets, or alternatively one data set may stretch over a number of volumes. Data sets are created using DD statements.

Data Control Blocks: The operating system must be provided with information describing the characteristics of a data set before the data set can be processed. This information is assembled in the data control block (DCB) associated with each data set. Data control blocks are automatically created for each data set that is to be processed by the program, and are completed from two sources:

1. Any information provided in the program is included first.

2. Information provided by the DD statement is then included, but cannot over-ride any information stated in the program.

In the case of an existing data set, further information is taken from the data-set label. Again, this cannot over-ride previously inserted information. Any DCB information provided by the programmer is checked by an appropriate routine to ensure its validity and to assign default values.

Data Set Labels: Data set labels, if requested by the programmer in the DD statement, are created by the operating system to store information relevant to the data set, such as name and retention period. Tapes must have been previously initialized. The labels can supplement information in the data control block and serve as identifiers during accessing. They are positioned at the beginning and end of the data set.

Records and Blocks: Records are the smallest items of data which can be read or written separately. Their length can be specified as fixed, variable or undefined. The unit of length is known as a byte, which is normally equivalent to one character. For mechanical reasons it is necessary tc have a fixed-length gap between each record. This means that the smaller the average length of the records, the smaller the amount of information that can be stored in a given area of storage. To conserve space a number of records can be grouped together to form a blcck, which is treated as a single record for I/O operations. The complete block is read into main storage and then unblocked for the required record to be processed. Record format and block size are defined in the data control block. For fixed-length records block size must be a multiple of record length. This multiplication factor is known as the blocking factor.

A control character can be specified for inclusion in each record of a data set. This selects carriage control when the data set is printed, or stacker when the data set is punched.

Data Set Organization: According to how they are going to be used, records can be organized within the data set in a number of ways, as described below. Only sequential organization can be used with ALGOL.

Sequential organization is a characteristic of I/O devices such as tape units. To access a particular record the data set must be read sequentially until the record is found. This is satisfactory for many applications where a large proportion of the records will be required on each run but could be time-consuming where data is being accessed randomly.

To avoid reading each record in turn the indexed sequential method is often employed, in which the location of the required record is found from an index at the beginning of its data set. On a disk pack the specification of a record location is broken down into two levels - cylinder and track. Each level has its own index. With large data sets up to three levels of master index can also be used. Overflow areas are provided for the primary storage area so that insertions can be made.

Alternatively, a data set can be partitioned into blocks of identical format called members. A directcry is built up at the beginning of the data set so that each member can be accessed independently by specifying its name as a suffix to the data set name. This form of data set is described as a library.

Direct organization allows records to be stored and retrieved using an absolute or relative address (cylinder, head, track). For example, an algorithm could be used tc determine the address from data in the record.

Access Language: When using assembler language, two access languages are available to store and retrieve records. The queued access language provides a full range of buffering and blocking facilities to improve processing efficiency. It can only be used with sequential and indexed sequential data sets.

The basic access language gives the programmer more direct control over the I/O device but does not provide buffering and blocking facilities. These must be constructed by the user (see OS Supervisor Services and Macro Instructions).

Access Methods: The data set organization and access language used are combined to fully describe the method of handling a data set, for example, Queued Sequential Access Method, Basic Partitioned Access Method, etc. The access method is specified in the data control block.

Input/Output Devices: Data can be stored on a number of input/output devices depending, among other things, on the method of data set organization required. The devices most commonly used in scientific and engineering installaticns are:

Card readers and punches Printers (output only) Paper tape devices Magnetic tape devices	All data handled by these devices is sequentially organized.
Disk storage devices Data cell storage devices Drum storage devices	These are direct access devices and can be used for sequential, indexed sequential or partitioned organization.

A console typewriter is used for direct two-way communication between the operator and the operating system.

Areas of main storage known as buffers are used to provide overlapping cf reading, writing and processing operations. The transfer of data between main storage and I/O devices is controlled through units known as channels.

PROCESSING PROGRAMS

In addition to the control program, a number of processing programs are included in the operating system, depending on the requirements of the installation. To carry out a job that contains a source program written in ALGOL the following processing programs are required:

1. ALGOL compiler
2. Linkage editor or loader

The ALGOL compiler processes the source program to translate it into machine language. The translated source program (known as the object module) is then processed by the linkage editor or, alternatively, by the loader. The linkage editor and the loader have a common function: they combine various routines, drawn from the ALGOL library (see 'Appendix A'), with the object module. When the linkage editor is used, the resulting program (known as the load module) is stored on an auxiliary storage device; subsequently, the load module may be read into main storage and executed. When the loader is used, the resulting program is executed directly without being transferred to auxiliary storage. The basic sequence of operations involved in compiling, linkage editing and executing or in compiling and loading an ALGOL program, is pictured in Figure 1.

ALGOL Compiler

This processing program is available for the F level of main storage size, and requires a minimum of 44K bytes. If extra storage capacity is provided it is used tc increase compiler capacity (see Figure 6).

Initialization and Termination: The standard method is used for initialization and termination of the compiler (see 'Supervisor'). At the end of the compilation one of the following return codes is generated:

0 normal conclusion. Cbject module has been generated unless both the NODECK and NOLOAD operations (see 'Appendix E') are specified in the invoking statement. No diagnostic messages have been listed.

4 object module has been generated unless both the NODECK and NCLOAD options are specified. Only warning diagnosic messages (severity code W) have been listed.

12 process has been completed but a complete object module could not be generated due to a serious error. Diagnostic messages (severity codes S and possibly W) have been listed.

16 process has been terminated abnormally due to a terminating error. A complete object module could therefore not be generated. Diagnostic messages (severity codes T and possibly W and S) have been listed. The severity codes are described in 'Appendix F'.

Output: A successful compilation of an ALGOL source program produces the following output:

1. An object module (described in 'Appendix D') which can be
   a. included in a data set for use as input to the linkage editor (optional) or the loader (optional)
   b. included in another data set to give some other form of output, such as a card deck (optional)
2. Information listings (described in Section 3)

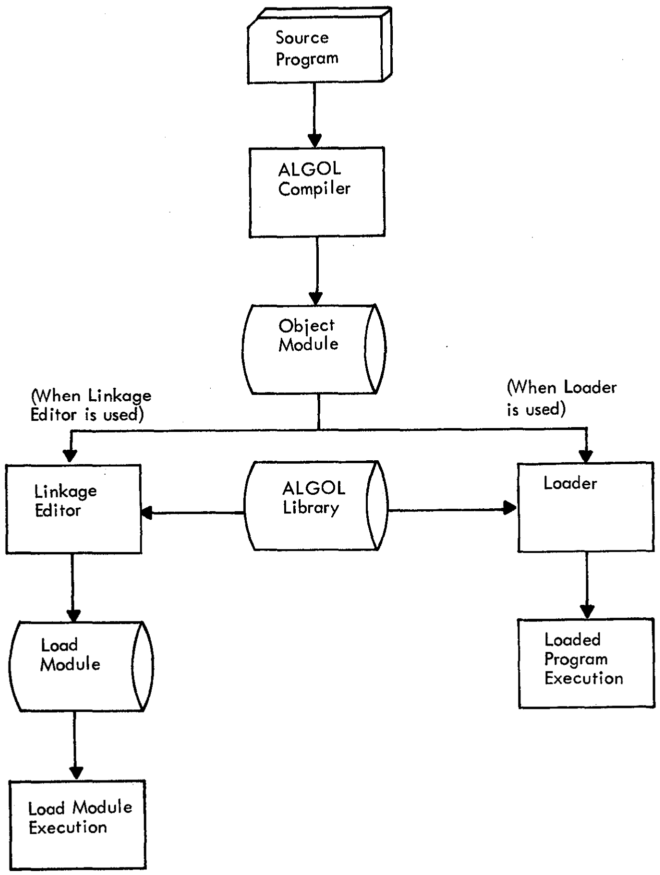


Figure 1. Basic Flowchart for Handling an ALGOL Program

Linkage Editor

The linkage editor is a standard processing program used for all languages accepted by the System/360. For ALGOL, it is used to include routines from the ALGOL library. It also has a wide range of optional functions, and is available for two levels of main storage size - E level (where it requires 15K or 18K bytes) and F level (where it requires 44K or 88K bytes). A full description is contained in CS Loader and Linkage Editor.

Initialization and Termination: The standard method is used for initialization and termination of the linkage editor (see 'Supervisor'). At the end of linkage editing one of the following return codes is generated:

0 meaning normal conclusion. A load module has been produced.

4 meaning a load module has been produced but a severity 1 error, which may cause an error at execution time has been detected and listed.

8 meaning a load module has been produced but a severity 2 errcr, which may cause an abnormal termination at execution time, has been detected and listed.

12 meaning a load module has been produced but a severity 3 errcr, which will cause an abnormal termination at execution time, has been detected and listed.

16 meaning process has been terminated abnormally. A severity 4 error has been listed.

Output: The following output can be prcduced by the linkage editor:

- A load module data set, stored on the output library SYSLMOD.
- Information listings (described in Section 3).

LOAD MODULE EXECUTICN

The load module produced by the linkage editor is loaded into main storage by the supervisor. When the loading operating is complete, the supervisor passes control to the load module, which is then executed.

Initialization and Termination: The standard method is used for initialization and termination of the load module (see 'Supervisor'). At the end of the execution, one of the following return codes is generated:

0 meaning normal execution has been performed.

4 meaning execution has been abnormally terminated due to an error. A diagnostic message has been listed.

Output: The following output is produced by a successful execution of a load module:

- Results, etc., as specified by the programmer.
- Information listings (described in Section 3).

Return Code	Loader Return Code	Loaded Program Return Code	Conclusion or Meaning
0	0	0	Program loaded successfully, execution successful
	4	0	The loader found an error that may cause an error during execution but no error occurred during execution of the loaded program
	8 LET	0	
4	0	4	Program loaded successfully, but an error occurred during execution of the loaded program.
	4	4	The loader found an error that may cause an error during execution and an error did occurr during execution of the loaded program.
	8 LET	4	
8	8		The loader found an error that could make execution impossible - the loaded program was not executed.
12	12		Loader could not load program successfully, execution impossible.
16	16		Loader could not load program, execution impossible.
Error Diagnostic (SYSPRINT or SYSLOUT data set) for the loader will show the severity of errors ecountered by the loader.			

Figure 2. Loader Step Return Codes

Loader

The loader is a standard processing program of the IBM System/360 Operating System. Its function is to load an object module, to link various required submodules from a submodule library, and to execute the resulting program. Processing of the object module by the loader and execution of the program are performed in a single step. By eliminating the intermediate output and retrieval of load modules involved when linkage editing and execution are performed in separate steps, the loader can be used to achieve a significant reduction in throughput time. The loader can also be used to load and execute a linkage editor produced load module. A full description of the loader is provided in OS Loader and Linkage Editor.

Initialization and Termination: The standard method is used for initialization and termination of a processed object or load module (see 'Supervisor'). At the end of loading, a return code is generated which reflects the results of processing by the loader, or the results of execution of the loaded program. The possible return codes are shown in Figure 2.

Output: The following output is produced by a successful loader step:

- Information listings (described in Section 3).
- Results, etc., as specified by the programmer.

Machine Configuration

To successfully carry out a job containing a source program written in ALGOL, a certain minimum machine configuration must be available. This is:

- An IBM System/360 Model 30, 40, 50, 65, 75, 85 or 91 with the scientific instruction set or an IBM System/370 Model 135 (or higher) with the scientific instruction set. Main storage size depends on the program being executed.
- For compilation, at least 64K bytes.
- For linkage editing, at least 32K bytes.

- For load module execution, variable, depending on the size and arrangement of the source program.
- For loading, 17K bytes plus the loaded program size (for MFT systems) or 18K bytes plus the loaded program size (for MVT systems).

These figures include the space used by the control program of the operating system.

- In a minimum configuration, all data sets may use a single direct-access I/O device, provided that the total size of the data sets which exist at any one time does not exceed the capacity of the device. A card reader and printer will also be needed, but these do not have to be part of the System/360 configuration.
- A console typewriter may be required for diagnostic messages if there is an error on the data set used for output listings, and also to allow direct two-way communication between the operator and the operating system.

Section 2: Source Program Handling

This section explains the job control statements which must be provided with each source program. These statements can either be written for each job, or a standard job control procedure can be written and cataloged in the operating system for use with a range of jobs.

Using such a cataloged procedure minimizes the number of job control statements that must be supplied by the programmer with each job. Therefore IBM provides:

- Four basic cataloged procedures for use with ALGOL.
- The means to temporarily over-ride these procedures if the user requires different or additional system support to that provided.
- The means for the user to modify permanently the IBM-supplied cataloged procedures or to write his own procedures and catalog them for permanent reference.

In the statement formats used in this section upper-case words must be coded exactly as they appear: lower case words are used to indicate where the programmer must supply information according to his own requirements.

IBM-Supplied Cataloged Procedures

The four cataloged procedures for ALGOL which are supplied by IBM are:

ALGOFC	compilation only
ALGOFCL	compilation and linkage editing
ALGOFCLG	compilation, linkage editing and execution
ALGOFCG	compilation and loading

To invoke these cataloged procedures, the programmer must supply the following job control statements:

1. A JOB statement to indicate the start of the job.
2. An EXEC statement indicating the name of the cataloged procedure to be used.
3. DD statements indicating the location of the source program and, for execution, the data sets used or created by the load module.

The following text indicates the minimum contents of these statements. For requirements beyond this, reference shculd be made to 'Appendix E'.

COMPILATION

The cataloged procedure to compile a scurce program is ALGOFC. The job control statements used in this cataloged procedure are shown in 'Appendix B'. The following statements can be used to invoke the ALGOFC cataloged procedure:

```
//jobname  JOB
//         EXEC ALGOFC
//SYSIN    DD {* or parameters defining an
                 input set containing the
                 source program}
```

where 'jobname' is the name of the job. If DD * is used then the source program must follow immediately afterwards in the input stream. For sequential scheduling, the source program must then be followed by a delimiter statement (/*).

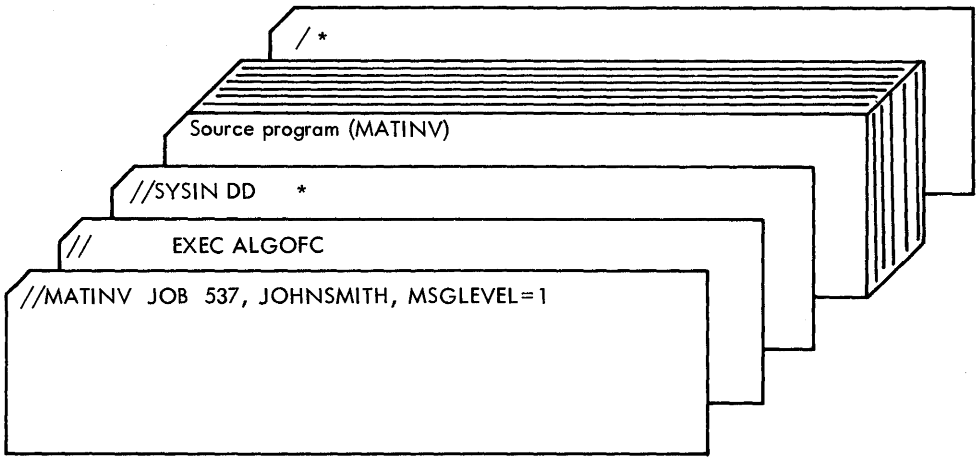
If more than one source program is to be compiled in the same job, all job control statements except the JOB statement must be repeated for each source program.

A sample deck of job control statements to compile an ALGOL source program is shown in Figure 3.

COMPILATION AND LINKAGE EDITING

The cataloged procedure to compile an ALGOL source program and linkage edit the resulting object module is ALGOFCL. The job control statements used in this cataloged procedure are shown in 'Appendix B'. The following statements can be used to invoke the ALGOFCL cataloged procedure:

```
//jobname  JOB
//         EXEC ALGOFCL
//SYSIN    DD  {* or parameters defining an
                  input data set containing
                  the source program}
```



```
/*
Source program (MATINV)
//SYSIN DD    *
//          EXEC ALGOFC
//MATINV  JOB  537, JOHNSMITH, MSGLEVEL=1
```

Figure 3. Sample Deck for Using ALGOFC Cataloged Procedure with a Single Source Program. This job compiles the MATINV source program used in Example 1 of 'Appendix E'.

where 'jobname' is the name assigned to the job. If DD * is used, then the source program must follow immediately afterwards in the input stream. For sequential scheduling, the source program must then be followed by a delimiter statement (/*).

If more than one source program is to be processed in the same job, then all job control statements except the JOB statement must be repeated for each source program.

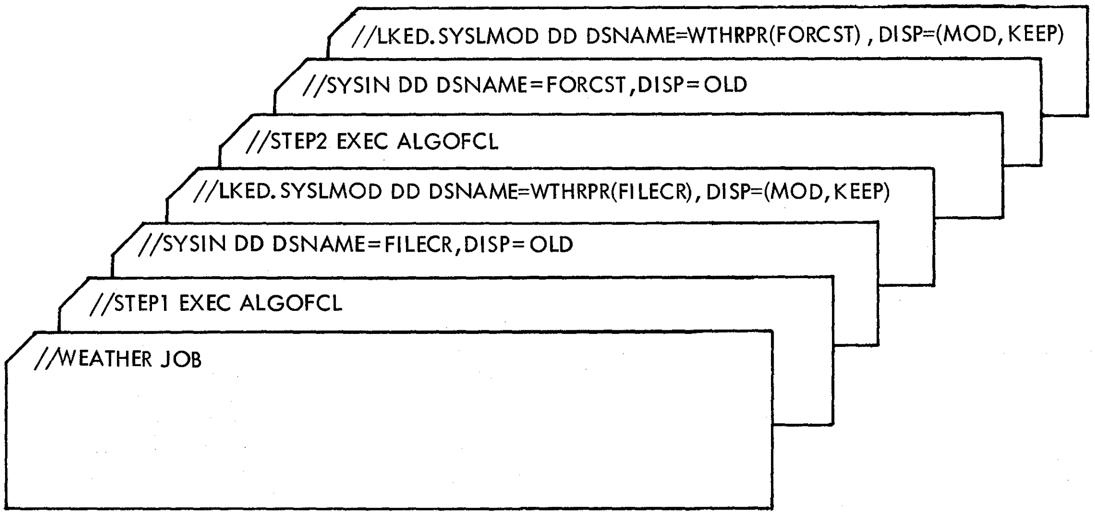
If it is required to keep a load module for use in a later job (as in the case when the load module is a precompiled procedure), then the SYSLMOD DD statement in the cataloged procedure must be over-ridden to specify a permanent data set. This has to be done for each load module that is kept. The over-riding statement is placed at the end of the job step to which it applies, and has the form:

```
//LKED.SYSLMOD DD DSNAME=dsname(member),
//                DISP=(MOD,KEEP)
```

where 'DSNAME' is the name of a partitioned data set and 'member' is the member name assigned to the load module on the partitioned data set.

Figure 36 shows the job control statements needed to compile and linkage edit a precompiled procedure.

A sample deck of job control statements to compile and linkage edit two source programs is shown in Figure 4.



```
//LKED.SYSLMOD DD DSNAME=WTHRPR(FORCST),DISP=(MOD,KEEP)
//SYSIN DD DSNAME=FORCST,DISP=OLD
//STEP2 EXEC ALGOFCL
//LKED.SYSLMOD DD DSNAME=WTHRPR(FILECR),DISP=(MOD,KEEP)
//SYSIN DD DSNAME=FILECR,DISP=OLD
//STEP1 EXEC ALGOFCL
//WEATHER JOB
```

Figure 4. Sample Deck for Using ALGOFCL Cataloged Procedure with two Source Programs. These two job steps compile and linkage edit the two source programs used in Example 3 of 'Appendix E'. Both source programs have been previously stored on intermediate I/O devices.

COMPILATION, LINKAGE EDITING AND EXECUTION

The cataloged procedure used to compile an ALGOL source program, linkage edit the resulting object module, and execute the load module produced by the linkage editor is ALGOFCLG.

The statements used in this cataloged procedure are shown in 'Appendix B'. The following statements can be used to invoke the ALGOFCLG cataloged procedure:

```
//jobname  JOB
//JOBLIB   DD DSNAME=dsname1,DISP=OLD
//         EXEC ALGOFCLG
//SYSIN    DD  {* or parameters defining an
                input data set containing
                the source program}
//GO.ALGLDD02 DD DSNAME=dsname2
       .     .     .       .
       .     .     .       .
       .     .     .       .
//GO.ALGLDD15 DD DSNAME=dsname15
```

where 'jobname' is the name assigned to the job. 'dsname1' is the name of a data set that contains a precompiled procedure (see Section 4) which is called by the load module being executed. The DD statement containing dsname1 need not be used if no precompiled procedure is used.

For a description of the correct use of the JOBLIB DD statement when more than one precompiled procedure is used in a job, or when a precompiled procedure resides on more than one data set, see 'Data Set Concatenation' in 'Appendix E'.

'dsname2'...'dsname15' are the names of input data sets required by the load module at execution time and output data sets to be created at execution time. In addition, two data sets for printed output (ddnames SYSPRINT and ALGDD01) are supplied by the cataloged procedure, and a data set for input only can be specified by using the following statement after the invoking sequence just given.

```
//GO.SYSIN DD  {* or parameters defining an
                  input data set}
```

If DD * is used then the data must follow immediately afterwards in the input stream. For sequential scheduling, the data must be followed by a delimiter statement (/*).

If more than one source program is to be processed and executed in the same job, then all job control statements except the JCB statement and the JOBLIB DD statement must be repeated for each source program.

A sample deck of job control statements required to compile, linkage edit and execute three source programs is shown in Figure 33.

COMPILATION AND LOADING

The cataloged procedure to compile a source program and to load and execute the compiled program (by use of the loader) is ALGOFCG. The job control statements used in this procedure are shown in 'Appendix B'.

The following job control statements may be used to invoke the ALGCFCG cataloged procedure:

```
//jobname JOB
//          EXEC  ALGOFCG
//ALGOL.SYSIN  DD {* or parameters defining
                    an input data set
                    containing the
                    source program}
//GO.ALGLDD02  DD  DSNAME=dsname2
     .          .     .     .
     .          .     .     .
     .          .     .     .
//GO.ALGLDD15  DD  DSNAME=dsname15
```

where 'jobname' is the name assigned to the job. 'dsname2'... 'dsname15' are the names of data sets required by and/or to be created by the loaded module. Three data sets for printed output (ddnames SYSLOUT, SYSPRINT and ALGLDD01) are supplied by the cataloged procedure. An additional data set for input only can be specified by using the following statement after the invoking sequence just given.

```
//GO.SYSIN DD {* or parameters defining an
                 input data set}
```

If DD * is used, then the data must follow immediately afterwards in the input stream. For sequential scheduling, the data must be followed by a delimiter statement (/*). If more than one source program is to be processed and executed in the same job, then all job control statements except the JOB statement must be repeated for each source program.

At system generation time, the user is advised to specify SYSLOUT as an alternative ddname to SYSPRINT for the printer data set used by the loader (see <u>OS Sysgen</u>). The loader cannot be used to load an ALGOL object module if the SYSPRINT data set is routed to a direct access device and no alternative name has been specified for the printer data set used by the loader.

A sample job containing the control statements needed to compile and load an ALGOL source program, by use of the ALGOFCG cataloged procedure, is shown in Figure 37.

OVER-RIDING CATALOGED PROCEDURES

The programmer can change any of the statements in a cataloged procedure, except the name of the program in a EXEC statement.

These over-riding conditions are temporary, and will be in effect only until the next job step is started. The following text describes methods of temporarily modifying existing parameters and adding new parameters to the EXEC and DD statements used in the cataloged procedures. The full list of parameters available to the ALGOL programmer for these statements, and detailed explanations of the parameters, is given in 'Appendix E'. The EXEC and DD statements used in the IBM-supplied cataloged procedures are shown in 'Appendix B'.

Over-riding EXEC Statements

In the EXEC statement, the programmer can change or add any of the keyword parameters by using the following format:

```
  keyword.procstep=option
```

where

'keyword' denotes any one of the parameters COND, PARM, ACCT, TIME, REGION or DPRTY that is to be changed or added to the procedure job step. TIME, REGION and DPRTY are valid only for priority scheduling.

'procstep' is the procedure job step in which the change or addition is to occur: either ALGOL, LKED or GO.

'option' is the new option required.

For example, if the EXEC statement used to invoke the ALGOFCLG cataloged procedure was written as:

```
//stepname EXEC ALGOFCLG,PARM.ALGOL=DECK,
//            PARM.LKED=XREF,
//            COND.GO=(3,LT,stepname.ALGOL)
```

then the following changes would be made to the ALGOFCLG cataloged procedure:

1. In the PARM parameter of the job step ALGOL, the option DECK would be used instead of the default option NODECK (assuming that the standard default NODECK was not changed at system generation). Over-riding this option will not affect the other default options assumed for this parameter.

2. In the job step LKED, the option XREF is specified for the PARM parameter. Since the options specified in the cataloged procedure were XREF, LIST and LET, this statement has the effect of deleting the options LIST and LET since they were not default options.

3. In the job step GO, the COND parameter code is changed from 5, as it appears in the cataloged procedure, to 3. In this example, the code 3 causes the job step GO to be bypassed if a warning message is generated during the job step ALGOL. Note that although the other options (LT and ALGOL) are not to be altered, the entire parameter being modified must be respecified.

If 'procstep' is not specified when over-riding a multi-step cataloged procedure, the operating system makes the following assumptions:

- COND, ACCT, REGION and DPRTY parameters apply to all procedure job steps.

- A PARM parameter applies to the first procedure job step and any options already specified in the PARM parameters for the remaining procedure job steps are cancelled.

- A TIME parameter specifies the computing time for the entire job and any options already specified in the TIME parameters for individual procedure job steps are cancelled.

Over-riding DD Statements

An additional DD statement is used in the invoking sequence for each DD statement in the cataloged procedure that is to be over-ridden. The following format is used:

```
//procstep.ddname DD parameter list
```

where

'procstep' is the procedure job step containing the DD statement to be over-ridden: either ALGOL, LKED or GO. If 'procstep' is omitted, then the first procedure job step is assumed. 'ddname' is the name of the DD statement to be over-ridden.

'parameter list' is the list of parameters that are being added or changed. In both cases the whole parameter must be specified. Unchanged parameters in the original statement need not be specified. For example, the statement

```
//ALGOL.SYSLIN DD SPACE=(400,(80,10))
```

will change the SPACE parameter of the SYSLIN DD statement in the ALGOL job step so that space will be allocated for 80 physical records instead of 40.

DD statements that are used to over-ride other DD statements in the cataloged procedures must be placed immediately after the EXEC statement invoking the cataloged procedure, and must be in the same order as their corresponding DD statements in the cataloged procedures.

Adding DD Statements

Complete, new DD statements that are to be added to the cataloged procedure use the same format as over-riding DD statements. The 'ddname' specified must not exist in the job step specified by 'procstep'. These new DD statements must follow immediately after the over-riding DD statements which apply to the same procedure job step.

User-Written Procedures

To supplement IBM-supplied cataloged procedures, the user can add his own procedures to the procedure library. However, it is not necessary to include the procedures in SYS1.PROCLIB until they have been tested. It is advisable to test the procedures as in-stream procedures (procedures included in the input deck), before they are cataloged. By using this facility the need for a job step to catalog the procedure in test runs is eliminated. For further information on in-stream procedures refer to OS JCL Reference. Cataloging procedures is accomplished using the IEBUPDTE utility program, described in OS Utilities.

The statements required in a procedure are:

- EXEC statements to invoke the programs.
- DD statements to define the data sets used by the programs.

Information required to write procedures is contained in the following text and in Appendix E.

COMPILATION

Invoking Statement

The ALGOL compiler consists of ten load modules contained in the link library, SYS1.LINKLIB, of the operating system. The compiler is activated by invoking its first load module, named ALGOL, which then internally invokes the other load modules of the compiler.

The usual method of invoking the compiler is by means of an EXEC statement of the form

```
//stepname EXEC PGM=ALGOL
```

where 'stepname' is the name assigned to the job step (optional).

Other EXEC statement parameters may be included if required (see 'Appendix E').

(A method of dynamically invoking the compiler within a job step, by means of the CALL, LINK, XCTL or ATTACH macro instructions, is described in Section 4.)

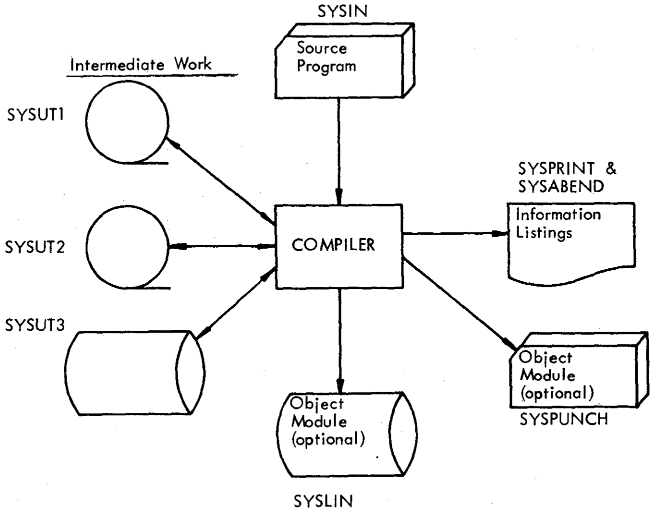


Figure 5. Flowchart Showing Data Sets Used by the Compiler

Data Sets Used

The data sets used in the compilation process are illustrated in Figure 5, and described in Figure 6. These data sets must be specified by the programmer with suitable DD statements.

Blocksize DCB information may be specified by the user for SYSIN, SYSLIN, SYSPRINT and SYSPUNCH. The maximum blocking factor depends on the main storage size available (see Figure 7). Record length is fixed at 80 bytes for SYSIN, SYSLIN and SYSPUNCH, and 91 bytes for SYSPRINT.

The space required for the compiler data sets depends on the size and structure of the source program; however, it can be assumed that only in rare cases will the object module exceed four times the source program, and usually much less will be required.

Purpose	Standard ddname	Devices required
For ALGOL source program	SYSIN	Card reader[1]
For object module to be used by linkage editor	SYSLIN	Direct access or magnetic tape
For compilation listings	SYSPRINT	Printer
For object module (copied from SYSLIN)	SYSPUNCH	Card punch[1]
For intermediate compiler working	SYSUT1	Direct access or magnetic tape
For intermediate compiler working	SYSUT2	Direct access or magnetic tape
For intermediate compiler working	SYSUT 3	Direct access

[1] Some form of intermediate storage, such as magnetic tape, may be used to reduce I/O delay for the central processing unit.

Figure 6. Data Sets Used by the ALGOL Compiler

The primary quantity specified in the SPACE parameter of the DD statements fcr SYSUT1, SYSUT2 and SYSUT3 must be large enough to contain the entire data set. The use of a secondary quantity for any of these data sets will increase the need fcr main storage by 40 percent. The following estimates can be used to allocate space on a 2311 direct access device:

SYSUT1 - 1 track per 100 source cards
SYSUT2 - 1 track per 100 source cards
SYSUT3 - 1 track per 200 source cards.

Processing of all data sets by the compiler is independent of the I/O device used except for the intermediate work data sets. These require magnetic tape or direct access devices.

LINKAGE EDITING

Invoking Statement

The linkage editor is usually invoked with an EXEC statement of the form:

```
//stepname EXEC PGM=IEWL
```

where 'stepname' is the name assigned to the job step (optional).

Other EXEC statement parameters may be included if required (see 'Appendix E'). IEWL specifies the highest-level linkage editor in the installation operating system.

(A method of dynamically invoking the linkage editor within a job step, by means of the CALL, LINK, XCTL or ATTACH instructions, is described in Section 4.)

Main storage sizes (in bytes) at which changes occur	Maximum blocking factor			
	SYSIN	SYSPRINT	SYSLIN	SYSPUNCH
45056 (44K)	5	5	5	1
51200 (50K)	5	5	5	5
59392 (58K)	5	5	5	5
67584 (66K)	5	5	5	5
77824 (76K)	5	5	5	5
90112 (88K)	20	20	40	20
104448 (102K)	20	20	40	20
120832 (118K)	20	20	40	20
139264 (136K)	20	20	40	20
159744 (156K)	20	20	40	20
184320 (180K)	40	40	40	40
212992 (208K)	40	40	40	40

Figure 7. Effect on Compiler Data Sets if more than 44K Bytes of Main Storage is Available.
The capacity of internal tables in the compiler is increased at each of the main storage sizes listed in this table, allowing, for example, a larger number of identifiers to be included in the source program. Therefore to get optimum performance, the user is recommended to use this list when specifying main storage size available to the compiler.

Data Sets Used

The data sets used by the linkage editor (see Figures 8 and 9) must be defined by the programmer with suitable DD statements.

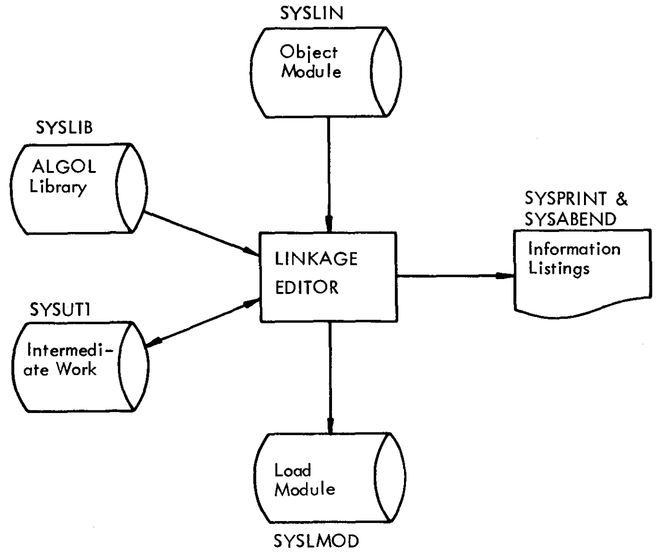


Figure 8. Flowchart Showing Data Sets Used by the Linkage Editor

Blocksize DCB information may be specified by the user for SYSLIN and SYSPRINT if the F level linkage editor is being used. Maximum blocking factor is 5 when 44K bytes of main storage size is available, and 40 when 88K bytes is available. Record length is fixed at 80 bytes for SYSIN and 121 bytes for SYSPRINT.

LOAD MODULE EXECUTION

Invoking Statement

The usual method of invoking the load module generated by the linkage editor is with EXEC statement of the form:

```
//stepname EXEC PGM=member name
```

where 'stepname' is the name assigned to the job step (optional).

'member name' indicates the name of the partitioned data set member which contains the load module. This name is specified by the programmer in the SYSLMOD DD statement for the linkage editor. Cther EXEC statement parameters may be included if required (see 'Appendix E').

(A method of dynamically invoking the load module within a job step, by means of the CALL, LINK, XCTL or ATTACH macro-instructions is described in Section 4.)

Purpose	Standard ddname	Devices used
For object module input	SYSLIN	Direct access or magnetic tape
For load module output, stored as a member of a partitioned data set	SYSLMOD	Direct access
For ALGOL library, SYS1.ALGLIB. A partitioned data set containing routines in load module form	SYSLIB	Direct access
For linkage editing listings	SYSPRINT	Printer[1]
For intermediate linkage editor working	SYSUT1	Direct access or magnetic tape

[1] Some form of intermediate storage, such as magnetic tape, may be used to reduce I/O delay for the central processing unit.

Figure 9. Data Sets Used by the Linkage Editor

Data Sets Used

Up to 16 data sets for use at execution time may be specified by the programmer in the ALGOL source program by using the appropriate data set number. The numbers used and the corresponding names of their DD statements are listed below.

Data set number used in ALGOL SOURCE PROGRAM	Corresponding DDNAMES
0	SYSIN
1	ALGLDD01
2	ALGLDD02
3	ALGLDD03
4	ALGLDD04
5	ALGLDD05
6	ALGLDD06
7	ALGLDD07
8	ALGLDD08
9	ALGLDD09
10	ALGLDD10
11	ALGLDD11
12	ALGLDD12
13	ALGLDD13
14	ALGLDD14
15	ALGLDD15

Any reference to a data set number by an I/O procedure within an ALGOL source program is translated into a reference to a data control block using the corresponding ddname. It is the responsibility of the programmer to supply the DD statements which correspond to the data set numbers used in the ALGOL source program.

The execution time data sets are illustrated in Figure 10 and described in Figure 11. For ALGLDD02 to ALGLDD15, case 1 in the column showing device used, applies if the source program contains any of the following:

- A backward respositioning specification by the procedures SYSACT4 or SYSACT13 for this data set.
- Both input and output procedure statements for this dat set.
- Procedure statements which prevent the compiler from recognizing whether either of these applies; for example, if the data set number or SYSACT function number is not an integer constant or if a precompiled procedure is used.

If the source program has already been compiled and linkage edited in a previous job, then the data set on which it has been stored (in load module form) must be concatenated to SYS1.LINKLIB. Data sets containing precompiled procedures called by the source program (see Section 4) must also by concatenated to SYS1.LINKLIB.

If the programmer specifies a TRACE, TRBEG or TREND option in the EXEC statement of the execution job step, the semicolon count (see Section 3) is stored intermediately on a data set with the ddname SYSUT1. The programmer must supply a corresponding DD statement if he uses this option. The semicolon count is converted to external form and transferred to the SYSPRINT data set as soon as the execution ends either by reaching the logical end of the source program or due to an error.

The space required for the semicolon count is:

For the main heading	6 bytes
For each semicolon	2 bytes
For each call of a precompiled procedure	12 bytes
For each physical record on SYSUT1	4 - 6 bytes

System/360 ALGOL permits data to be temporarily stored on and retrieved from external devices without conversion, using the ALGOL I/O procedures PUT and GET. If the programmer uses this facility in his source program, then he must supply a DD statement with the ddname SYSUT2. The device specified by this statement for storing such intermediate data should be a direct access device to guarantee reasonable performance, though programming is performed independently between magnetic tape and direct access devices. All data passed by a single PUT is stored as one record. This record will be as long as the data passed, plus 8 bytes. The maximum record length accepted is 2048 bytes.

The DCB information which may be specified by the user for execution time data sets is blocksize, record format and record length, except for the trace and PUT/GET data sets (ddnames SYSUT1 and SYSUT2) for which only blocksize may be specified (up to a maximum of 2048 bytes).

Where SYSACT8 is used in the ALGOL program and record format is specified in the DD statement, RECFM=FA or RECFM=FBA must be specified. If either one of these formats is specified, SYSACT8 must be used in the ALGOL program.

For information not provided, default values will be inserted by a routine in the ALGOL library. In particular, blocksize is assumed as 2048 bytes for SYSUT1 and SYSUT2 if none is specified.

The record length for the SYSPRINT data set is fixed at 91 bytes.

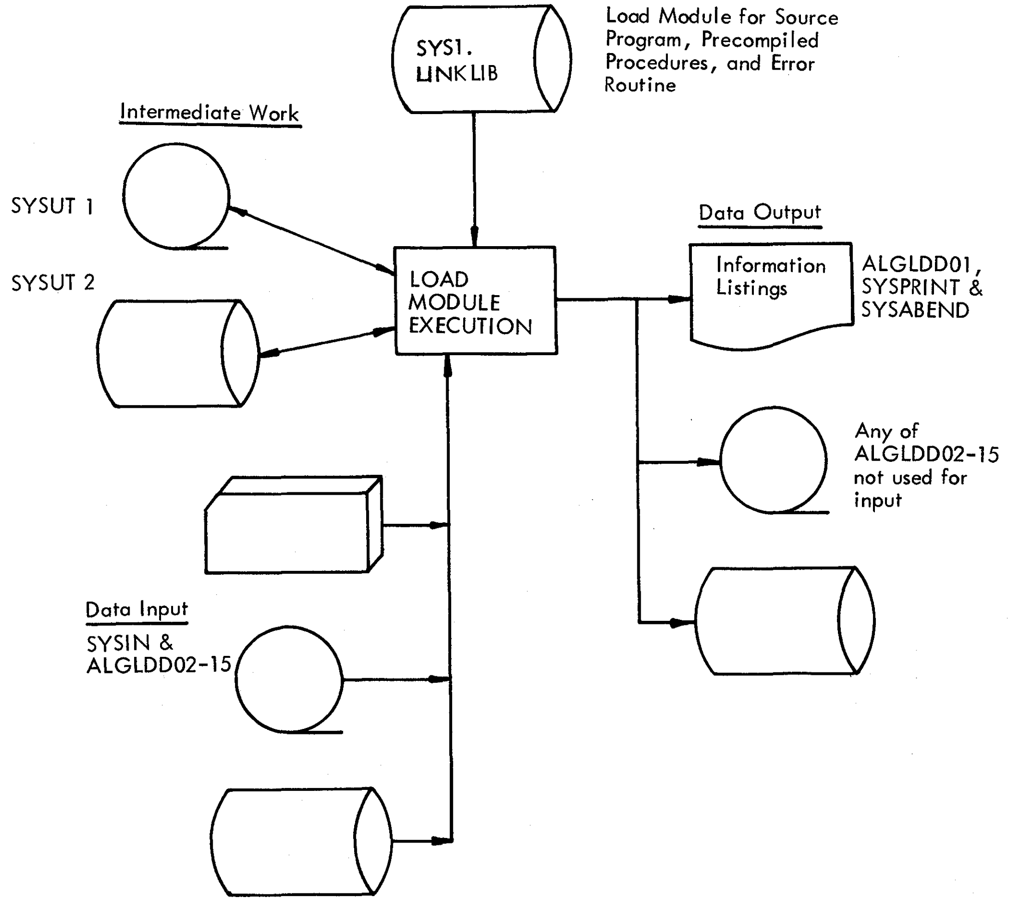

SYS1. LINKLIB
Load Module for Source Program, Precompiled Procedures, and Error Routine
Intermediate Work
SYSUT 1
SYSUT 2
LOAD MODULE EXECUTION
Data Output
Information Listings
ALGLDD01, SYSPRINT & SYSABEND
Any of ALGLDD02-15 not used for input
Data Input
SYSIN & ALGLDD02-15

Figure 10. Flowchart Showing Data Sets Used at Load Module Execution. The data input and output requirements are variable.

Purpose	Standard ddname	Device Used
For data input tc load module	SYSIN	Any input device
For execution time listings	SYSPRINT	Printer[1]
For data output	ALGLDD01	Printer[1]
For data input or output	ALGLDD02 . . . ALGLDD15	1.Direct access or magnetic tape 2.Any
For intermediate storage of semicolon counter when TRACE is specified	SYSUT1	Direct access or magnetic tape
For temporary storage when PUT is specified	SYSUT2	Direct access or magnetic tape

[1] Some form of intermediate storage, such as magnetic tape, may be used to reduce I/O delay for the central processing unit.

Figure 11. Data Sets Used at Execution Time

LOADING

An object module may be loaded and executed in a single job step by use of the loader. The loader can also be used to load and execute a linkage editor processed load module.

Invoking Statement

The loader may be invoked by an EXEC statement of the following form:

```
//stepname  EXEC  PGM=LOADER
```

where 'stepname' is the name assigned to the job step (optional). LCADER specifies the loader program in the installation's operating system.

If the input to the loader is a load module generated from an ALGOL scurce program, the EXEC statement must include the following parameter

```
PARM='EP=IHIFSAIN'
```

IHIFSAIN is the entry point name of a load module generated from an AIGCI source program. Other loader options may be specified in the PARM field. (See 'Appendix F' in this publication and _OS Loader and Linkage Fditor_.)

A method of dynamically invoking the loader within a job step, by means of the CAII, LINK, XCTL and AITACH instructions, is described in Section 4.

Data Sets Used

The data sets used by the loader and by the loaded program or load module (see Figures 12 and 13) must be defined by the programmer with suitable DD statements.

For the following data sets, record lengths are fixed as indicated:

Data Set	Record Length
SYSLIN	80 bytes
SYSLOUT	121 bytes
SYSPRINT	91 bytes

Other information on the data sets used by the loaded program or load module will be found in the preceding section titled 'Load Module Execution'.

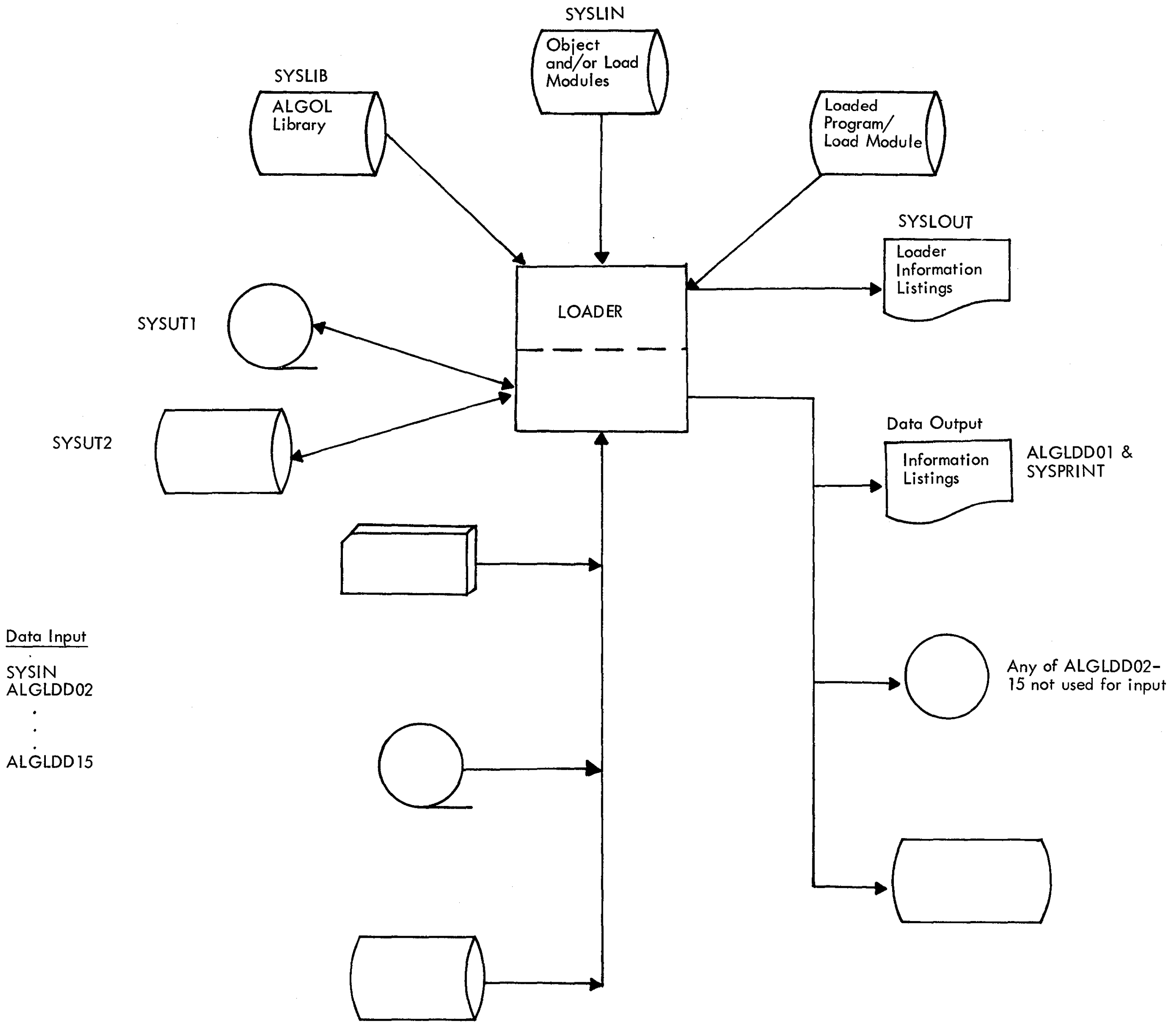


Figure 12. Chart Showing Data Sets Used by the Loader and by a Loaded Program or Load Module in a Load-and-Execute Step

Purpose	Standard ddname	Devices used
<u>Loader</u>		
For object module and/or load module input	SYSLIN	Direct access or magnetic tape
For ALGOL Library, SYS1. ALGLIB. A partitioned data set containing routines in load module form	SYSLIB	Direct access
For loader information listings	SYSLOUT[2]	Printer[1]
<u>Loaded Program/Load Module</u>		
For data input	SYSIN	Any input device
For execution time listings	SYSPRINT	Printer[1]
For data output	ALGLDD01	Printer[1]
For data input or output	ALGLDD02 • • • ALGLDD15	1. Direct access or magnetic tape (see text) 2. Any input/output device
For intermediate storage of semicclon counter when TRACE is specified	SYSUT1	Direct access or magnetic tape
For temporary storage of data by load module (using PUT statement)	SYSUT2	Direct access or magnetic tape

[1] Some form of intermediate storage, such as magnetic tape, may be used to reduce I/O delay for the central proccesing unit.

[2] SYSLOUT must be specified at system generation as an alternative ddname to SYSPRINT for the printer used by the loader.

Figure 13. Data Sets Used by the Loader and by the Loaded Program or Load Module

Section 3: Information Listings

To assist the programmer to find the cause of any faults in the processing cr execution of his program, various forms of information listings are produced for the compilation, linkage editing and execution operations. Some of these listings are optional. Examples are illustrated in Figures 14 to 19.

Control Program Listings

All three operations may produce listings generated by the control program. These are described in OS Messages and Codes. The ABEND macro instruction for specifying the main storage dump is described in OS Supervisor Services and Macro Instructions.

Compilation Listings

A successful compilation of an AIGCL source program produces the following information listings:

- Job control statement information according to which MSGLEVEL option was specified in the JOB statement.
- The source program supplemented by a count of the semicolons occurring in the ·program (cptional).
- A table giving details of all indentifiers used in the program (optional).
- Any warning diagnostic messages.
- Information on main storage requirements at execution time.

If a serious diagnostic message is produced (meaning that object module generation has ended), then the source program and identifier table listings will be printed in full if they have been requested, but the information on main storage requirements will not be printed. If a terminating diagnostic message is produced, then the source program and identifier table listings can be printed only as far as they have been produced.

SOURCE PROGRAM

If the SOURCE option has been specified, the source program is transferred by the compiler to an output data set in order to be listed by a printer. This source program is supplemented by a semicolon count, which is referred to in the diagnostic messages to help localize errors.

The compiler generates this semicolon count when scanning the source program by counting all semicolons occurring in the source program outside strings, except those following the delimiter 'COMMENT'. The value of this semicolon count at the beginning of each record of the source program is printed at the left of that record. It is assigned by the compiler in order to have a clear, problem-oriented reference. Any reference to a particular semicolon number refers to the segment of source program following the specified semicolon; for example, the semicolon number 5 refers to the program segment between the fifth and sixth semicolons.

IDENTIFIER TABLE

If the SOURCE option has been specified, a list of all identifiers declared or specified within the source program is transferred by the compiler to the output data set for printing after the source program listing. This identifier table gives information about the characteristics and internal representation of all identifiers. The identifiers are grouped together within the identifier table according to their scopes.

All blocks and procedure declarations within the source program are numbered according to the order of occurrence of their opening delimiters 'BEGIN' or 'PROCEDURE'. Therefore, if the body of a procedure declaration is a block, then usually this block has the same number as the procedure declaration itself. These numbers are called program block numbers (even if they belong to a procedure declaration and not to a block).

Each line in the table contains entries for up to three identifiers. A line begins with the number of the program block in which the identifiers were declared or specified, the value of the semicolon count at the commencement of the program block, and the number of the immediately surrounding program block. Each identifier entry contains:

1. The external name of the identifier as it appears in the source program. Space for six characters is provided and, if necessary, the identifier is truncated.

2. The type key, as described below.

3. The number of dimensions (for array identifiers), components (for switch identifiers) cr parameters (for procedure identifiers). This position is blank for all other types of identifiers.

4. The displacement for the quantity denoted by the identifier, as explained below.

The <u>type key</u> consists of five characters denoting the type characteristics of the identifier. These characters are as follows (b represents blank):

In first position:	R when real I when integer B when Boolean b when anything else
In second position:	L when label S when switch T when string (text) b when anything else
In third position:	A when array P when procedure b when anything else
In fourth position:	N when formal parameter called by name V when for mal parameter called by value b when declared identifier (not formal parameter)
In fifth position:	C when precompiled (code) procedure b when anything else

Examples of these are:

For a real variable	Rbbbb
For a Boolean array	BbAbb
For a formal parameter specified integer procedure called by name	IbPNb
For a precompiled procedure	bbPbC

The <u>displacement</u> is in hexadecimal form and has the following meaning:

- For all identifiers denoting simple variables, arrays and formal parameters, it is the relative position of their values in the data storage area, as described below.

- For all identifiers denoting labels, procedures and switches (if not specified as formal parameters), it is the relative position of the corresponding entry in the label address table, as described below. This position is known as the label number (LN).

The space allocated to each identifier is as fcllows:

For formal parameters:	8 bytes
For Boolean identifiers:	1 byte
For integer identifiers:	4 bytes

For real identifiers: 4 bytes when SHORT is specified; 8 bytes when LCNG is specified.

For arrays: see 'Storage Mapping Function' below.

At execution time, for each program block, a <u>data storage area</u> (DSA) is created dynamically at each entry of the program block and is released when leaving it. The lengths of the data storage area and the relative positions of all data contained in them are determined by the compiler. These relative positions, together with the program block numbers, uniquely identify the quantities of an ALGOL program. Two forms are used according to whether the SHCRT or LONG option was specified in the invoking statement.

The data storage area of a program block contains locations for:

1. The values of simple variables

2. The storage mapping functions of arrays (see below)

3. In the case of formal parameters, the type characteristics and addresses of the actual parameters

4. Intermediate results, addresses, etc.

A label address table is created by the compiler and transferred to the object module. In general it is used at execution time to load a branch register before any branch is performed. It contains addresses corresponding to:

1. Library modules required

2. Labels

3. Procedure declarations

4. Switch declarations

5. Internal branches ('IF', 'FOR', etc.)

The storage mapping function describes the storage layout of an array. The storage that the storage mapping function requires in the DSA can be calculated from

$s = 4(d + 5) + X$,

where

s = number of bytes in storage mapping function

d = number of dimensions in array

X = 4 if LONG is specified and is an even number, 0 otherwise

DIAGNOSTIC MESSAGES

During the compilation as many programming errors as possible are detected and appropriate diagnostic messages are produced to help the programmer to identify them. Diagnostic messages are caused by:

1. Programming errors. These are detected and reported by the compiler as far as they do not depend on the dynamic flow of the program. Programming errors depending on the dynamic flow of the program are detected and reported by the load module.

2. Violations of capacity limitations. Such violations are detected and reported by the compiler, where possible. Those which cannot be detected at compile time are detected and reported by the load module at execution time.

3. I/O errors caused by malfunction of channels or external devices are reported when they occur.

4. Control card errors not detected by the job scheduler.

5. Program interrupts.

The diagnostic messages are transferred to the output data set to be listed by a printer. 'Appendix F' contains a list of the messages that may be produced by the ALGOL compiler.

STORAGE REQUIREMENTS

Following the diagnostic messages, the compiler transfers information about the execution time storage requirements to the output data set if the compilation finished successfully. This information gives no exact storage estimate of the object module execution because the storage allocation for data is performed dynamically at execution time and depends on the flow of control through the object module and on the amount of data at execution time.

For example, the data storage area belonging to a program block is allocated only as long as that program block is active. In the case of recursive procedures more than one generation of the corresponding data storage area may be required. The storage needed for the array is not contained in a data storage area and depends on the execution time values of the bounds of the array.

Nevertheless, a programmer knowing the structure of his program may gain rough storage estimates from the following information given by the compiler.

1. Main storage required by the object module, including tables and constant pool.

2. A list of the main storage requirements of all data storage areas. This list consists of one entry for each program block, containing the program block number, and the number of bytes required for the corresponding data storage area.

Linkage Editing Listings

A successful linkage editing can produce the following information listings:

- Job control statement information according to which MSGLEVEL option was specified in the JOB statement.

- Disposition data, listing the options specified and the status of the load module in the output library.

- Diagnostic messages (severity code 1).
- A cross reference table of the load module, or alternatively, a module map (both optional).

If a diagnostic message of severity code 2 or 3 is produced, other information listings might not be produced. If a diagnostic message of severity code 4 is produced, other information listings will not be produced.

DIAGNOSTIC MESSAGES

A description of the diagnostic messages that may be produced by the linkage editor is contained in 'Appendix F'.

MODULE MAP

If MAP is specified in the invoking statement for the linkage editor, then a module map is transferred to the output data set to be listed by a printer. The module map shows all control sections (the smallest separately relocatable units of a program) in the load module and all entry names (to routines in the ALGOL library) in each control section. The control sections are arranged in ascending order according to their origins (which are temporary addresses assigned by the linkage editor prior to loading for execution). The entry names are listed below the control section in which they are defined. The origins and lengths (in bytes) of the control sections and the location of the entry names are listed in hexadecimal form. Unnamed control sections are identified by $ in the list.

At the end of the module map is the entry address of the instructions with which processing of the module begins. It is followed by the total length of the module, in bytes. Both values are in hexadecimal form.

CROSS-REFERENCE TABLE

If XREF is specified in the invoking statement for the linkage editor, the cross reference table is transferred to the output data set to be listed by a printer.

The cross reference table consists of a module map and a list of cross references for each control section. In the list of cross references, each address constant that refers to a symbol defined in another control section is listed with its assigned location (in hexadecimal form), the symbol referred to, and the name of the control section in which the symbol is defined.

If a symbol is unresolved after processing by the linkage editor, it is identified by $UNRESOLVED in the list. However, if an unresolved symbol is marked by the never call function, it is identified by $NEVER-CALL.

The entry address and total length are listed after the list of cross references.

Execution Time Listings

A successful execution of the load module produces the following information listings:

- Job control statement information according to which MSGLEVEL option was specified in the JOB statement.
- The ALGOL program trace, which is a list of the semicolon numbers assigned by the compiler (optional).

If an error is detected during execution of the load module, additional information listings are printed before the trace:

- A diagnostic message
- The contents of the data storage areas (optional)

DIAGNOSTIC MESSAGES

Any error detected at execution time causes abnormal termination. A diagnostic message is produced which is transferred to an output data set to be listed by a printer. The diagnostic messages which may be produced during load module execution are listed in 'Appendix F'.

DATA STORAGE AREAS

If DUMP is specified in the invoking statement for the execution operation, the data storage areas (DSA) in main storage are transferred to the output data set to be listed by a printer. They are listed in the reverse order to which they were created.

A DSA is created for each call of a program block (see 'Compilation Listings') and exists in main storage as long as the call is effective. The DSA contains:

1. All execution-time values of variables declared or specified in the program block except for arrays. The array values are stored separately but are included in the listing because they are referenced by the storage mapping function which is contained within the DSA.

2. Intermediate results (known as the object-time stack).

The information listed for each DSA consists of:

- Name of load module

- Program block number

- Description of program block; either BLOCK, PROCEDURE or TYPE PROCEDURE

- The values in the DSA, in batches according to their category, that is, formal parameters, declared identifiers and object-time stack, arrays called by value, and declared arrays.

The values are those which exist at the time the error was detected (in hexadecimal form). The displacement in the DSA of the first value in each line is printed at the beginning of each line. This is a six-digit hexadecimal number.

For formal parameters, each entry has 16 digits, and in the case of parameters called by name the entry contains an address constant pointing indirectly to the value.

For declared indentifiers and the object-time stack, the identifier entries are listed first and they can be located using the identifier table if it was listed by the compiler. The object-time stack contains various intermediate results and addresses which are not directly related to the identifiers in the source program.

For arrays, the length depends on the storage mapping function. The displacement of the storage mapping function in the DSA is given for each array.

In the listings, real values have a length of 8 hexadecimal digits when SHORT is specified and 16 digits when LONG is specified. They are in standard floating-point representation. Integer values have a length of 8 hexadecimal digits and are in standard fixed-point representation.

Boolean values have a length of 2 hexadecimal digits which appear as 00 for 'FALSE' and 01 for 'TRUE'.

An editing routine inserts blanks between each set of 8 digits to improve readability.

ALGOL PROGRAM TRACE

A program trace, listing the semicolon numbers assigned by the compiler (see 'Compilation Listings') in the order the corresponding semicolons were encountered during execution, is transferred to an output data set to be listed by a printer if TRACE, TRBEG or TREND is specified in the invoking statement for the execution. The completeness of the trace depends on the option or options specified (see 'Appendix E'). Only the semicolons actually passed through at execution time are included in the trace.

If a precompiled procedure is used in the program and TRACE is specified, then the semicolon numbers for the procedure are included in the correct position within the program. The appropriate load-module name (first four characters only) is inserted at the beginning of the listings and each time a change occurs in the first four characters of the module name.

Loader and Execution Listings

The information listings printed by a successful loader step may include the following two categories of information:

1. Information specific to the processing of the loaded program by the loader. Depending on the options specified for the job scheduler and the loader, and on the outcome of loader processing, the information may include:

   - A list of the job control statements used to invoke the loader (provided MSGLEVEL=1 is specified).

   - A list of the options specified for and implemented by the loader.

   - A storage map of the loaded program, showing the name and absolute address of every control section and entry point defined in the program. The storage map is printed if the MAP option is specified.

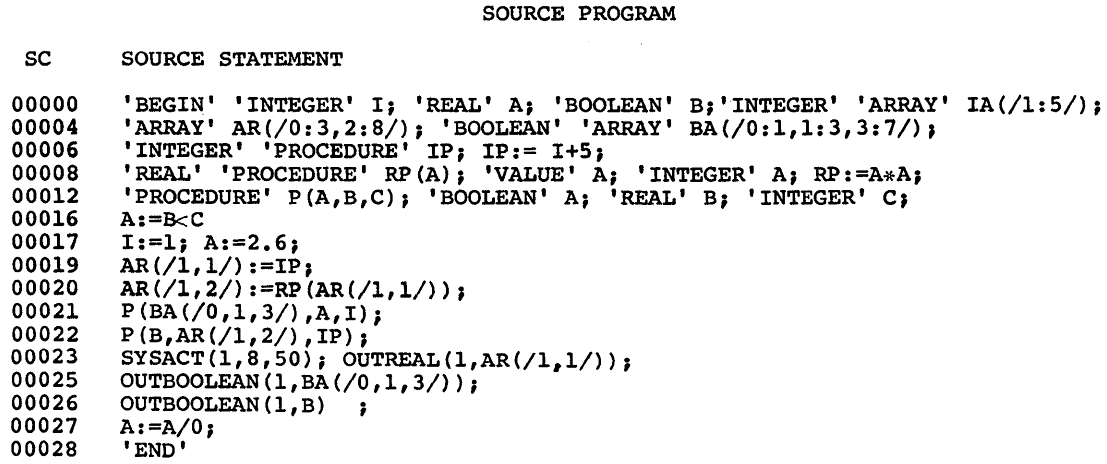
- Diagnostic message, if one or more errors in the loaded program are detected. The error messages generated by the loader are similar to those generated by the linkage editor. A description of the message format is provided in 'Appendix F'.

2. Information relative to the execution of the loaded program. Depending on the execution options specified and on the successful execution of the loaded program, the information printed may include:

   - A diagnostic message in the event a program error (causing the loaded program to be abnormally terminated) is detected. Execution time diagnostic messages are listed in 'Appendix F'.

   - Listings of the contents of all existing data storage areas in main storage at the time of an execution time error, provided the DUMP option is specified. The data storage area is described above under 'Execution Time Listings'.

   - Program trace information, as described under 'Execution Time Listings', provided one of the options TRACE, TRBEG or TREND is specified for the loaded program.



```
                              SOURCE PROGRAM

 SC     SOURCE STATEMENT

00000   'BEGIN' 'INTEGER' I; 'REAL' A; 'BOOLEAN' B;'INTEGER' 'ARRAY' IA(/1:5/);
00004   'ARRAY' AR(/0:3,2:8/); 'BOOLEAN' 'ARRAY' BA(/0:1,1:3,3:7/);
00006   'INTEGER' 'PROCEDURE' IP; IP:= I+5;
00008   'REAL' 'PROCEDURE' RP(A); 'VALUE' A; 'INTEGER' A; RP:=A*A;
00012   'PROCEDURE' P(A,B,C); 'BOOLEAN' A; 'REAL' B; 'INTEGER' C;
00016   A:=B<C
00017   I:=1; A:=2.6;
00019   AR(/1,1/):=IP;
00020   AR(/1,2/):=RP(AR(/1,1/));
00021   P(BA(/0,1,3/),A,I);
00022   P(B,AR(/1,2/),IP);
00023   SYSACT(1,8,50); OUTREAL(1,AR(/1,1/));
00025   OUTBOOLEAN(1,BA(/0,1,3/));
00026   OUTBOOLEAN(1,B)  ;
00027   A:=A/0;
00028   'END'
```

Figure 14. Example of Source Program Listing

```
                                  IDENTIFIER TABLE

PBN SC     PBN      NAME     TYPE  DM DSP      NAME     TYPE   DM DSP      NAME   TYPE  DM DSP
           SURR                    PR LN                       PR LN                    PR LN


001 00000 000      A        R         01C      AR       R A    02 03C      B      B        020
                   BA       B A    03 058      I        I         018      IA     I A   01 024
                   IP       I P    00 070      P          P    03 078      RP     R P   01 074

002 00006 001      IP       I P    00 070

003 00008 001      A        I  V      020      RP       R P    01 074

004 00012 001      A        B  N      018      B        R N       020      C      I  N     028
```

Figure 15. Example of Identifier Table Listing.
This corresponds to the program in Figure 14.

```
                         STORAGE REQUIREMENTS (DECIMAL)

OBJECT MODULE SIZE 1840 BYTES.

DATA STORAGE AREA SIZES

  PBN   BYTES       PBN   BYTES       PBN   BYTES        PBN   BYTES        PBN   BYTES

  001    136        002      32       003     40         004     60
```

Figure 16. Example of Storage Requirements Listing.
This corresponds to the program in Figure 14.

```
                           ----  CROSS REFERENCE TABLE  ----

CONTROL SECTION                ENTRY

  NAME      ORIGIN  LENGTH       NAME     LOCATION     NAME     LOCATION     NAME     LOCATION     NAME     LOCATION

                00     730
                               IHIDSTAB      6D8     IHIENTIF      724
IHISYSCT*      730     5EC
IHISOREA*      D20     328
                               IHISORAR      D20     IHISOREL      D30



IHIIORTN*     2580     B58

                               IHIIOROQ     2580     IHIIOROP     25AC     IHIIORNX     28C4     IHIIORCL     2BOC
                               IHIIORCP     2C72     IHIIORGP     2D38     IHIIORCN     2D3C     IHIIOREN     2D76
                               IHIIOREV     2DCE     IHIIORED     2E40     IHIIORCI     2F44     IHIIORER     2FCC

LOCATION  REFERS TO SYMBOL  IN CONTROL SECTION

   61C           IHISYSCT        IHISYSCT
   658           IHISOREL        IHISOREA
   660           IHIOBOOL        IHIOBOOL
   D08           IHIIORCL        IHIIORTN



  1F48           IHIFSARB        IHIFSARB
  1F5C           IHIIORCP        IHIIORTN
  1F81           IHIFSARA        IHIFSARA
ENTRY ADDRESS      IF24
TOTAL LENGTH       30D8
```

Figure 17. Example of Cross-Reference Table Listing.
This is part of the table produced from the program in Figure 14. A Module Map Listing would contain only the list of Control Sections and Entry Names, plus the Entry Address and Total Length Information. Control Sections marked with an asterisk were included from a library during automatic library call.

```
IHI031I  SC=00027  PSW= FF05000F 48005E22    DIVISION BY ZERO, FLOATING POINT

MODULE = GO          PROGRAM BLOCK NUMBER = 001      (BLOCK)

         DECLARED IDENTIFIERS AND OBJECT TIME STACK
000018   00000001  4129999A  0001FF2C  01000000   0001E49C   0001F4A0   0001E4B4   00000014
000038   00000004  02000024  0001E428  0001E430   0001E4A0   00000070   0000001C   00000004
000058   0300003C  0001E408  0001E410  0001E42E   0000001E   0000000F   00000005   00000001
000078   0001E44C  0000581C  0001F560  400058C

         SMF DISPLACEMENT IN DSA =  000058        DECLARED ARRAY
000000   00000000  00000000  00000000  00000000   00000000   00000000   00000000   00000000

         SMF DISPLACEMENT IN DSA =  00003C        DECLARED ARRAY
000000   00000000  00000000  00000000  00000000   00000000   00000000   41600000   42240000
000020   00000000  00000000  00000000  00000000   00000000   00000000   00000000   00000000
000040   00000000  00000000  00000000  00000000   00000000   00000000   00000000   00000000
000060   00000000  00000000  00000000  00000000

         SMF DISPLACEMENT IN DSA =  000024        DECLARED ARRAY
000000   00000000  00000000  00000000  00000000   00000000
```

Figure 18. Example of Error Message and Data Storage Area Listing.
This is the listing produced from the program in Figure 14 when the division by zero was encountered.

```
  ALGOL PROGRAM TRACE

MODULE     SEMICOLON NUMBERS

  GO       00001 00002 00003 00004 00005 00006 00008 00012 00017 00018 00019 00007 00020
           00009 00010 00011 00021 00013 00014 00015 00016 00022 00013 00014 00015 00016
           00007 00023 00024 00025 00026 00027
END OF ALGOL PROGRAM EXECUTION
```

Figure 19. Example of Program Trace Listing.
This was produced from the program in Figure 14.

Section 4: Programming Considerations

Capacity Limitations

In addition to those given in OS_ALGCL_Language, the following restrictions must be observed when writing an ALGOL source program:

Number of blocks and procedure declarations (NPB)	≤255
Number of for statements	≤255
Number of indentifiers declared or specified in one block or procedure (F is at most twice the number of for statements occurring in that block.)	≤179-F for type procedures ≤180-F otherwise
Length of letter string serving parameter delimiter	≤1024 letters when the main storage available is <50K ≤2000 letters otherwise
Length of label indentifier	≤1024 characters when the main storage available is <50K ≤2000 letters otherwise
Length of source program	≤255K
Number of semicolons in the whole program	≤65535
Number of nested blocks, compound statements, fcr statements and procedure declarations	≤999
Number of labels declared or additionally generated by the compiler	≤1024
The compiler generates the following additional labels:	
For each switch declaration	2
For each procedure declaration	2
For each procedure activation (including function designators)	1
Fcr each 'THEN' and 'ELSE'	1
For each for statement	at most L+3 where L is the number of for list elements
Length of constant pool	≤(256-NPB) x 4096 bytes

The requirements of components within the pool are:

Integer constant	4 bytes
Real constant (SHORT)	4 bytes
Real constant (LONG)	8 bytes
String (in bytes)	2+number of symbols of open string between the cutermost string quotes

The constant pool is divided into blocks of 4096 bytes each. The first block contains the integer constants 0 to 15 (64 bytes). All strings together are restricted to fill not more than the rest of this block (4096 - 64 - 2S bytes, where S = number of strings).

No constant occurring more than once in the source program is stored twice in the same block; however, it may possibly be stored more than once in different blocks. Up to seven bytes may be left unused.

Length of data storage area for each block or procedure declaration	≤4096 bytes
Number of blank spaces serving as delimiters on I/O data sets	≤255
Number of records in a data set	≤32760
Number of records per section	≤255
Number of entries in the Note Table (see below)	≤127
Identification number (N) used by PUT or GET	0≤N≤65535

(The Note Table stores information to retrieve records which may be required again later. An entry for a record is made each time the ALGOL I/O procedures PUT and SYSACT13 are executed, and each time an input operation, with backward repositioning, follows an output operation on the same data set.)

Invoking a Program Within a Job Step

Any one of the four macro-instructions, CALL, LINK, XCTL, or ATTACH, may be used to dynamically invoke the compiler, linkage editor, loader or a load module within a job step. This is an alternative to the more usual method of invoking a program by starting a job step with an EXEC statement. Full details of the four macro instructions are given in <u>OS Supervisor Services and Macro Instructions</u>.

To invoke a program with the CALL macro instruction, the program must first be loaded into main storage, using the LOAD macro instruction. This returns, in general register 15, the entry address which is used by the CALL macro instruction. The instructions used could be:

```
LOAD   EP=member name

LR     15,0

CALL   (15), (option address), VL
```

To invoke a program with one of the LINK, XCTL or ATTACH macro instructions would require:

```
LINK   EP=member name,

       PARAM=(option address), VL=1
or
XCTL   EP=member name
or
ATTACH EP=member name,

       PARAM=(option address), VL=1
```

'member name' specifies the name of the member of a partitioned data set which contains the program required.

For the compiler, member name=ALGCL

For the linkage editor, member name=IEWL

For the loader, member name=LOADER

For the load module, member name is specified by the programmer in the SYSLMOD DD statement for the linkage editor.

'option address' specifies the address of a list containing the options required by the user. Where the program invoked is the loader (member name=LOADER) and the input to the loader consists solely of one or more linkage-editor-produced load modules, the option list must include the parameter EP=IHIFSAIN. The list must begin on a half-word boundary. The first two bytes contain a number giving the number of bytes in the remainder of the list. (If no options are specified this number must be zero). The list itself contains any of the options available to the PARM parameter in an EXEC statement (see 'Appendix E').

When using CALL, LINK or ATTACH to invoke the compiler, other ddnames may be used in place of the standard ddnames given in Section 2 for the data sets and an alternative page number (instead of the normal 001) may be specified for the start of output listings.

If alternative ddnames are used, then in the statement invoking the compiler 'option address' must be followed by 'ddname address' giving the address of a list containing the alternative ddnames. If alternative page numbers are used, 'page address' giving the address of a location containing the alternative page number must be placed after 'ddname address'; though if alternative ddnames are not required, 'ddname address' may be replaced by a comma.

The ddname list must begin on a half-word boundary. The first two bytes contain a number giving the number of bytes in the remainder of the list. The list itself contains up to ten 8-byte fields, separated by commas, for specifying alternative ddnames for the data sets. As only seven data sets are used by the compiler, three of the fields are left blank. The alternative ddnames must be listed in the following order:

Purpose_of_Data_Set	Standard_ddname
Output of object module for linkage editor or loader	SYSLIN
--Three blank fields--	
Source program input	SYSIN
Information listings	SYSPRINT
Output of object module for card deck	SYSPUNCH
Intermediate work	SYSUT1
Intermediate work	SYSUT2
Intermediate work	SYSUT3

The field for a data set which does not use an alternative ddname must be left blank if there is an alternative ddname following. Otherwise the field is omitted.

The location containing the page number must begin on a half word boundary. The first two bytes contain a number giving the number of bytes in the remainder of the location (namely, four). These four bytes contain the number for the first page of the output listings, and on return to the invoking program they will contain the number of the last page.

An example of an invoking statement and the associated lists, for the compiler, is:

```
COMPILE LINK EP=ALGOL,PARAM=
             (OPTIONS,DDNAMES,PAGE),
             VL=1

OPTIONS DC   H'25',C'PROCEDURE,DECK,
             SIZE=90112'

DDNAMES DC   H'35',C'OUTPUTbb,3CL8'b',
             C'INPUTbbb',3CL8'b',
             C'CARDDECK'

PAGE    DC   H'04',F'62'
         b = BLANK
```

In this case, the PROCEDURE and DECK options are specified and 88K bytes of main storage are made available. Alternative ddnames are specified for SYSLIN, SYSIN and SYSPUNCH, and 62 is specified as the first page number for the output listings.

Precompiled Procedures

An ALGOL program may invoke one or more subprograms written in the ALGOL language or in the Assembler language and stored on a partitioned data set in load module form. Subprograms of this type are known as precompiled procedures.

A precompiled procdure to be invoked by an ALGOL program must be nominally declared in the calling program. The declaration consists of a normal procedure heading, followed by the delimiter 'CODE' representing the procedure body. The name of the precompiled procedure declared in the calling program must be the load module name of the precompiled procedure.

A precompiled procedure is loaded into main storage when control passes to the program block in which the precompiled procedure is declared, and is deleted when control leaves that block. Where possible, a precompiled procedure should be nominally declared in the outermost block of the calling ALGOL program. The declaration of a precompiled procedure in another precompiled procedure which is frequently invoked, should be avoided. This saves execution time by reducing the number of loadings of the precompiled procedure.

The precision of real values must be the same, SHORT or LONG, in the callng ALGOL program and the precompiled procedure. If the installation allows multiprogramming, the REUS option ('Appendix E') may not be specified for the precompiled procedure load module, in the statement invoking the linkage editor.

ALGOL LANGUAGE PROCEDURES

A precompiled procedure written in the ALGOL language must satisfy the rules, as | stated in OS ALGOL Language, governing any normal procedure declaration. That is to say, the source module should comprise a procedure heading and a procedure body. The source module should not be enclosed by the delimiters 'BEGIN' and 'END'.

An ALGOL procedure to be invoked in a later program must be compiled, linkage edited and stored on a partitioned data set. In the invoking statement, the source module must be identified as a precompiled procedure by specifying the opticn PROCEDURE.

An example of the job control statements needed to compile and linkage edit a precompiled procedure is provided in Figure 36. Figure 37 illustrates the jcb control statements needed to compile, linkage edit and execute an ALGOL program in which a precompiled procedure is called.

ASSEMBLER LANGUAGE PROCEDURES

A sample Assembler language procedure, and an ALGOL program in which the procedure is nominally declared and called, are shown in Figure 21. Figure 37 contains an example of the job control statements needed to compile, linkage edit and execute an ALGCL program in which a precompiled procedure is called.

In writing an Assembler language procedure, certain rules must be observed. These rules are outlined below under the headings Entry and Start, Definitions, Register Use, Parameter Handling, and Termination.

In the instructions given below the programmer may specify any valid names in the name fields, provided the appropriate name is used in all references.

Entry and Start

The entry point of the module must be defined as follows (the names shown are examples only):

```
ENTRY  DC  A(PBTAB,0,PARMDEF)
```

where 'ENTRY' is the location specified in the END statements; 'PBTAB' references a Program Block Table (see 'Definitions', item 1); 0 represents a dummy label; and PARMDEF references a list of two-byte parameter definition constants or characteristics (Figure 20), as follows:

```
PARMDEF DC  XL2'characteristic 1'
        DC  XL2'characteristic 2'
        .
        .
        .
        DC  XL2'characteristic n'
        (First instruction executed)
```

The list must include a characteristic for each formal parameter and must be followed by the first instruction to be executed in the module. If the procedure has no parameters, PARMDEF must reference the initial instruction.

Definitions

The following data must be defined in the Assembler language procedure.

1. A 16-byte table, called the Program Block Table, must be defined:

```
PBTAB  DS  F
       DC  CL4'(proc. name)'
       DS  F
       DC  H'(DSA length)'
       DC  X'04'['08' if type-
             procedure]
       DC  X'0p' p=no. of formal
             parameters
```

   'proc. name' represents the first four characters of the module name. 'DSA length' represents the length of the procedure's data storage area. The length is 24 (+8 if the procedure is type-qualified), +8 x number of fcrmal parameters. The Program Block Table must be addressed by an address constant at the procedure entry point (see 'Entry and Start') and should preferably be defined at the base address of the procedure (see 'Register Use', item 4).

2. Certain registers used in communicating with Fixed Storage Area

routines must be symbolically named (see 'Register Use', item 1).

3. The following symbolic displacement values must be defined for those Fixed Storage Area routines which are invoked in the procedure:

```
CAP1      EQU  X'0D4'
CAP2      EQU  X'0D8'
PROLOGFP  EQU  X'0DC'
RETPROG   EQU  X'0E4'
EPILOGP   EQU  X'0E8'
CSWE1     EQU  X'0F4'
VALUCALL  EQU  X'118'
```

See 'Parameter Handling' and 'Termination'.

4. A list of parameter definition constants, identifying the character of the formal parameters, if any, must be defined. See 'Entry and Start' and Figure 20.

5. An address constant containing the address of the Program Block Table (item 1 above) and a parameter definition list, must be defined at the load module entry point.

Register Use

The standard IBM linkage conventions are not implemented in any code generated by the compiler involving a transfer of control between an ALGOL load module and a submodule. For this reason, provision must be made in a submodule to insure that externally used registers to be used internally are, at entry, saved in a local save area (and reloaded before exit), and that, where necessary, internally used registers are saved in advance of every parameter call.

All general-purpose and floating-point registers may be freely used in an Assembler language procedure, subject to the restrictions itemized below.

1. In the code sequences for calling actual parameters (see 'Parameter Handling'), registers 8, 10, 11, 13, 14 and 15 are symbolically referenced. Every register so referenced in a calling sequence within the precompiled procedure must be defined as follows:

```
ADR      EQU   8
CDSA     EQU  10
PBT      EQU  11
FSA      EQU  13
STH      EQU  14
BRR      EQU  15
```

2. During every call for an actual parameter and before final exit from the precompiled procedure, registers CDSA (10), PBT (11) and FSA (13) must contain their values at entry to the procedure. At entry, CDSA addresses the Assembler language procedure's data storage area; PBT addresses the Program Block Table (see 'Definitions', item 1); and FSA addresses the Fixed Storage Area. If any of these registers are used internally, other than in actual parameter calls, their contents must be saved in a local save area at entry to the procedure, and must be reloaded before all parameter calls and before final exit.

3. Before every call for an actual parameter, the contents of all internally used registers required after the parameter call should be saved in a local save area and reloaded on return.

4. All registers except register 10, 11 and 13 are subject to varying use during a parameter call. The programmer is advised to use register 11 as base register and to specify the Program Block Table ('Definitions', item 1) in the USING statement, as illustrated in Figure 21. This insures that the base register is always correctly loaded before return to the procedure.

Parameter Handling

A call for an actual parameter must be implemented by means of an appropriate calling sequence, which depends on the character of the parameter and on whether it is called by name or by value.

In the instructions given below, the notation 'displ' represents the displacement of a field reserved for the formal parameter in the precompiled procedure's data storage area. The displacement of the storage field of the nth formal parameter is 24 + 8(n-1), except in the case of a type procedure, where it is 32 + 8(n-1).

Important Note: Before every call for an actual parameter, all locally used registers should be saved and registers CDSA, PBT and FSA should contain their original values at entry to the precompiled procedure (see 'Register Use'). On return from a parameter call, locally used registers should be reloaded.

Call by Name

1. Formal parameter specified 'ARRAY', 'STRING' or type 'REAL', 'INTEGER' or 'BOOLEAN':

```
BAL  BRR,CAP1(FSA)
DC   H'8'
DS   H
L    ADR,displ(CDSA)
```

   On return, register ADR addresses the actual parameter value or string or the actual array's storage mapping function. The storage mapping function describes the storage layout of the array. Bytes 8 to 11 contain the address of the first element in the array. The array elements are arranged in ascending order, a given subscript being regarded as a unit of the subscript position immediately to the left. For example, if an array is declared A(/1:2,1:2), the elements are arranged as follows:

   A(/1,1/), A(/1,2/), A(/2,1/), A(/2,2/)

2. Formal parameter specified 'LABEL':

```
BAL  BRR,CAP1(FSA)
DC   H'8'
DS   H
L    ADR,displ(CDSA)
B    RETPROG(FSA)
```

3. Formal parameter specified 'SWITCH':

```
BAL    BRR,CAP1(FSA)
DC     H'8'
DS     H
L      ADR,displ(CDSA)
LA     BRR, i[i=component number]
BAL    STH, CSWE1(FSA)
B      RETPROG(FSA)
```

The sequence causes an unconditional branch to the labelled statement in the calling ALGOL program.

4. Formal parameter specified 'PROCEDURE' or '<type>' 'PROCEDURE' with j formal parameters:

```
BAL  BRR,CAP1(FSA)
DC   H'8'
DS   H
L    ADR,displ(CDSA)
BAL  BRR,PROLOGFP(FSA)
DC   A(CODESEQ1)
DC   XL2'characteristic 1'
DC   H'j'
DC   A(CODESEQ2)
DC   XL2'characteristic 2'
DS   H
      .
      .
      .
DC   A(CODESEQj)
DC   XL2'characteristic j'
DS   H
```

   'Characteristic 1' represents the two-byte constant (Figure 20) which identifies the character of the first actual parameter.

   'CODESEQ1' represents the symbolic address of an actual parameter code sequence corresponding to the first parameter, as follows:

```
CODESEQ1 LA   ADR,paramaddr1
         B    CAP2(FSA)
```

   where 'paramaddr1' represents the address of the actual parameter. (If the parameter is a string, the first two bytes of the actual parameter should contain the string length +2.) A similar code sequence must be included in the procedure for each of the j parameters of the procedure, and each code sequence must be addressed by an address constant, as shown above.

   Execution of the calling sequence causes an actual procedure to be called.

Call by Value

Formal parameter specified 'ARRAY' or type 'REAL, 'INTEGER' or 'BOOLEAN':

```
BAL  BRR,CAP1(FSA)
DC   H'8'
DS   H
L    ADR,displ(CDSA)
BAL  BRR,VALUCALL(FSA)
DC   H'displ'
DC   CL2'characteristic'
```

'displ' represents the displacement of the formal parameter's storage field in the data storage area; 'characteristic' represents the two-byte characteristic (Figure 20) of the formal parameter.

In the case of a type specification, the calling sequence causes the value of the actual parameter to be moved into the 8-byte field of the formal parameter. In the case of an array, the address of the array's storage mapping function is stored in the first four bytes of the formal parameter's storage field. Bytes 8 to 11 of the storage mapping function contain the address of the first element of the array.

Termination

At the close of a precompiled prccedure, the following must be observed.

1. Registers CDSA, PBT and FSA must, where necessary, be reloaded with their original contents at entry to the precompiled procedure.
2. If the precompiled procedure is type-qualified, the value of the procedure must be stored at displacement 24 in the data storage area. The latter is addressed by CDSA.
3. The terminal instruction must be

```
B   EPILOGP (FSA)
```

This returns control to the calling ALGOL program.

Type of Parameter	Characteristic Halfword (in hexadeicmal form)		Result after call of actual parameter
	When called by name	When called by value	
STRING	CB10		ADR contains address of string
REAL	C212		AdR contains address of real value
REAL		C222	DISP in CDSA contains real value
INTEGER	C211		ADR contains address of integer value
INTEGER		C221	DISPL in CDSA contains integer value
BOOLEAN	C213		ADR contains address of Boolean value
BOOLEAN		C223	DISPL in CDSA contains Boolean value
ARRAY or REAL	CA16		ADR contains address of storage mapping function (see below)
ARRAY		CA26	DISPL in CDSA contains address of storage mapping function
INTEGER ARRAY	CA15		ADR contains address of storage mapping function
INTEGER ARRAY		CA25	DISPL in CDSA contains address of storage mapping function
BOOLEAN ARRAY	CA17		ADR contains address of storage mapping function
BOOLEAN ARRAY		CA27	DISPL in CDSA contains address of storage mapping function
LABEL	CA18		ADR contains address of label
LABEL		CA28	ADR contains address of label
SWITCH	CA1C		ADR contains address of switch
PROCEDURE	CAD0		If the actual procedure is parameter-less then procedure is called, otherwise ADR contains address of procedure
REAL PROCEDURE	CAD2		If the actual procedure is parameter-less then procedure is called, and ADR contains address of real value, otherwise ADR contains address of procedure
REAL PROCEDURE		C2E2	DISPL in CDSA contains real value
INTEGER PROCEDURE	CAD1		If the actual procedure is parameter-less then procedure is called, and ADR contains address of integer value, otherwise ADR contains address of procedure
INTEGER PROCEDURE		C2E1	DISPL in CDSA contains integer value
BOOLEAN PROCEDURE	CAD3		If the actual procedure is parameter-less then procedure is called, and ADR contains address of Boolean value, otherwise ADR contains address of procedure
BOOLEAN PROCEDURE		C2E3	DISPL in CDSA contains Boolean value

Figure 20. Table of Parameter Characteristics for an Assembler Language Precompiled Procedure.
The storage mapping function describes the storage layout of an array. Byte 0 contains a value denoting the number of subscripts in the array. Bytes 8 to 11 contain the address of the first element in the array. Bytes 16 to 19 contain a value denoting the size of the array.

```
         START
*
ADR      EQU    8
CDSA     EQU    10                        MANDATORY
PBT      EQU    11                          REGISTER
FSA      EQU    13                            DEFINITIONS
BRR      EQU    15
*
REGV1    EQU    CDSA                      LOCAL
REGADV1  EQU    FSA                         REGISTER
REGV2    EQU    12                            DEFINITIONS (OPTIONAL)
*
CAP1     EQU    X'0D4'                    MANDATORY
VALUCALL EQU    X'118'                      FIXED
EPILOGP  EQU    X'0E8'                        STORAGE AREA
RETPROG  EQU    X'0E4'                          DEFINITIONS
*
         USING PBTAB,PBT
PBTAB    DS     F
         DC     CL4'COMP'                 PROGRAM
         DS     F                           BLOCK
         DC     H'48'                         TABLE
         DC     X'04C3'
*
ENTRY    DC     A(PBTAB,C,PARMDEF)
*
ALSAVE   DS     2F                        SAVE AREA FOR CDSA AND FSA
USSAVE   DS     15F                         AND FOR LOCAL REGISTERS
ONE      DC     H'1'                      CONSTANT
PARMDEF  DS     0H
         DC     XL2'C211'                 CHARACTERISTIC OF V1
         DC     XL2'C221'                                   V2
         DC     XL2'CA18'                                   L
*
         ST     CDSA,ALSAVE               SAVE CDSA
         ST     FSA,ALSAVE+4                AND FSA
         BAL    BRR,CAP1(FSA)             CALL
         DC     H'8'                        V1
         DS     H                             BY
         L      ADR,24(CDSA)                    NAME
         LR     REGADV1,ADR
         L      REGV1,0(ADR)              LOAD V1
         STM    12,10,USSAVE              SAVE LOCAL REGISTERS
         L      CDSA,ALSAVE               RELCAD CDSA
         L      FSA,ALSAVE+4                AND FSA
         BAL    BRR,CAP1(FSA)             CALL
         DC     H'8'                        V2
         DS     H                             BY
         L      ADR,32(CDSA)                    VALUE
         BAL    BRR,VALUCALL(FSA)         V2 IS CONVERTED TO INTEGER AND
*                                           STORED IN DSA
         DC     H'32'
         DC     XL2'C221'
         MVC    USSAVE(4),32(CDSA)        MOVE  V2 TO SAVE AREA
         LM     12,1C,USSAVE              RELOAD LOCAL REGISTERS
*                                         REGV2 CONTAINS V2
         CR     REGV1,REGV2               COMPARE V1 TO V2
         BH     LEXIT                     V1 > V2
         AH     REGV1,ONE                 V1 ¬> V2: ADD 1 TO V1
         ST     REGV1,0(REGADV1)          STORE V1
*
         L      CDSA,ALSAVE               RELOAD CDSA
         L      FSA,ALSAVE+4                AND FSA
         B      EPILOGP(FSA)              RETURN TO CALLING PROGRAM
*
LEXIT    EQU    *
         L      CDSA,ALSAVE               RELOAD CDSA
         L      FSA,ALSAVE+4                AND FSA
         BAL    BRR,CAP1(FSA)             CALL
         DC     H'8'                        L
         DS     H                             BY
         L      ADR,4C(CDSA)                    NAME
         B      RETPROG(FSA)              RETURN TO CALLING PROGRAM
*
         FND    ENTRY
```

Figure 21. An Assembler Language Procedure. The procedure is declared under the name COMP (in the ALGOL program shown above) with the formal parameters V1, V2 and L. V1 and V2 are integers, while L is a label. COMP is called by the ALGOL program and compares V1 to V2. If V1≤V2, the constant 1 is added to V1, and control is returned to the next instruction in the calling program. If V1>V2, control is returned to the calling program at the address specified for label L.

```
'BEGIN'
      'INTEGER' I;
      'PROCEDURE' COMP(V1,V2,L); 'VALUE' V2; 'INTEGER' V1,V2; 'LABEL' L;
      'CODE'
      'COMMENT' THIS NOMINALLY DECLARES THE ASSEMBLER PROCEDURE COMP;
      ININTEGER (0,I);
CONT: COMP(1,200.5,OUT);
      'GOTO' CONT;
OUT:
'END'
```

Figure 22. An Invoking ALGOL Program.
The ALGOL program shown above reads a number from Data Set Number 0, assigns the number to the variable I, and invokes the Assembler language procedure COMP. The call to COMP includes three actual parameters: the variable I, the constant 200.5, and the label OUT. COMP compares I 201 (200.5 converted to integer). If I ≤ 201, COMP adds 1 to I and returns control to the next statement in the ALGCL program. COMP is then called again. The call is repeated until I > 201, at which time COMP passes control to the statement labelled OUT.

Appendix A: ALGOL Library Routines

When processing the source program, the compiler detects and specifies any routines that need to be combined with the generated object module before it can be executed. These routines are contained in the System/360 Operating System ALGOI library - a partitioned data set with the external name SYS1.ALGLIB. The routines are in load module form and the linkage editor combines them with the object module to produce an executable load module. There are three types of routines - fixed storage area routines, mathematical routines and input/output routines. Additionally, an error routine, stored on the operating system link library, SYS1.LINKLIB, is called at execution time if an error occurs.

Initialization and termination of the library routines is performed using the standard method (see 'Supervisor' in Section 1).

Fixed Storage Area

General routines required to some degree by all object modules are combined into a single load module known as the fixed storage area (IHIFSA). These routines are used to initialize and terminate execution of the ALGOL program, to handle the DSA when entering or leaving a program block or procedure, to produce the program trace, to load precompiled procedures, to get main storage for arrays, to convert values from real to integer and integer to real, to call actual parameters, to handle branches in the program, to handle program interrupts, etc.

Mathematical Routines

Standard mathematical functions contained in ALGOL have corresponding mathematical routines in the library, except for ABS, SIGN and LENGTH which are handled by the compiler, and ENTIER which is contained in the fixed storage area. Routines exist in each case for both long and short precision of real numbers.

These mathematical routines are taken from the System/360 Operating System FORTRAN IV library and modified to conform to the ALGOL language requirements without affecting the mathematical methods used. Full details of these routines are contained in OS_FORTRAN_IV_Library.

Input/Output Routines

Data transfer between the load module and external data sets is performed by input/output routines. These routines correspond to the ALGOL I/O procedures and are mostly contained on separate load modules (see Figure 23). In addition, there is a single load module, IHIIOR, which contains a number of commonly used subroutines.

Error Routine

If an error is detected during execution of the load module, an error routine (in SYS1.LINKLIB) is invoked. Its main purpose is to construct the error message and produce the data storage area listing before passing to the termination routine in the FSA. If a second error occurs while the first is being handled (due, for example, to an I/O error or because the object module has overwritten part of the ALGOL library or control program), then termination takes place immediately and incomplete information listings may be produced.

Module Name		When Used	Storage Estimate (bytes)
ALGOL	FORTRAN IV		
IHIERR		When an error is detected at execution time	4270
IHIFDD	IHCFDXPD	For an exponentiation (** or 'POWER') using long precision base and long precision exponent	200
IHIFDI	IHCFDXPI	For an exponentiation (** or 'POWER') using long precision base and integer exponent	140
IHIFII	IHCFIXPI	For an exponentiation (** or 'POWER') using integer base and integer exponent	170
IHIFRI	IHCFRXPI	For an exponentiation (** or 'POWER') using integer base and integer exponent	140
IHIFRR	IHCFRXPR	For an exponentiation (** or POWER) using short precision base and short precision exponent	200
IHIFSA		For every object (except those for precompiled procedures)	5030
IHIGPR		For either GET or PUT	2420
IHIIAR		For INARRAY or INTARRAY	120
IHIIBA		For INBARRAY	70
IHIIBO		For INBOOLEAN	530
IHIIDE		For either INREAL or ININTEGER	1560
IHIIOR		For every object module	2910
IHIISY		For INSYMBOL	270
IHILAT	IHCLATAN	For a long precision arctangent operation (ARCTAN)	320
IHILEX	IHCLEXP	For a long precision exponential operation (EXP)	450
IHILLO	IHCLLOG	For a long precision logarithmic operation (LN)	310
IHILOR		For a long precision OUTREAL operation	730
IHILSC	IHCLSCN	For a long precision sine or cosine operation (SIN or COS)	370
IHILSQ	IHCLSQRT	For a long precision square root operation (SQRT)	140
IHIOAR		FOR OUTARRAY	120
IHIOBA		For OUTBARRAY	70
IHIOBO		For OUTBOOLEAN	400
IHIOIN		For OUTINTEGER	410
IHIOST		For OUTSTRING	300
IHIOSY		For OUTSYMBOL	290
IHIOTA		For OUTARRAY	120
IHIPTT		For a long precision INREAL or OUTREAL operation	270
IHISAT	IHCSATAN	For a short precision arctangent operation (ARCTAN)	200
IHISEX	IHCSEXP	For a short precision exponential operation (EXP)	280
IHISLO	IHCSLOG	For a short precision logarithmic operation (LN)	210
IHISOR		For a short precision OUTREAL operation	810
IHISSC	IHCSSCN	For a short precision sine or cosine operation (SIN or COS)	260
IHISSQ	IHCSSQRT	For a short precision square root operation (SQRT)	170
IHISYS		For SYSACT	1520

Figure 23. Table of ALGOL Library Modules.
All are contained in SYS1.ALGLIB except IHIERR which is in SYS1.LINKLIB.
For mathematical routines, the corresponding name in the FORTRAN IV library is also given.

Appendix B: IBM-Supplied Cataloged Procedures

The four cataloged procedures for ALGOL that were introduced in Section 2 are contained in the procedure library, SYS1.PROCLIB, of the operating system. They consist of the job control statements listed below.

In order to provide support for the dedicated work file facility, temporary dsnames are specified in all four procedures for the temporary data sets SYSUT1, SYSUT2, and SYSUT3.

The procedures may be used with any of the operating system job schedulers. When parameters required by a particular scheduler are encountered by another scheduler not requiring those parameters, either they are ignored or alternative parameters are substituted automatically. For example, if these procedures are used with a sequential scheduler the following parameters, which are required for the multiprogramming option with variable number of tasks (MVT), are treated as follows:

REGION=xxxxK is ignored
SYSOUT=B is interpreted as UNIT=SYSCP
DISP=SHR is interpreted as DISP=(OLD,KEEP)

Before use, these procedures should be studied with a view to modifying them for greater efficiency within the particular environment of the installation.

In installations using the MVT option of the operating system, the REGION specifications for the compilation and linkage editing steps must be altered where necessary to suit the available storage. The REGION specification for the compilation step must be at least 4K bytes greater than the storage specified in the compiler SIZE option. When a blocked SYSIN data set is used, the REGION specification may have to be altered (see Figure 7). In the three procedures in which the linkage editor is invoked, a REGION of 96K has been specified for the linkage editing step. If necessary, this REGION specification may be reduced to conserve storage. The minimum REGION specifications for the various design levels of the Linkage Editor are:

Linkage Editor	REGION Specification
E15	24K
E18	26K
F44	54K
F88	96K
F128	136K

Installations using the MVT option must also insert a REGION specification for the execution step in procedure ALGCFCLG, unless the default interpretation is acceptable. The default interpretation is the size required by the system task initiator (i.e., 50K).

Installations not using the MVT option of the operating system should remove the superfluous parameters.

In addition, the following general recommendations should be considered:

When the MVT option is used, a SPACE parameter may be required for SYSPRINT if the device is other than a printer.

The PARM fields for compilation and linkage editing steps should follow installation conventions

The SPACE and UNIT parameters for temporary data sets should be modified according to installation configuration and conventions

Blocking factors should be specified for output data sets

For further information on writing installation cataloged procedures, see the publication OS Data Management for System Programmers.

Compilation, ALGOFC

```
//ALGOL EXEC PGM=ALGOL,REGION=48K                                       00020000
//SYSPRINT DD SYSOUT=A                                                  00040000
//SYSPUNCH DD SYSOUT=B                                                  00060000
//SYSLIN DD DSN=&LOADSET,UNIT=SYSSQ,SEP=SYSPUNCH,SPACE=(3600,(10,4)),  *00080000
//          DISP=(MOD,PASS)                                             00100000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SEP=SYSPRINT,SPACE=(1024,(50,10))    00120000
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1024,(50,10))      00140000
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1024,(40,10))                 00160000
```

Compilation and Linkage ALGOFCL

```
//ALGOL EXEC PGM=ALGOL,REGION=48K                                       00020000
//SYSPRINT DD SYSOUT=A                                                  00040000
//SYSPUNCH DD SYSOUT=B                                                  00060000
//SYSLIN DD DSN=&LOADSET,UNIT=SYSSQ,SEP=SYSPUNCH,SPACE=(3600,(10,4)),  *00080000
//          DISP=(MOD,PASS)                                             00100000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SEP=SYSPRINT,SPACE=(1024,(50,10))    00120000
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1024,(50,10))      00140000
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1024,(40,10))                 00160000
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,LET',COND=(5,LT,ALGOL),REGION=96K  00180000
//SYSPRINT DD SYSOUT=A                                                  00200000
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)                              00220000
//       DD DDNAME=SYSIN                                                00240000
//SYSLIB DD DSN=SYS1.ALGLIB,DISP=SHR                                    00260000
//SYSLMOD DD DSN=&GOSET(GO),UNIT=SYSDA,DISP=(MOD,PASS),                *00280000
//           SPACE=(1024,(50,20,1))                                     00300000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SEP=(SYSLIB,SYSLMOD),               *00320000
//          SPACE=(1024,(50,20))                                        00340000
```

Compilation, Linkage Editing and Execution, ALGOFCLG

```
//ALGOL EXEC PGM=ALGOL,REGION=48K                                       00020000
//SYSPRINT DD SYSOUT=A                                                  00040000
//SYSPUNCH DD SYSOUT=B                                                  00060000
//SYSLIN DD DSN=&LOADSET,UNIT=SYSSQ,SEP=SYSPUNCH,SPACE=(3600,(10,4)),  *00080000
//          DISP=(MOD,PASS)                                             00100000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SEP=SYSPRINT,SPACE=(1024,(50,10))    00120000
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1024,(50,10))      00140000
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1024,(40,10))                 00160000
//LKED EXEC PGM=IEWL,PARM='XREF,LIST,LET',COND=(5,LT,ALGOL),REGION=96K  00180000
//SYSPRINT DD SYSOUT=A                                                  00200000
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)                              00220000
//       DD DDNAME=SYSIN                                                00240000
//SYSLIB DD DSN=SYS1.ALGLIB,DISP=SHR                                    00260000
//SYSLMOD DD DSN=&GOSET(GO),UNIT=SYSDA,DISP=(MOD,PASS),                *00280000
//           SPACE=(1024,(50,20,1))                                     00300000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSDA,SEP=(SYSLIB,SYSLMOD),               *00320000
//          SPACE=(1024,(50,20))                                        00340000
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((5,LT,ALGOL),(5,LT,LKED))            00360000
//ALGLDD01 DD SYSOUT=A                                                  00380000
//SYSPRINT DD SYSOUT=A                                                  00400000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SPACE=(1024,(20,10))                 00420000
```

Compilation and Loading, ALGOFCG

```
//ALGOL  EXEC PGM=ALGOL,REGION=48K                                      00020000
//SYSPRINT DD SYSOUT=A                                                  00040000
//SYSPUNCH DD SYSOUT=B                                                  00060000
//SYSLIN DD DSN=&LOADSET,UNIT=SYSSQ,SEP=SYSPUNCH,SPACE=(3600,(10,4)),  *00080000
//          DISP=(MOD,PASS)                                             00100000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SEP=SYSPRINT,SPACE=(1024,(50,10))    00120000
//SYSUT2 DD DSN=&SYSUT2,UNIT=SYSSQ,SEP=SYSUT1,SPACE=(1024,(50,10))      00140000
//SYSUT3 DD DSN=&SYSUT3,UNIT=SYSDA,SPACE=(1024,(40,10))                 00160000
//GO EXEC PGM=LOADER,PARM=(MAP,LET,PRINT),COND=(5,LT,ALGOL)             00180000
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)                              00200000
//SYSLIB DD DSN=SYS1.ALGLIB,DISP=SHR                                    00220000
//SYSLOUT DD SYSOUT=A                                                   00240000
//SYSPRINT DD SYSOUT=A                                                  00260000
//ALGLDD01 DD SYSOUT=A                                                  00280000
//SYSUT1 DD DSN=&SYSUT1,UNIT=SYSSQ,SPACE=(1024,(20,10))                 00300000
```

Appendix C: Card Codes

The card deck of the source program is punched line for line from the text written on the coding sheets. The card code used can be either a 53 character set in Extended Binary Coded Decimal Interchange Code (EBCDIC), or a 46 character set in Binary Coded Decimal (BCD). The latter character set has been established as standard for ALGOL by the International Standard Organization (ISO) and Deutsche Industrie Normen (DIN). Figure 24 shows these two codes.

Characters	Card Codes	
	EBCDIC	ISO/DIN
A to Z	12-1 to 0-9	12-1 to 0-9
0 to 9	0 to 9	0 to 9
+	12-8-6	12
-	11	11
*	11-8-4	11-8-4
/	0-1	0-1
=	8-6	8-3
,	0-8-3	0-8-3
.	12-8-3	12-8-3
'	8-5	8-4
(	12-8-5	0-8-4
)	11-8-5	12-8-4
blank	no punch	no punch
<	12-8-4	
>	0-8-6	
|	12-8-7	
&	12	
¬	11-8-7	
:	8-2	
;	11-8-6	

Figure 24. Source Program Card Codes

Appendix D: Object Module

The object module is in a form acceptable as input to the linkage editor, that is, its records are card images having the format of ESD, RLD, TXT and END cards (see Figure 25). It is stored either on a data set (ddname SYSLIN) in the linkage editor library, or on an output data set (ddname SYSPUNCH), or on both. The parameters LOAD and DECK, used to specify these storage options are described in 'Appendix E'.

The object module consists of:

1. An initial ESD card defining the control section. For a precompiled procedure, the procedure name (up to 6 characters) is assigned to the control section and entered into this record.

2. The Constant Pool containing all constants and strings in the module.

3. The generated instructions.

4. The Label Address Table (see Section 3) for addressing branch instructions in the module.

5. The Program Block Table containing an entry for every program block. This table indicates the active generation of data storage areas (see Section 3) and length of each data storage area.

6. The Data Set Table containing information on the current status of all data sets used. This table is not produced for precompiled procedures.

7. Program start information.

8. An END card.

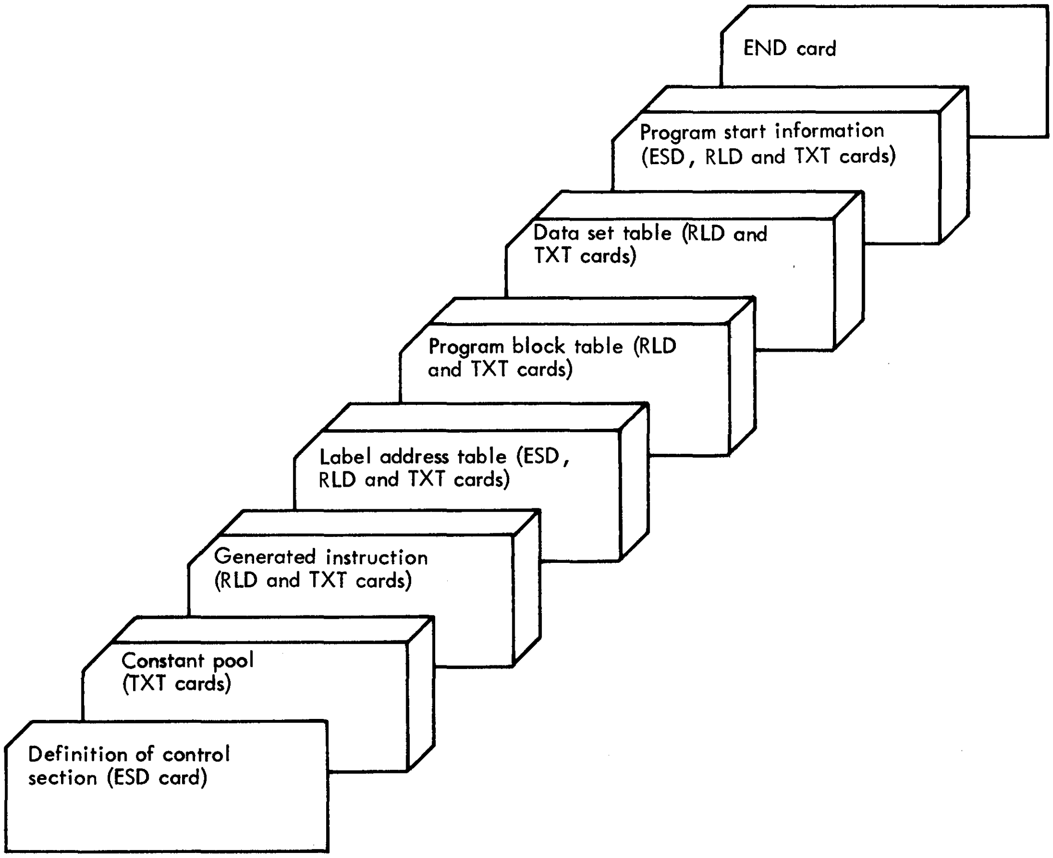

END card
Program start information (ESD, RLD and TXT cards)
Data set table (RLD and TXT cards)
Program block table (RLD and TXT cards)
Label address table (ESD, RLD and TXT cards)
Generated instruction (RLD and TXT cards)
Constant pool (TXT cards)
Definition of control section (ESD card)

Figure 25. The Object Module Card Deck.
The ESD (External Symbol Dictionary) cards contain the external symbols that are defined or referred to in the module. The RLD (Relocation Dictionary) cards contain addresses used in the module. The TXT (Text) cards contain the constants and instructions used in the module. The END card indicates the end of the module.

Appendix E: Using Job Control Language

This appendix describes the method of writing job control statements, and explains the options most frequently used by the ALGOL programmer. A full description of Job Control Language is given in OS JCL Reference.

ALGOL operates under the following options of the operating system:

1. Multiprogramming with a fixed number of tasks (MFT), using a priority scheduler,

2. Multiprogramming with a variable number of tasks (MVT), using a priority scheduler.

Communication between the user and the operating system (via the job scheduler) is effected through eight job control statements:

1. Job Statement (JOB)

2. Execute Statement (EXEC)

3. Data Definition (DD)

4. PROC Statement

5. Command Statement

6. Delimiter Statement (/*)

7. Null Statement (//)

8. Comment Statement (//*)

Parameters coded in these statements aid the job scheduler in regulating the execution of jobs and job steps, retrieving and disposing of data, allocating input/output resources, and communicating with the operator.

The control statements and their parameters are explained individually elsewhere in this appendix.

Control Statement Format

Control statements are distinguished from other statements by identifying characters (//, /* and //*), which must appear in columns 1 and 2 or 1, 2 and 3 of the standard 80-column card. Control statements contain four fields, namely the name, operation, operand, and comments fields. In some statements one or more of these fields may be vacant.

The name, operation and operand fields in a control statement may not extend beyond column 71. Column 72 must be left blank unless the statement is to be continued on another card. A statement, other than a command or comment statement, may be continued on an additional card by interrupting the statement at the end of an operand, following the operand with a comma, and (optionally) placing any nonblank character in column 72. The continuation card commences with the initial characters // in columns 1 and 2, followed by text starting in any column from 4 through 16.

Comment must be separated from the last operand by one or more blanks. If the comment is to be continued on another card, it may be interrupted at any convenient point and a non-blank character is put in column 72. The continuation card commences with the initial characters // and the comment restarts on any column from 4 to 71 inclusive.

The valid formats of each control statement are shown in Figure 26. 'Name' denotes an identifying name assigned by the programmer to the control statement. A name may contain from one to eight alphameric characters, the first of which must be alphabetic. The name is placed immediately after the initial // characters. If the name is omitted, then at least one blank must separate the // characters from the control statement operation. 'operand' denotes one or more parameters, separated by commas.

Control Statement	Valid Format
JOB	//name JOB operand comments[1]
EXEC	//name[1] EXEC operand comments[1]
DD	//name[1] DD operand comments[1]
PROC	//name[1] PROC operand comments[1]
Command	//operation (command) operand comments[1]
Delimiter	/* comments[1]
Null	//
Comment	//* comments
	[1]optional

Figure 26. Format of Control Statements

Conventions for Format Description

The conventions used in this manual for describing control statements are as follows:

Upper case letters and punctuation marks (except those listed below) represent information to be coded exactly as shown.

Lower case letters are general terms requiring substitution of specific information by the programmer.

These punctuation marks have a special meaning:

- (hyphen) links lower case words to form a single term for substitution

- (underscore) indicates the option that will be assumed if none is specified

{ } (braces) mean only one of the options contained must be selected

[ ] (brackets) mean information contained may be omitted

... (ellipsis) means that preceding item can be repeated successively a number of times.

Control Statement Coding

In the following description, certain terms are used to indicate external names which are to be specified by the programmer. These terms and their meanings are:

Term	Meaning
jobname	name of job
progname	name of program
stepname	name of job step
ddname	name of DD statement (the standard ddnames which may be specified are described in Section 2)
procname	name of cataloged procedure
procstep	name of the job step within a cataloged procedure
dsname	name of data set

It is often convenient to use two or more qualification levels to specify a data set name. The highest level reference is stated first. Thus in Figure 27, data set D.M.H. is found by searching the index of each volume in turn, starting with the system residence volume (the primary vclume in the operating system), to find the location of data set D. This, when searched, will contain the location of data set D.M. which in turn will contain the location of data set D.M.H.

```
volume index    _A__D__________Z_

data set D      _A__________M__Z_

data set D.M.   _A_______H_____Z_
```

Figure 27. Data Set Cataloging Using Qualified Names

A maximum of 44 characters can be used for a qualified name. Thus, since a simple name can consist of between one and eight characters, and each name must be separated by the character period (.), a maximum of 22 qualification levels is possible.

Data set names can also be qualified by a suffix, that is, 'dsname (element)', to indicate the relative generation number. For example, WEATHER (0) is the current generation of the data set named WEATHER. The preceding generation would be WEATHER (-1). A new generation during creation is known as WEATHER (+1), at the end of the job it becomes WEATHER (0). A suffix is also used to indicate the name of a member of a partitioned data set, or the area of an indexed sequential data set.

There are four types of job control parameters for inclusion in the operand fields: positional parameters, keyword parameters, positional subparameters and keyword subparameters.

Positional parameters must be stated first, and where more than one can be included they must be listed in the order given in the following descriptions. A comma must be substituted in place of any positional parameter omitted, if it is to be followed by another positional parameter, for example,

```
//name  operation  pos1,,pos3......
```

Keyword parameters can be listed in any order. They contain a keyword followed by an equal sign (=) and some specific information. All keyword parameters are optional since a default option will exist for any which must be specified.

One or more subparameters can be substituted for a positional parameter and also for the information to the right of the equal sign in the keyword parameter.

Positional subparameters have the same configuration and restrictions as positional parameters.

Keyword subparameters have the same configuration and restrictions as keyword parameters.

When two or more subparameters are used, they must be separated by commas and the list enclosed in parentheses, for example,

```
// name  operation  pos1,pos2,key1=value,
//                     key2=(sub1,sub2)
```

Since some special characters, such as the comma, parenthesis, blank and equal sign, have a special significance when used in control statements, no special characters can usually be used in job control information provided by the user. There are, however, some exceptions to this rule. The special characters @, $, and # can be represented normally. All other special characters, except the apostrophe, can be represented normally in the programmer's name in the JOB statement, the acounting information in the JOB and EXEC statements, and the PARM parameter options in the EXEC statement, provided that the information is enclosed in apostrophes (replacing the parentheses for a list of more than one subparameter). An apostrophe within this information is represented by two consecutive apostrophes.

JOB STATEMENT

The name field of the JOB statement must contain the external name for the job (jobname).

The operation field must contain the characters JOB.

The parameters available for the operand field are listed in Figure 28, where:

accounting information
identifies the installation account number to which the computer time for this job is to be charged. If the installation has an appropriate accounting routine, the account number can be followed by other subparameters, which are fixed by the user for his own installation. If the account number is omitted then its absence must be indicated with a comma.

programmer's name
identifies the person responsible for the job. It must not exceed 20 characters.

TYPRUN=HOLD
indicates that the job is not to be processed until a RELEASE command is issued by the operator.

PRTY=job priority
indicates the relative priority of the job. A number from 0 to 13 is specified, with 13 being the highest priority.

COND=((code, operator),...)
allows conditions for the termination of the job to be specified. Up to eight (code, operator) specifications may be included in a COND parameter. Any number between 0 and 4095 is substituted for 'code' and one of the following six relationships is substituted for 'operator'.

Operator	Meaning
GT	greater than
GE	greater than or equal to
EQ	equal to
NE	not equal to
LE	less than or equal to
LT	less than

Positional parameters	[accounting-information] [programmer's -name]

Keyword parameters
 	
(all optional)
	CLASS=jobclass
	TYPRUN=HOLD
	PRTY=job-priority
	COND=((code,operator),...)
	MSGLEVEL= $\left\{\begin{matrix} \text{m} \\ \text{(m,n)} \end{matrix}\right\}$
	MSGCLASS=classname
	REGION=nnnnnK
	ROLL =($\left\{\begin{matrix} \underline{\text{YES}} \\ \text{NO} \end{matrix}\right\}$, $\left\{\begin{matrix} \text{YES} \\ \underline{\text{NO}} \end{matrix}\right\}$)
	TIME=(minutes, seconds)

Figure 28. JOB Statement Parameters

At the completion of each job step, unless a system error occurs, the operating system will generate a return code between 0 and 4095 (see Section 1) to indicate if the program was executed successfully or not. If any of the code numbers stated in the COND parameter is related to the return code in the way specified by the associated operator then the job is terminated. For example, if

```
COND=((50,GE),(60,LT))
```

then the job will continue as long as the return codes range from 51 through 60.

MSGLEVEL= $\left\{\begin{matrix} \text{m} \\ \text{(m,n)} \end{matrix}\right\}$

specifies the information the job scheduler is to write as output from a job. 'm' denotes an integer (0,1, or 2) indicating the job control statements to be printed, as follows:

m=0: only the JOB statement is to be printed

m=1: all job control statements, including cataloged procedure statements (with actual parameters substituted for symbolic parameters), are to be printed

m=2: all input job control statements, but no cataloged procedure statements are to be printed

'n' denotes an integer constant (0 or 1) indicating whether allocation and/or termination messages are to be printed, as follows:

n=0: no allocation and/or termination messages are to be written unless the job terminates abnormally

n=1: all allocation and/or termination messages are to be written

If MSGLEVEL=0 or MSGLEVEL=1 is specified, the system assumes MSGLEVEL=(0,1) or MSGLEVEL=(1,1) respectively. If the MSGLEVEL parameter is omitted, the default value defined in the reader interpreter procedure is assumed.

MSGCLASS=classname

allows job scheduler messages to be written in a system output class other than the one normally used by the installation. The user can fix up to 36 different classes (A to Z and 0 to 9), depending on device type, priority, destination, etc., for these messages. This parameter is not necessary if the normal class (A) is required.

REGION=nnnnK

indicates the main storage size that is to be allocated to the job (including system components) instead of the default value established in the input reader procedure. nnnn is replaced by a value between 0 and 16384; thus 32 would represent 32 x 1024 = 32768 bytes. This parameter can be used only with priority scheduling.

CLASS=jobclass

indicates the relative class of a job in systems with MFT. 'jobclass' is replaced by an alphabetic character, A through O.

ROLL =($\left\{\begin{matrix} \underline{\text{YES}} \\ \text{NO} \end{matrix}\right\}$, $\left\{\begin{matrix} \text{YES} \\ \underline{\text{NO}} \end{matrix}\right\}$)

indicates the rollout/rollin attributes associated with a job in MVT systems. The first subparameter specifies if the job steps in this job can be rolled out to provide main storage space for job steps in other jobs. The second parameter specifies if the job steps in other jobs may be rolled out to provide main storage space for job steps in this job. The ROLL parameter can be specified in EXEC statements to control rollout/rollin for individual job steps.

TIME=(minutes,seconds)

limits the computing time used by a job by assigning a maximum time for its completion. If the job is not completed in this time, it is terminated.

The time is coded in minutes and seconds. The number of minutes cannot exceed 1439 (23 hours, 59 minutes); the number of seconds cannot exceed 59. (If the job execution time is expected to exceed 1439 minutes, TIME=1440 can be coded to eliminate job timing.)

If the TIME parameter is omitted, the default job time limit (as established in the cataloged procedure for the reader/interpreter) is assumed.

EXEC STATEMENT

The name field contains the external name of the job step (stepname). It may be omitted if no reference is to be made to the EXEC statement in another statement.

The operation field must contain the characters EXEC.

The parameters available for the operand field are listed in Figure 29, where:

PGM=progname

indicates that the job step executes the program named 'progname'. The program must reside on a partitioned data set.

PGM=*.stepname.ddname

indicates that the job step executes the program named by the DSNAME parameter of a DD statement named 'ddname' that was included in a previous job step named 'stepname' in the same job. If 'stepname' refers to a job step invoking a cataloged procedure then a job step within the procedure can be specified by putting its name after 'stepname'; that is, 'stepname.procstep'. The program must reside on a partitioned data set.

PROC=procname

indicates that the job step executes the cataloged procedure named 'procname'.

procname

has the same effect as PROC=procname

TIME=(minutes,seconds)

limits the computing time for the job step. If 'seconds' only is specified then a comma must be substituted for 'minutes'. If 'minutes' only is specified then the parentheses can be deleted.

```
COND=((code,operator,stepname),...
                    ... [,{EVEN}],...)
                          {ONLY}
```

allows conditions to be specified for bypassing and/or for executing a job step.

A condition specification of the form (code, operator, stepname) specifies that the job step is to be bypassed if a comparison, using the relation denoted by 'operator', between the number denoted by 'code' and the return code issued by the preceding job step denoted by 'stepname', is satisfied (true). The terms 'code' and 'operator' are governed by the same stipulations as those specified for these terms in the JOB statement. If 'stepname' is not specified, the condition code test is applied to all preceding job steps. If a test is to be applied to a step in a cataloged procedure, then the name of the job step which invoked the procedure, followed by the procedure step name, must be specified, as follows: 'stepname.procstep'.

The EVEN and ONLY subparameters are mutually exclusive. One or the other may be specified, either alone or in combination with up to seven return code tests. EVEN specifies that the step is to be executed in any event (irrespective of an abnormal termination by a preceding job step), unless one or more of the return code tests specified in this step are satisfied. ONLY specifies that the step is to be executed only if a preceding job step is terminated abnormally, and provided none of the return code tests specified in this step is satisfied.

PARM=subparameter list

indicates the special options which the programmer has chosen to apply to the job step. Each option, or subparameter, in the subparameter list is represented by a keyword (in a few cases, the subparameter may have the form keyword=number). The subparameters, separated by commas, may be listed in any order. If two or more subparameters are listed, then the list must be enclosed in apostrophes. Parentheses may be used instead of apostrophes if the subparameter list contains no special characters other than the comma. The subparameter list, including apostrophes, may be a maximum of 100 characters in length.

The options which may be exercised for the job steps compilation, linkage editing, program execution and program loading (by use of the loader) are listed below. In

Positional parameters	$\left\{\begin{array}{l}\text{PGM=program}\\\text{PGM=*.stepname.ddname}\\\text{PROC=name}\\\text{procname}\end{array}\right\}$
Keyword parameters (all optional)	$\left\{\begin{array}{l}\text{TIME}\\\text{TIME.procstep}\end{array}\right\}$=(minutes, seconds) $\left\{\begin{array}{l}\text{COND}\\\text{COND.procstep}\end{array}\right\}$=((code,operator,stepname),...) $\left[,\left\{\begin{array}{l}\text{EVEN}\\\text{ONLY}\end{array}\right\}\right]$,....) $\left\{\begin{array}{l}\text{PARM}\\\text{PARM.procstep}\end{array}\right\}$=subparameter-list $\left\{\begin{array}{l}\text{ACCT}\\\text{ACCT.procstep}\end{array}\right\}$=accounting-information $\left\{\begin{array}{l}\text{REGION}\\\text{REGION.procstep}\end{array}\right\}$ =nnnnnK1 ROLL=($\left\{\begin{array}{l}\underline{\text{YES}}\\\text{NO}\end{array}\right\}$,$\left\{\begin{array}{l}\text{YES}\\\underline{\text{NO}}\end{array}\right\}$) $\left\{\begin{array}{l}\text{DPRTY}\\\text{DPRTY.procstep}\end{array}\right\}$ = $\left\{\begin{array}{l}\text{m}\\\text{(m,n)}\\\text{(,n)}\end{array}\right\}$

Figure 29. EXEC Statement Parameters

most cases, each option represents a choice between two alternatives, one of which, called the default option, is assumed to apply unless the other is specified, either at this stage or at system generation. In the lists which follow, the keyword associated with the default option is underscored.

ALGOL Compiler Options

All of the alternative options but PROGRAM and TEST can be changed to the default option at system generation. Abbreviated forms are provided for most of the option keywords. The abbreviations, indicated below, may be used in place of the full keywords.

PROGRAM or PROCEDURE: The source program is either an ALGOL program in the sense of the ALGOL syntax (PROGRAM) or an ALGOL procedure to be compiled separately and used with other programs or procedures (PROCEDURE). Abbreviated forms PG or PC.

SHORT or LONG: The internal representation of real values is in full words (SHORT) or double words (LONG). Abbreviated forms SP or LP.

NODECK or DECK: An object module, stored on the data set specified in the SYSPUNCH DD statement, either is not to be generated (NODECK); or is to be generated (DECK). Abbreviated forms ND or D.

LOAD or NOLOAD: The compiler is to either generate an object module for use as input to the linkage editor, using the data set specified in the SYSLIN DD statement (LOAD); or is not to generate this object module (NOLOAD). Abbreviated forms L or NL.

SOURCE or NOSOURCE. The source program and identifier table listings are either to be printed (SOURCE); or not to be printed (NOSOURCE). Abbreviated forms S or NS.

EBCDIC or ISO: The card code used to write and keypunch the source program is either a 53 character set in EBCDIC (EBCDIC): or the 46 character set in BCD which has been established as standard for ALGOL by ISO and DIN (ISO). Abbreviated forms EB or I.

TEST or NOTEST: The generated object module is or is not to include coding useful in execution time error detection and diagnosis. The coding consists of instructions to produce the semicolon count, instructions to check the values of subscript expressions against array bounds, and instructions to check the dimensions of formal arrays against the dimensions of actual arrays. Abbreviated forms T or NT.

SIZE=45056 or SIZE=number: The main storage size that is available to the compiler is either 45,056 bytes or the size in bytes denoted by number. 'Number' must not be less than 45056 and must not exceed 999999.

Linkage Editor Options

For the linkage editing job step the options are of two types: those which specify the output listings required, and those specifying attributes for the load module.

The options to control output listings are:

LIST: All job control statements processed by the linkage editor are to be listed on the diagnostic output data set.

MAP or XREF: A map of the load module is to be produced (MAP) or a cross-reference table of the load module is to be produced (XREF) comprising a load module map and a list of all address constants that refer to other control sections.

The options specifying load module attributes which can be used with ALGOL programs are:

REUS: A load module is to be produced that is serially reusable, that is, it can be used by more than one task, but only one task at a time.

DC: A load module is to be produced that is downward compatible, that is, if the load module is produced by an F level linkage editor then it can be reprocessed by an E level linkage editor.

LET or XCAL: The load module is to be marked as executable even when a severity 2 error is detected (LET); or the load module is to be marked as executable even though valid exclusive references between the segments have been made (XCAL). A severity 2 error could make execution impossible and would normally lead to the load module being marked as not executable. It includes the situation over-ridden by XCAL.

NCAL: The linkage editing automatic library call mechanism is not to call library members to resolve external references within the object module. The load module is marked as executable even though unresolved external references have been recognized.

All the linkage editor subparameters are optional.

Program Execution Options

For the execution job step of an ALGOL program the options are:

TRACE: The semicolon count produced during the compilation process is to be printed as a list. This gives information on the dynamic flow of the program and is known as a program trace.

TRBEG=number: A limited program trace is to be produced beginning at the semicolon specified by 'number' and ending at the physical end of the program.

TREND=number: A limited program trace is to be produced beginning at the physical beginning of the program and ending at the semicolon specified by 'number'.

The last two options may be specified together to define the beginning and end of the trace. When either is specified, TRACE may be omitted, but in that case precompiled procedures would not be included. If TRACE is specified with TRBEG or TREND, then only a limited program trace is produced, but it will include precompiled procedures executed in that part of the program.
No program trace is possible if NOTEST has been specified for the compilation process.

DUMP: A partial main storage dump is to be produced if an error occurs. The dump shows the contents of the data storage areas and arrays.

All of the execution time subparameters are optional.

Loader Options

For the loader step, options may be specified both for the loader and for the loaded program or load module. The options are specified together in the PARM field, as follows:

PARM='loader options/program options'

where 'loader options' denotes the option keywords (separated by commas) specified for the loader, and 'program options' denotes the option keywords specified for the loaded program or load module. The two keyword lists must be separated by (/). If there are no loader options, the program options must begin with a slash. The entire PARM field may be omitted if no options are to be specified for the loader or the loaded program (or load module).

The program options (TRACE, TRBEG, TREND and DUMP) are described above.

The loader options are:

MAP or NOMAP: A map of the loaded program, listing external names and their absolute storage addresses, is or is not to be produced on the SYSLOUT data set. If the input deck does not include a SYSLOUT DD statement, the option is ignored.

RES or NORES: An automatic search of the link pack area queue is or is not to be made. The search is always made after processing the primary input (SYSLIN) and before searching the SYSLIB data set.

CALL or NOCALL: An automatic search of the SYSLIB data set is or is not to be made. If the input deck does not include a SYSLIB DD statement, the option is ignored.

LET or NOLET: The loader is or is not to try to execute the object program in the event that a severity 2 error condition is found. A severity 2 error condition is one that could make execution of the loaded program impossible.

SIZE=number or SIZE=100K: The size of dynamic main storage that can be used by the loader is either the size in bytes denoted by 'number' or 100K bytes. Normally, this value will be 17K plus the size of the program to be loaded (for MFT systems) or 18K plus the loaded program size (for MVT systems).

EP=name: The name denoted by 'name' is the external name to be assigned as the entry point of the loaded program. If all input to the loader consists of load modules, the parameter EP=IHIFSAIN must be specified. IHIFSAIN is the entry point of an ALGOL program.

PRINT or NOPRINT: Diagnostic messages are or are not to be produced on the SYSPRINT data set.

ACCT=accounting information
allows accounting information associated with the job step to be passed to the installation's accounting routines, using subparameters which are fixed by the user for his own installation.

REGION=nnnnnK
indicates the main storage size for the job step if it has not already been specified in the JOB statement.

ROLL=$\left(\begin{Bmatrix}\underline{\text{YES}}\\\text{NO}\end{Bmatrix},\begin{Bmatrix}\text{YES}\\\underline{\text{NO}}\end{Bmatrix}\right)$
declares the job step's ability or inability to be temporarily rolled out of main storage, as well as the job step's ability or inability to cause the temporary rollout of another job step. If the first subparameter is YES, the present job step may be temporarily transferred to auxiliary storage, in the event another job step, qualified to cause rollout, requires additional main storage space beyond its original region. If the first subparameter is NO, then the present job step cannot be rolled out.

If the second subparameter is YES, the present job step is qualified to cause the rollout of another job step, in the event the present job step requires additional space beyond its original region. If the second subparameter is NO, then the present job step cannot cause rollout.

When the present job step invokes a cataloged procedure, ROLL attributes may be specified for an individual step in the procedure, as in the following example: ROLL.procstep=(YES,YES), where 'procstep' denotes the name of the particular step. If no step name is given, then the attributes specified apply to all steps in the cataloged procedure.

The ROLL parameter may be used only in MVT systems.

$$\text{DPRTY=}\left\{\begin{array}{l}\text{m}\\ \text{(m,n)}\\ \text{(,n)}\end{array}\right\}$$

assigns a dispatching priority to the job step. This parameter can be used only with priority scheduling. 'm' and 'n' denote integers in the range 0-15. 'm' is converted by the system into an internal priority and 'n' added to this priority to obtain the dispatching priority. Where possible, 'm' should be 14 or less, as the priority 15 is assigned to certain system takes. If the DPRTY parameter is omitted, the job step is assigned the priority specified for the job.

DD STATEMENT

The name field contains an identifying name (ddname) for the DD statement.

The operation field must contain the characters DD.

The parameters available for the operand field are listed in Figure 30, where:

*

indicates, when used as a positional parameter, that the required data follows immediately after this DD statement. The asterisk must be the only non-blank character in the operand field. For sequential scheduling it can be used only once in each job step, and the data must be followed by a delimiter statement.

DUMMY

indicates that the user's problem program is to be executed without any I/O operations on the data set. This can be used for debugging, and also for bypassing data set references in a regularly used program, for example, the first run of an updating program when there is no old master to be processed.

$$\text{DSNAME=}\left\{\begin{array}{l}\text{dsname}\\ \text{dsname(number)}\\ \text{dsname(membername)}\end{array}\right\}$$

'dsname' denotes the name of an existing data set or the name defined for a data set to be created in the present job step. In the latter case, if the data set is to be kept (see the DISP parameter below), the name thus defined is the name by which the data set must be identified in other jobs. Within the present job, the data set may be identified in later steps either by the defined name or by reference to the DD statement in this job step (see the item after next).

If the data set being defined is an indexed sequential data set (in which case a group of DD statements are required), the data set name must be followed by one of the terms INDEX, PRIME or OVFLOW, whichever applies, in parentheses. 'dsname (number)' denotes the name and generation number of a generation data group. 'dsname(membername)' denotes the name of a partitioned data set.

$$\text{DSNAME=}\left\{\begin{array}{l}\text{\&dsname}\\ \text{\&\&dsname}\end{array}\right\}$$

specifies the name of a temporary data set that is to be deleted at the end of the present job. The data set may be identified, within this job, either by the name '&dsname' or '&&dsname', whichever applies, or by reference to the DD statement in which the data set is first identified (see next item).

A temporary data set name preceded by a single ampersand (i.e., '&dsname') occurring inside a cataloged procedure is treated as a symbolic parameter if a value is assigned to it in an EXEC statement which invokes the procedure or in a PROC statement in the procedure. Where the DD statement refers to a member of a temporary partitioned data set, the temporary data set name should be followed by the member name, i.e., &&dsname(membername). Similarly, in a group of DD statements defining an indexed sequential data set, the temporary data set name should be followed by one of the terms INDEX, PRIME or OVFLOW, whichever applies, in parentheses.

$$\text{DSNAME=}\left\{\begin{array}{l}\text{\&ddname}\\ \text{\&\&ddname}\end{array}\right\}$$

indicates that a pre-allocated data set is to be used. This parameter can be used only in systems with MVT. 'ddname' denotes the name of a DD statement in the initiator cataloged procedure which defines the pre-allocated data set to be used. All parameters used to define a new data set must also be coded; if the pre-allocated data set cannot be assigned, the parameters are used to create a temporary data set. (For detailed information on pre-allocated data sets, refer to the publication OS Data Management for System Programmers.)

DSNAME=*.stepname.ddname
indicates that the data set is the one specified in a preceding DD statement named 'ddname' occurring in the job step named 'stepname'. If the data set was specified in the current job step then 'stepname' must be omitted. If 'stepname' refers to a job step invoking a cataloged procedure, a job step within the procedure can be specified by putting its name after 'stepname'; that is, '*stepname.procstep.ddname'.

Note. If the DSNAME parameter is omitted, the operating system will assign a unique name to any data set created by the job step.

```
DCB=(*.stepname.ddname )
    {dsname            }
    (subparameter-list )
```

indicates that the data control block for the data set specified in the DD statement named 'ddname' in the job step named 'stepname', or alternatively the cataloged data set named 'dsname', is to be repeated for the current DD statement. 'Stepname' must be omitted if it refers to the current job step, or may be qualified in the same way as the DSNAME parameter if it refers to a job step in a cataloged procedure. If additional information is substituted for 'subparameter list' then this over-rides the corresponding subparameters in the repeated information. Alternatively, 'subparameter list' can be used alone to specify data control block information.

The subparameter list for the data sets used when processing and executing an ALGOL program contains the following keyword subparameters:

BLKSIZE=number, is used to specify blocksize. 'Number' is blocksize in bytes, and for fixed length records must be a multiple of record length.

RECFM=F [B] [A], is used to specify record format. F=fixed length, B=blocked, A=control character incorporated to control printed output format.

Positional parameters (all optional)	{*
	DUMMY}

Keyword parameters (all optional, though DSNAME can be omitted only when the asterisk positional parameter is used	DSNAME= { dsname
	&dsname
	&ddname
	*.stepname.ddname }
	
	DCB= { *.stepname.ddname
	dsname
	subparameter-list }
	
	{ AFF=ddname
	SEP=subparameter-list }
	
	UNIT=subparameter-list
	
	{ SPACE=subparameter-list
	SPLIT=subparameter-list
	SUBALLOC=subparameter-list }
	
	VOLUME=subparameter-list
	
	LABEL=subparameter-list
	
	{ DISP=subparameter-list
	SYSOUT=subparameter-list }

Figure 30. DD Statement Parameters

LRECL=value, is used to specify record length. 'Value' is actual length in bytes.

All other valid DCB options are fixed.

AFF=ddname
indicates that the data set has affinity with the data set specified by the DD statement named 'ddname' and is to use the same channel.

SEP=list-of-ddnames
indicates that the data set is to use a separate channel to the ones used by the data sets specified by the DD statements named in the 'list-of-ddnames'.

UNIT=subparameter list
specifies the class and quantity of I/C devices to be allocated for use by a data set. The subparameter list has two forms, either one of which may be used in an individual statement. The two forms are:

1	Positional subpara-meters	classname$\left\{\begin{matrix}, \underline{1}\\ \text{,number}\\ , \text{P}\end{matrix}\right\}$[,DEFER]
	Keyword subparameter	[SEP=list-of-ddnames]
2	Keyword subparameter	AFF=ddname

'classname' indicates the device class. These names are divided into two categories.

- Those automatically incorporated in the operating system when it is generated. These are of two types - specific unit names, such as 2400 (for a magnetic tape drive) and 1403 (for a printer); and general classnames, that is,

  SYSCP for any card punch

  SYSSQ for any magnetic tape or direct-access device

  SYSDA for any direct-access device.

- Additional names fixed by the user for his installation when the operating system is generated.

'number' indicates the number of devices to be allocated. If the data set is cataloged but the number of devices used is unknown, then 'P' substituted for 'number' will ensure that the correct number is assigned.

DEFER indicates that the volume need not be mounted on the I/O device until the data set is called in the program. This subparameter must not be used with an indexed sequential data set or a new output data set on a direct-access device.

SEP=list-of-ddnames indicates for direct-access devices that, if possible, the data set is not to use the same access arm as the data sets specified by the DD statements, given in the 'list-of-ddnames.'

AFF=ddname indicates that the data set is to use the same I/O devices as the data set specified in the DD statement named 'ddname' in the same job step.

SPACE=subparameter list
indicates the space required when a direct-access device is specified in the UNIT parameter. Space may be requested (a) in terms of a given number of tracks, cylinders or blocks, with no particular track address being specified, or (b) in terms of a given number of tracks, starting at a particular track address.

(a) Where the space request is made in terms of a given number of tracks, cylinders or blocks, with no address specified, the subparameter list depends in part on the organization of the data set.

For a sequential data set, the general form of the subparameter list is

```
({TRK  },(quty,[increment],drctry)[...])
 {CYL  }
 {Blksz}
```

The first subparameter indicates the unit in which the space requested is expressed, namely tracks, cylinders or blocks. The unit of a block is indicated by the blocksize in bytes. 'quty' denotes the number of tracks, cylinders or blocks requested. 'Increment' denotes the incremental number of tracks, cylinders or blocks which are to be added to the space allocation whenever the data set exhausts its last allocation. The last term, [...], represents a list of further optional parameters, explained at the end of this item.

For a partitioned data set, the general form of the subparameter list is

```
({TRK  },(quty,[increment],drctry)[...])
 {CYL  }
 {Blksz}
```

The first three subparameters are identical with those described in the preceding paragraph. 'Drctry' denotes the number of 256-byte blocks to be allocated to the data set directory. The last term, [...], represents a list of further optional parameters, explained at the end of this item.

For an indexed sequential data set, the general form of the subparameter list is

```
({TRK  },(quantity,,index)[...])
 {CYL  }
 {Blksz}
```

The first three subparameters are identical with those described in a preceding paragraph. 'Index' denotes the number of cylinders required for the data set index.

The term [...] contained in each of the preceding three symbolic parameter lists represents the following list of additional optional subparameters:

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

RLSE indicates that any unused space remaining after the data set has been created, is to be released.

CONTIG specifies that space is to be allocated in contiguous tracks or cylinders. MXIG specifies that the largest single block of auxiliary storage available is to be allocated to the data set. ALX requests that up to five areas of contiguous storage, each at least as large as the area requested, be allocated. Where this request cannot be fully satisfied, the system allocates as many blocks as are available.

ROUND specifies that, when the space request is expressed in blocks, the space request be rounded to an integral number of cylinders.

(b) Where the space request is made in terms of a given number of tracks starting at a specific track address, the general form of the subparameter list is

```
(ABSTR, quantity, address[,directory])
```

'Quantity' denotes the number of tracks required. 'Address' denotes a number representing the relative address of the first track where the space allocation is to begin. The tracks are numbered consecutively, starting with 0 for the first track on the volume. The first track cannot be allocated. 'Directcry', a subparameter required when a data set is partitioned, denotes the number cf 256-byte blocks required for the data set directory.

SPLIT=subparameter list
provides a means of requesting space on a direct-access device in such a way as to divide (or split) each cylinder between two or more associated data sets. This can be used to minimize access arm movements when two or more data sets with corresponding records are processed simultaneously.

The splitting of cylinders requires a sequence of DD statements, the first of which specifies the space per cylinder required for the first data set, as well as the total space required for all associated data sets. Each succeeding DD statement specifies the space request for one of the other associated data sets. The space request may be expressed in cylinders and tracks or in terms of blocks.

Where the space request is expressed in cylinders and tracks, the subparameter list of the SPLIT parameter in the leading DD statement has the following general form:

```
        { quantity                   }
(n,CYL,{(quantity[,increment])}     )
```

where 'n' denotes the number of tracks per cylinder required by the first data set, and 'quantity' denotes the total number of cylinders to be allocated for all associated data sets. Each succeeding DD statement in the group must contain the parameter SPLIT=n, where 'n' denotes the number of tracks per cylinder to be allotted to the associated data set. 'Increment' denotes an additional amount of space tobe allocated any one data set each time it exhausts its original space.

When the space request is expressed in blocks, the subparameter list in the leading DD statement has the following general form:

```
            { quantity                }
(%,blksize,{(quantity[,increment])}  )
```

where '%' denotes the percentage of tracks per cylinder to be allocated to the first data set, 'blksize' denotes the average block length in bytes; and 'quantity' denotes the total number of blocks required. Each succeeding DD statement in the group must contain the parameter SPLIT=%, where '%' denotes the percentage of tracks per cylinder to be allotted to the associated data set.

SUBALLOC=subparameter list
provides a method of placing a number of data sets consecutively on a direct-access volume. The method consists in suballocating a portion of the space allocated to a data set in a preceding DD statement, to another data set. Suballocations are made from the front of the space allocated to the original data set. The original data set may be used only for suballocations. The general form of the subparameter list is:

```
({TRK  },(quantities)[,ddname         ])
 {CYL  }             [,stepname.ddname]
 {Blksz}             [                ]
```

where (quantities) denotes

```
(quantity[,increment][,directcry])
```

The first subparameter indicates the unit in which the suballocation request is expressed, namely tracks, cylinders or blocks, a block being indicated by the average block length in bytes. 'Quantity' denotes the number of tracks, cylinders or blocks to be suballocated. 'Increment' denotes the additional space to be allocated to the data set when its original allocation is exhausted. Increments are made from available space on the vclume. 'Directory' denotes the number of 256-byte blocks required for the directory of a partitioned data set. 'Stepname' and 'ddname' denote the names of the job step and DD statement where the original data set is defined. If the original DD statement is contained in the same job step, 'stepname' may be omitted.

DISP=subparameter list
indicates the status of the data set and its disposition at the end of a job step. The subparameter list may contain from one to three positional subparameters, as follows:

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

The first subparameter in the list indicates the status of the data set, the second indicates the data set's disposition after a normal termination of the job step, and the third parameter indicates the disposition of the data set at the end of the job step, in the event the job step abnormally terminates.

NEW specifies that the data set is to be generated in this job step, and would be deleted at the end of the job step unless KEEP, PASS or CATLG is specified.

OLD specifies that the data set already exists, and would be kept at the end of the job step unless PASS or DELETE is specified.

MOD specifies that the data set already exists and is to be modified in this job step. If the data set cannot be found by the operating system then this parameter is equivalent to NEW.

SHR specifies that, in a multiprogramming environment, an existing data set may be used simultaneously by more than one job.

DELETE specifies that the space used by the data set (including that in the data set catalog, etc.) is to be released at the end of the job step.

KEEP specifies that the data set is to be kept at the end of the job step.

PASS specifies that the data set is to be referred to in a later step of this job, at which time its final disposition, or a further pass, will be specified.

CATLG specifies that the data set is to be cataloged at the end of the job step. Thus KEEP is implied. The catalog structure must already exist.

UNCATLG specifies that the data set is to be deleted from the catalog at the end of the job step. KEEP is implied.

SYSOUT=subparameter list
specifies the printing or punching operation to be used for the data set. The 'subparameter list' is:

classname
(classname[,progname][,number])

'classname' specifies the system output class to be used. Up to 36 different classes (A to Z, 0 to 9) may be fixed by the user for his installation, according to device type, priority, destination, etc. The standard classname is A.

Classes 0-9 should only be used when the other classes are insufficient.

'progname' can be used to specify the name of a user-written output routine.

'number' can be used to specify an installation form number to be assigned to the output.

For sequential scheduling, the 'subparameter list' consists of only the standard class-names A and B. SYSOUT=B is interpreted as UNIT=SYSCP.

OUTLIM=number
specifies the maximum number of logical records that a data set being routed through the output stream may contain. It is used only in statements where the SYSOUT parameter is coded in the same operand.

'number' indicates the maximum number of records for the data set. That number can be in the range 1 - 16,777,215. If OUTLIM=0 or no OUTLIM is coded, no output limiting is done.

OUTLIM is used in MFT and MVT systems that use the System Management Facilities Option. This facility can be used to give management a certain amount of control over the jobs run on their system. For more detailed information refer to the description of the OUTLIM parameter in <u>OS JCL Reference</u>.

VOLUME=subparameter list
indicates the volume or volumes assigned to the data set. If the data set is cataloged this parameter is not necessary. The 'subparameter list' is:

Positional subparameters	([PRIVATE] [,RETAIN] [,number] [,value])
Keyword subparameters	SER = list-of-serial-numbers REF = dsname *.ddname *.stepname.ddname *.stepname.procstep.ddname

PRIVATE specifies that the volume is to be dismounted after the job step and that other data sets will not be assigned to the volume unless a specific request is made.

RETAIN specifies that, if possible, the volume is to remain mounted until referred to in a later DD statement, or until the end of the job, whichever is first.

'number' is any number between 2 and 9999, and is used if an input or output operation on a cataloged data set residing on more than one volume does not start on the first volume of the data set. The number specifies the volume on which input or output is to start (for example, 3 indicates the third volume of the data set).

'value' specifies the number of volumes required by an output data set. It is not required if SER or REF is used.

SER=list-of-serial-numbers, specifies the serial numbers allocated by the user to the volumes required by the data set. These serial numbers can consist of between one and six characters.

```
REF=(dsname                     )
    ( *.ddname                  )
    ( *.stepname.ddname         )
    ( *.stepname.procstep.ddname)
```

specifies that this data set is to use the same volume or volumes as the data set specified by one of the alternative subparameter forms. If the latter data set resides on more than one tape volume, then only the last volume (as specified in the SER subparameter) can be used.

LABEL=subparameter list
indicates the type of label or labels associated with the data set. If the data set is cataloged this parameter is

not necessary. The general form of the subparameter list is:

```
┌([n ] ┌,NL  ┐[,PASSWORD]┌,EXPDT=yyddd┐) ┐
│      │,SL  │           └,RETPD=dddd  ┘  │
│      │,NSL │                            │
│      │,SUL │                            │
└      └,BLP ┘                            ┘
EXPDT=yyddd
RETPD=dddd
```

'n' is any number between 2 and 9999, and specifies the position of the data set on the volume (for example, 3 would indicate the third data set on the volume).

NL, SL, NSL, and SUL specify the type of label or labels to be used, that is, no labels, standard labels, non-standard labels, and standard and user labels, respectively. The routines to produce non-standard labels must be written and incorporated into the operating system by the user. BLP indicates that label processing is to be bypassed.

PASSWORD specifies that the data set is to be accessible only through the use of a password. To retrieve the data set, the operator must respond to a message by issuing the correct password.

EXPDT=yyddd specifies that the data set cannot be updated without operator intervention, until the data given by yy (year) and ddd (day).

RETPD=dddd specifies that the data set is to be retained for the number of days given by dddd.

PROC STATEMENT

The PROC statement is used to assign default values to symbolic parameters defined in a cataloged procedure. When the PROC statement is used, it appears as the first control statement in the procedure. The general form of the PROC statement is:

```
//procname PROC symparam=default value
```

where 'symparam' denotes a symbolic parameter in the cataloged procedure.

COMMAND STATEMENT

The command statement enables commands to be issued to the system via the input stream. The available commands and the appropriate operands specifiable in the command statement are explained in OS Operator's Reference.

DELIMITER STATEMENT

The delimiter statement, containing the characters /* in columns 1 and 2 of the 80-column punched card, marks the end of a data set in the input stream. In systems with MFT or MVT the end of a data set in the input stream defined by a DD * statement need not be marked by a delimiter statement.

NULL STATEMENT

The null statement consists solely of the characters // in columns 1 and 2. It is used to mark the end of a job in the input stream so as to insure that the card reader is effectively closed.

COMMENT STATEMENT

The comment statement, containing the characters //* in columns 1, 2, and 3, followed by comment in any columns from 4 through 80, is used for inserting comment before or after any control statement.

Using a Private Library

A load module to be executed with the aid of the job control facilities of the operating system may be contained in the system library (SYS1.LINKLIB) or in a user's private library. Except when otherwise indicated by control statements in the input stream, or when a load module has been created in the same job, the operating system assumes that any load module identified in an EXEC statement is contained in the system library. If a load module is contained in the system library, it may be executed by specifying its name in the EXEC statement and without explicity defining the SYS1.LINKLIB data set.

If a load module is contained in a private library, it may be executed only if the data set comprising the library is explicitly identified, by means of a suitable DD statement. Identifying a private libary is equivalent to combining (or concatenating) the private library with the system library, since the operating system searches the system library if it cannot find a load module in the private library.

A private library may be concatenated to the system library by means of the JOBLIB DD statement and/or a STEPLIB DD statement in one or more job steps.

The JOBLIB DD statement may appear once in each job and must immediately follow the JOB statement. The statement

```
//JOBLIB DD DSNAME=dsname,DISP=OLD
```

specifies that the operating system is to search for each load module named in the succeeding EXEC statements, first in the private library denoted by 'dsname' and then in the system library. This method of search applies to every step in the job, unless otherwise specified by a STEPLIB DD statement in the particular job step. One or more other private libraries may be specified by a list of additional DD statements, in which the name field is vacant, immediately following the JOBLIB statement.

A STEPLIB DD statement may be used once in each job step and may appear in any position following the EXEC statement. The statement

```
   //STEPLIB DD DSNAME=dsname,DISP=OLD
```

specifies that the operating system is to search for the load module named in the preceding EXEC statement, first in the private library denoted by 'dsname' and then, if necessary, in the system library. This method of search applies only to the job step in which the STEPLIB DD statement appears. If a JOBLIB DD statement is contained in the job, its effect is suspended during the step in which the STEPLIB DD statement appears. The statement

```
   //STEPLIB DD DUMMY
```

nullifies the JOBLIB DD statement for the particular step, and limits the load module search to the system library.

Job Control Language Examples

Five different types of jobs are described here to illustrate the use of job control language. Some of the subparameters used, such as I/O device classnames and volume serial numbers, may change for different installations.

Example 1: Executing a Single Load Module

Statement of problem (see Figure 31): A set of 80 matrices are contained in data set SCIENCE.MATH.MATRICES. Each matrix is an array containing real variables. The size of the matrices vary from 2x2 to 25x25; the average size is 10x10. The matrices are to be inverted using a program MATINV contained in a partitioned data set MATPROGS. Each inverted matrix is to be written as a single record on the data set SCIENCE.MATH.INVMATRS. The first variable in each record is to denote the size of the matrix. Each matrix is to be printed.

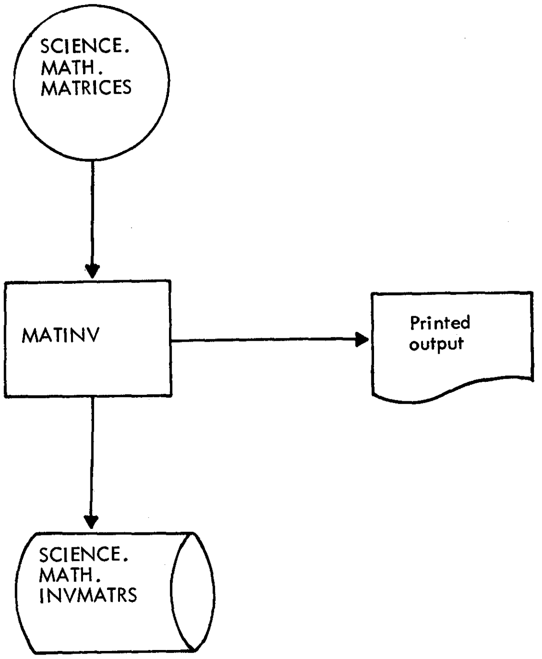


Figure 31. I/O Flow for Example 1

Explanation of coding: The job control statements used in Figure 32 specify that:

1. The job is
   - to be charged to the installation's account number 537
   - the responsibility of John Smith
   - to have all control statements (plus control statement diagnostic messages if an error occurs) printed on the normal system output device.
2. The partitioned data set MATPROGS is concatenated with the operating system library, SYS1.LINKLIB.
3. The program to be executed is MATINV.
4. The input data set is SCIENCE.MATH.MATRICES

5. The printed output is to use the standard output format class for the installation.

6. The output data set is

   - to be cataloged
   - to use the device dass DACLASS
   - to use volume 1089W
   - to use a separate channel to the input data set
   - to have space reserved for 80 records, each 1500 bytes long. This space is to be incremented in 9-record units each time more is required and any unused space is to be released. The space is contiguous and aligned on cylinder boundaries.
   - to have fixed-length blocked records, 300 bytes long, and a maximum block size of 1500 bytes.

Example 2: Compiling, Linkage Editing and Executing Three Source Programs

Statement of problem (see Figure 33): Raw data from a rocket test firing is contained in a data set RAWDATA. The forecasted results for this firing are contained in a data set PROJDATA. A program PRCGRD is to be used to produce refined data from these two data sets.

The refined data is to be stored in a temporary data set and used by a program ANALYZ, containing a series of equations, to develop values from which graphs and reports can be generated. Parameters needed by ANALYZ are contained on a cataloged data set PARAMS.

The values are to be stored on a temporary data set and used by a program REPORT to print graphs and reports. The programs PROGRD, ANALYZ and REPORT are written in ALGOL. They are still in source program form, and therefore must be compiled and linkage edited before execution.

Explanation of coding: The job control statements used in Figure 34 specify that:

1. The job is

   - the responsibility of John Smith
   - to have all control statements (plus control statement diagnostic messages if an error occurs) printed on the normal system output device for information listings.

2. The first job step invokes the ALGOFCLG cataloged procedure (see 'Appendix B') to process and execute the ALGOL source program (PROGRD) entered in the input stream.

3. The other input data sets are RAWDATA and PROJDATA. RAWDATA is also entered in the input stream.

4. The temporary output data set is

   - to be called REFDATA and to be passed for use in a later job step
   - to use the device class TAPECLS
   - to be written on volume 2107, which is to remain mounted for use later
   - to have fixed-length records, 80 bytes long, and a maximum block size of 400 bytes

5. The second job step invokes the ALGOFCLG cataloged procedure to process and execute the ALGOL source program (ANALYZ) entered in the input stream

6. The SYSLMOD DD statement in the LKED step of the cataloged procedure is overridden to specify that the load module produced by the linkage editor is to be a new member, ANALYZ, of temporary partitioned data set GOSET

7. The other input data sets are REFDATA and PARAMS. Both will be kept at the end of the job step

8. The temporary output data set is

   - to be called VALUES and is to be passed for use in a later job step.
   - to use the device class TAPECLS.
   - to be written on volume 2108.
   - to have fixed length records, 68 bytes long, and a maximum block size of 204 bytes.

9. The third job step invokes the ALGOFCLG cataloged procedure to process and execute the ALGOL source program (report) entered in the input stream. The output data will be listed on the printer specified in the cataloged procedure.

10. The SYSLMOD DD statement in the LKED step of the cataloged procedure is overridden to specify that the load module produced by the linkage editor is to be a new member, REPORT, of the temporary paritioned data set GOSET

11. The other input data set is VALUES which will be kept at the end of the job step

```
//INVERT JOB 537,JOHNSMITH,MSGLEVEL=1
//JOBLIB DD DSNAME=MATPROGS,DISP=OLD
//INVERT EXEC PGM=MATINV
//SYSIN DD DSNAME=SCIENCE.MATH.MATRICES,DISP=OLD
//SYSPRINT DD SYSOUT=A
//ALGLDD05 DD DSNAME=SCIENCE.MATH.INVMATRS,DISP=(NEW,CATLG),SEP=SYSIN, *
//             SPACE=(1500,(80,9),RLSE,CONTIG,ROUND),VOLUME=SER=1089W,  *
//             DCB=(RECFM=FB,BLKSIZE=1500,LRECL=300),UNIT=DACLASS
```

Figure 32. Job Control Statements for Example 1

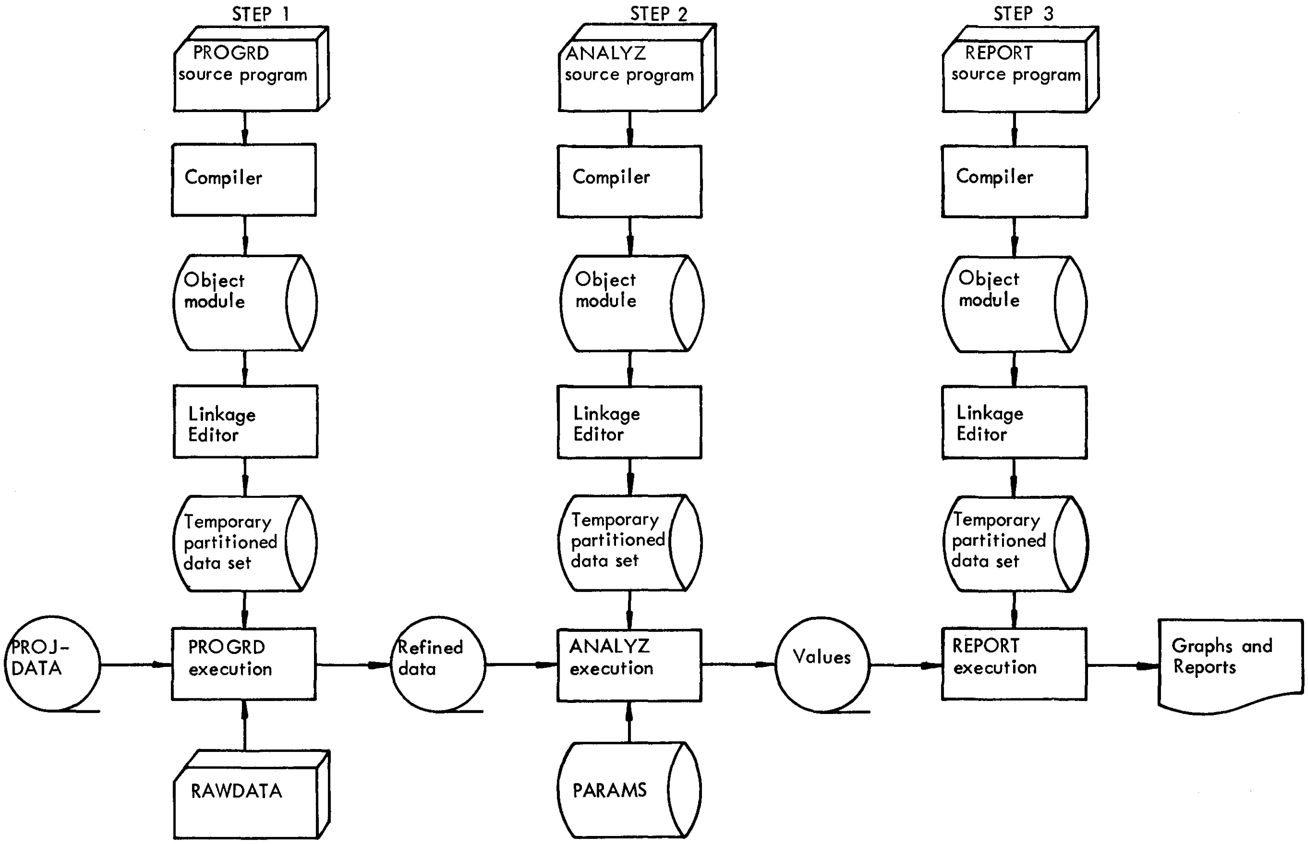


Figure 33. Basic I/O Flow for Example 2.
The data sets for information listings, ALGOL library routines, intermediate work and the execution time error routine are not shown above.

```
//TESTFIRE JOB ,JOHNSMITH,MSGLEVEL=1
//STEP1 EXEC ALGOFCLG
//ALGOL.SYSIN DD *
  SOURCE PROGRAM (PROGRD)
/*
//GO.ALGLDD11 DD DSNAME=PROJDATA,DISP=OLD
//GO.ALGLDD12 DD DSNAME=&REFDATA,DCB=(RECFM=F,BLKSIZE=400,LRECL=80),     *
//               DISP=(NEW,PASS),UNIT=TAPECLS,VOLUME=(RETAIN,SER=2107)
//GO.SYSIN DD *
  INPUT DATA (RAWDATA)
/*
//STEP2 EXEC ALGOFCLG
//ALGOL.SYSIN DD *
  SOURCE PROGRAM (ANALYZ)
/*
//LKED.SYSLMOD DD DSNAME=&GOSET(ANALYZ)
//GO.ALGLDD06 DD DSNAME=*.STEP1.ALGLDD12,DISP=OLD
//GO.ALGLDD07 DD DSNAME=PARAMS,DISP=OLD
//GO.ALGLDD03 DD DSNAME=&VALUES,DCB=(RECFM=F,BLKSIZE=204,LRECL=68),      *
//               DISP=(NEW,PASS),UNIT=TAPECLS,VOLUME=SER=2108
//STEP3 EXEC ALGOFCLG
//ALGOL.SYSIN DD *
  SOURCE PROGRAM (REPORT)
/*
//LKED.SYSLMOD DD DSNAME=&GOSET(REPORT)
//GO.ALGLDD14 DD DSNAME=*.STEP2.ALGLDD03,DISP=OLD
```

Figure 34. Job Control Statements for Example 2

Example 3: Executing Two Load Modules

Statement of problem (see Figure 35): Data on current weather conditions is to be read from cards and used by the program FILECR to create a new generation of a data set WEATHER, and also to print a report.

Then the new generation and the three immediately preceding generations of the WEATHER data set are to be used by the program FORCST to produce a printed weather forecast. The programs FILECR and FORCST are contained in a partitioned data set WTHRPR.

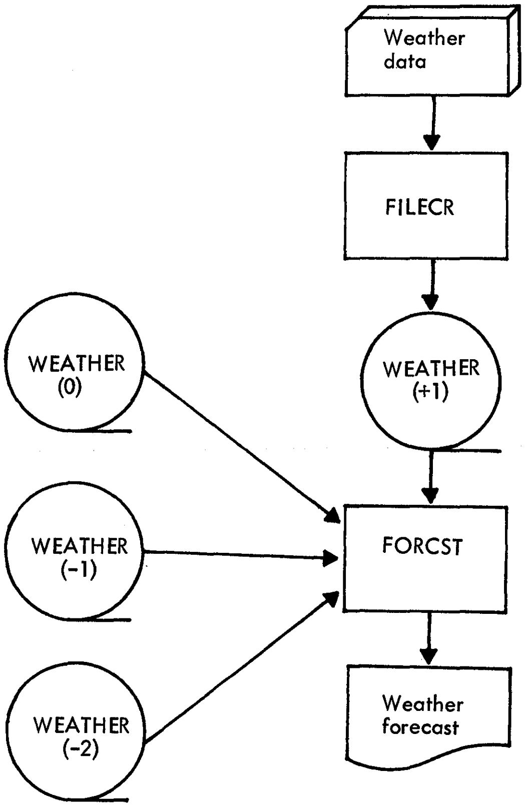


Figure 35. I/O Flow for Example 3

Explanation of coding: The job control statements used in Figure 36 specify that:

1. The job is to have control statement messages plus the relevant control statement printed on the normal system output device only if an error occurs

2. The partitioned data set WTHRPR is concatenated to the operating system library, SYS1.LINKLIB

3. The first job step executes the program FILECR

4. The output data set is

   - a new generation of the data set WEATHER.
   - to use the device class HYPERT.
   - to be written on volume 0012 which need not be mounted until the data set is opened, and is then to remain mounted for later use.
   - to be cataloged and have standard labels.
   - to be retained for 30 days.
   - to have fixed length records, 80 bytes long, and a maximum block size of 400 bytes.

5. The printed output is

   - to use the device class PRINTER.
   - to use a separate channel to the output data.

6. The input data is included in the input stream.

7. The second job step executes the program FORCST.

8. The input data sets are the last four generations of WEATHER, all of which are to be kept at the end of the job step.

9. The output data set is
   - to use the device class PRINTER.
   - to use a separate channel to the last two generations of WEATHER.

Example 4: Compiling and Linkage Editing an ALGOL Precompiled Procedure

Statement of problem: The ALGOL language procedure ADD is to be compiled, linkage edited and stored in load module form as a member on the partitioned data set PREPROC, for use in subsequent programs. An illustration of a program in which ADD is invoked is provided in Example 5.

Explanation of coding: The job control statements used in Figure 37 specify that:

1. The job is to have all control statements (plus control statement diagnostic messages if an error occurs) printed on the normal system output device.

2. The job step is to invoke the ALGOFCL cataloged procedure to compile and linkage-edit the source module, which is identified as an ALGOL precompiled procedure.

3. A new partitioned data set named PREPROC is to be allocated and cataloged; the procedure ADD is to be stored on the data set as a member; and a primary allocation of 30 tracks (plus a secondary allocation of 10 tracks, if needed) and a directory of

```
//WEATHRP JOB MSGLEVEL=0
//JOBLIB DD DSNAME=WTHRPR,DISP=(OLD,PASS)
//CREATE EXEC PGM=FILECR
//ALGLDD02 DD DSNAME=WEATHER(+1),DCB=(RECFM=F,BLKSIZE=400,LRECL=80),    *
//             VOLUME=(RETAIN,SER=0012),LABEL=(,SL,RETPD=0030),          *
//             UNIT=(HYPERT,,DEFER),DISP=(NEW,CATLG)
//ALGLDD01 DD UNIT=PRINTER,SEP=ALGLDD02
//SYSPRINT DD UNIT=PRINTER,SEP=ALGLDD02
//SYSIN DD *
  WEATHER DATA
/*
//FORECAST EXEC PGM=FORCST
//ALGLDD04 DD DSNAME=WEATHER(+1),DISP=OLD
//ALGLDD07 DD DSNAME=WEATHER(0),SEP=ALGLDD04,DISP=OLD
//ALGLDD08 DD DSNAME=WEATHER(-1),DISP=OLD
//ALGLDD09 DD DSNAME=WEATHER(-2),DISP=OLD
//ALGLDD01 DD UNIT=PRINTER,SEP=(ALGLDD04,ALGLDD07)
//SYSPRINT DD UNIT=PRINTER,SEP=(ALGLDD04,ALGLDD07)
```

Figure 36. Job Control Statements for Example 3

ten 256-byte records is to be assigned to the data set.

Example 5: Compiling, Linkage Editing and Executing an ALGOL Program which Invokes a Precompiled Procedure

Statement of problem: An ALGOL program in which the precompiled procedure ADD (Example 4) is invoked, is to be compiled, linkage edited and executed.

The job control statements in Figure 38 specify:

1. The job is to have all control statements (plus control statement diagnostic messages if an error occurs) printed on the normal system output device.
2. The partitioned data set PREPROC, containing the precompiled procedure ADD, is to be concatenated to the operating system library, SYS1.LINKLIB
3. The job step is to invoke the ALGOFCLG cataloged procedure to compile, linkage edit and execute the ALGOL source program.

```
//CODEPC JOB MSGLEVEL=1
//STEP EXEC ALGOFCL,PARM.ALGOL=PROCEDURE
//ALGOL.SYSIN DD *
  'PROCEDURE' ADD(A,B,C);
  'REAL' A,B,C;
    C:=A+B;
/*
//LKED.SYSLMOD DD DSNAME=PREPROC(ADD),DISP=(NEW,CATLG),UNIT=SYSDA,     *
//             SPACE=(TRK,(30,10,10)),VOLUME=SER=222222
```

Figure 37. Job Control Statements and Source Module for Example 4

```
//MAINPG JOB MSGLEVEL=1
//JOBLIB DD DSNAME=PREPROC,DISP=OLD
//STP1 EXEC ALGOFCG
//ALGOL.SYSIN DD *
  'BEGIN'
    'REAL' E,F,G;
    'PROCEDURE' ADD(A,B,C);
    'REAL' A,B,C;
      'CODE';
    E:=5.6;
    F:=-7.8;
    ADD(E,F,G);
    OUTREAL(1,G)
  'END'
/*
```

Figure 38. Job Control Statements and Source Module for Example 5

Appendix F: Diagnostic Messages

This section describes the messages and the appropriate responses to messages by the compiler, the linkage editor, and the ALGOL object program at execution time.

Compiler Messages

The following table describes the format and gives other pertinent information about ALGOL compiler messages.

Component Name	IEX
Program Producing Message	ALGOL compiler.
Audience and Where Produced	For programmer: SYSPRINT data set. For operator: console.
Message Format	IEXnnnI s nnnnn text nnn Message serial number. s Severity code: W Warning; the compiler internally modifies the program being compiled and continues processing; the modification may or may not correct the program, but it allows compilation to continue. S Serious; the compiler attempts to modify the program internally, including skipping or changing parts of it; generation of the object module is stopped, but syntax checking continues. T Compilation is terminated. nnnnn Semicolon number, right-adjusted and in decimal; if the error cannot be related directly to a point in the program, nnnnn is blank. text Message text.
Associated Publications	OS ALGOL Language, GC28-6615

IEX001I W nnnnn INVALID CHARACTER DELETED

Explanation: A character not recognized by the compiler has been deleted from the program.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX002I W nnnnn ILLEGAL PERIOD. PERIOD DELETED.

Explanation: The character period has been used wrongly and deleted from the program. It can be used only as a decimal point, or as part of a colon or semicolon.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX003I W nnnnn INVALID COLON AFTER (six characters). COLON DELETED.

Explanation: The character colon has been used wrongly and has been deleted from the program. It can be used only after a label, between subscript bounds, within a parameter delimiter or as part of an assign symbol.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX004I T nnnnn LETTER STRING TOO LONG

Explanation: A letter string used to supply explanatory information exceeds capacity limitations.

Programmer Response: Probable user error. Shorten the letter string and recompile. If the problem recurs, do the following before calling IBM:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX005I S nnnnn IDENTIFIER BEGINS WITH INVALID CHARACTER. IDENTIFIER DELETED.

Explanation: An identifier has been deleted because it does not begin with an alphabetic character.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX006I T nnnnn LABEL CONTAINS TOO MANY CHARACTERS

Explanation: A label identifier has been used whose length exceeds capacity limitations.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX007I W nnnnn LABEL BEGINNING WITH (up to six characters) CONTAINS INVALID CHARACTER. COLON DELETED.

Explanation: A label has been deleted because it contains a character of other than alphameric type.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX008I W nnnnn LABEL BEGINS WITH INVALID CHARACTER. COLON DELETED.

Explanation: A label has been deleted because it does not begin with an alphabetic character.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX010I S nnnnn SPECIFICATION PART OF PROCEDURE (identifier) INCOMPLETE.

Explanation: Not all of the formal parameters used in a procedure have been specified.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX011I S nnnnn PROGRAM STARTS WITH ILLEGAL DELIMITER.

Explanation: If the compiler option PROGRAM(PG) has been specified, the source text must start with 'BEGIN'. If the option PROCEDURE(PC) has been specified the source text must start with one of the following:

1. 'PROCEDURE'
2. 'REAL''PROCEDURE'
3. 'INTEGER''PROCEDURE'
4. 'BOOLEAN''PROCEDURE'

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX012I W nnnnn TWO APOSTROPHES AFTER (six characters). FIRST APOSTROPHE DELETED.

Explanation: In this context, two apostrophes cannot be used together so one has been deleted.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX013I W nnnnn APOSTROPHE ASSUMED AFTER DELIMITER BEGINNING WITH (up to six characters).

Explanation: All delimiters involving words must begin and end with apostrophes. One has been left out of the program and has been inserted by the compiler.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX014I S nnnnn DELIMITER BEGINNING WITH (up to six characters) INVALID. FIRST APOSTROPHE DELETED.

Explanation: An invalid sequence of characters has been used after an apostrophe which apparently started a delimiter. The apostrophe is therefore deleted to remove the delimiter status from the characters but still include them in the program.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX015I W nnnnn MISSING SEMICOLON AFTER 'CODE'. SEMICOLON INSERTED.

Explanation: Self-explanatory.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX016I S nnnnn IDENTIFIER BEGINNING WITH (up to six characters) CONTAINS INVALID CHARACTER. IDENTIFIER DELETED.

Explanation: A character other than an alphameric type has been used in an identifier and so the identifier has been deleted.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX017I S nnnnn MORE THAN 65535 SEMICOLONS. SEMICOLON COUNTER RESET TO ZERO.

Explanation: Number of semicolons used exceeds capacity limitations. Duplicate numbers are allocated.

Programmer Response: Probable user error. Make precompiled procedures of suitable parts of source program. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX018I W nnnnn DELIMITER 'COMMENT' IN ILLEGAL POSITION

Explanation: 'COMMENT' has not been placed after a 'BEGIN' or a semicolon. Compilation continues normally.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX020I T nnnnn BLOCKS, COMPOUND STATEMENTS, FOR STATEMENTS, AND PROCEDURE DECLARATIONS NESTED TO TOO MANY LEVELS.

Explanation: Structure of program causes it to exceed capacity limitations.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX021I S nnnnn DECLARATOR (declarator) IN ILLEGAL POSITION.

Explanation: A declarator must come between either 'BEGIN' and the first statement of a block, or 'PROCEDURE' and the procedure body.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX022I T nnnnn MORE THAN 255 PROGRAM BLOCKS.

Explanation: Number of program blocks used exceeds capacity limitations.

Programmer Response: Probable user error. Make precompiled procedures of suitable parts of source program. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX023I S nnnnn STRING POOL OVERFLOW.

Explanation: Total length of strings used exceeds capacity limitations.

Programmer Response: Probable user error. Make precompiled procedures of suitable parts of source program. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX024I S nnnnn DELIMITER 'CODE' IN ILLEGAL POSITION. 'CODE' DELETED.

Explanation: 'CODE' has not been placed immediately after a procedure heading so it has been deleted.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX025I S nnnnn SPECIFIER 'STRING' OR 'LABEL' IN ILLEGAL POSITION. SPECIFICATION DELETED.

Explanation: 'STRING' and 'LABEL' have been used outside procedure heading, so they have been deleted.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX026I S nnnnn PARAMETER (identifier) MULTIPLY SPECIFIED. FIRST SPECIFICATION USED.

Explanation: Self-explanatory.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX027I W nnnnn PARAMETER (identifier) MISSING FROM FORMAL PARAMETER LIST. SPECIFICATION IGNORED.

Explanation: A parameter has been specified in a procedure heading which does not exist in the formal parameter list, so it has been ignored.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX028I S nnnnn DELIMITER 'VALUE' IN ILLEGAL POSITION. VALUE PART DELETED.

Explanation: 'VALUE' has been placed outside a procedure heading so the value part has been deleted.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX029I W nnnnn SPECIFICATION PART PRECEDES VALUE PART.

Explanation: The specification part in a procedure heading has been incorrectly placed before the value part.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX030I W nnnnn PARAMETER (identifier) REPEATED IN VALUE PART.

Explanation: A parameter has been included in the value part of a procedure heading more than once.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX031I W nnnnn LEFT PARENTHESIS NOT FOLLOWED BY / AFTER ARRAY IDENTIFIER (identifier). SUBSCRIPT BRACKET ASSUMED.

Explanation: The subscript bounds after an array identifier have been preceded by a left parenthesis instead of a subscript bracket.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX032I S nnnnn MISSING RIGHT PARENTHESIS IN BOUND PAIR LIST OF ARRAY (identifier). DECLARATION DELETED.

Explanation: The right parenthesis has been omitted in the list of subscript bounds for an array identifier, so the declaration is deleted.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX033I T nnnnn MORE THAN 16 DIMENSIONS OR COMPONENTS IN DECLARATION OF (identifier).

Explanation: The number of dimensions or components used with an array or switch identifier exceeds the maximum allowed.

Programmer Response: Probable user error. Rearrange the structure of the source program to avoid the capacity limitation. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX034I S nnnnn ARRAY SEGMENT (identifier) NOT FOLLOWED BY SEMICOLON OR COMMA. CHARACTERS TO NEXT SEMICOLON DELETED.

Explanation: An array segment must be followed by a semicolon if it is the only or last segment of an array declaration; or a comma if it is followed by another segment.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX035I W nnnnn ILLEGAL PERIOD IN ARRAY OR SWITCH LIST. PERIOD DELETED.

Explanation: A period has been used wrongly in an array or switch list and deleted from the program. A period can be used only as a decimal point, or as part of a colon or semicolon.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX036I T nnnnn MORE THAN 15 PARAMETERS IN DECLARATION OF (identifier).

Explanation: The number of formal parameters specified for a procedure exceeds the maximum allowed.

Programmer Response: Probable user error. Rearrange the structure of the source program to avoid the capacity limitation. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX037I S nnnnn SEMICOLON MISSING AFTER FORMAL PARAMETER LIST OF (identifier). CHARACTERS TO NEXT SEMICOLON DELETED.

Explanation: The formal parameter list of a procedure must be followed by a semicolon.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings

available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX038I T nnnnn TOO MANY IDENTIFIERS DECLARED IN A BLOCK.

Explanation: Number of identifiers declared in a block exceeds capacity limitations.

Programmer Response: Probable user error. Rearrange the structure of the source program to avoid the capacity limitation. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX039I S nnnnn nnn MISSING 'END' BRACKETS. OPEN BLOCKS, COMPOUND STATEMENTS, FOR STATEMENTS, AND PROCEDURE DECLARATIONS CLOSED.

Explanation: Syntax of ALGOL requires that a program contains the same number of 'BEGIN's and 'END's. The number of 'END's specified by nnn have been omitted in this case so any open block and statements are closed.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.case so any open blocks and statements are closed.

IEX041I T nnnnn MORE THAN 255 FOR STATEMENTS.

Explanation: Number of for statements used in a program exceeds capacity limitations.

Programmer Response: Probable user error. Make precompiled procedures of suitable parts of source program. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX042I W nnnnn 'BEGIN' PRECEDES PRECOMPILED PROCEDURE. 'BEGIN' DELETED.

Explanation: A precompiled procedure has been specified so a 'BEGIN' is not required.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX043I S nnnnn EQUAL NUMBER OF 'BEGIN' AND 'END' BRACKETS FOUND. REMAINING PART OF PROGRAM IGNORED.

Explanation: The compiler assumes it has reached the end of the program when the number of 'END' brackets equals the number of 'BEGIN' brackets.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX044I T nnnnn NO SOURCE PROGRAM FOUND.

Explanation: For example, there has been an incorrect card code specification.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX045I S IDENTIFIER (identifier) MULTIPLY DECLARED. LAST DECLARATION USED.

Explanation: An identifier has been declared more than once in a program block heading. The last declaration is taken to be the one required.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX047I S ILLEGAL CALL BY VALUE OF IDENTIFIER (identifier).

Explanation: A procedure, switch or string has been wrongly called by value.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX080I S nnnnn OPERAND BEGINNING WITH (up to six characters) IS SYNTACTICALLY INCORRECT.

Explanation: Invalid characters have been used in the operand. If the six characters are all periods, this may indicate the internal representation of a string or logical value.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX081I S nnnnn IDENTIFIER (identifier) NOT DECLARED.

Explanation: An identifier has been used which is not declared in a block or procedure heading.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX082I S nnnnn REAL CONSTANT BEGINNING WITH (up to twelve characters) OUT OF RANGE.

Explanation: A real constant has been assigned a value which is outside capacity limitations.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX083I W nnnnn INTEGER BEGINNING WITH (up to twelve characters) OUT OF RANGE. INTEGER CONSTANT CONVERTED TO REAL.

Explanation: An integer constant has been assigned a value which is outside storage capacity limitations, so it has been converted to a real constant.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX084I W nnnnn PRECISICN CF REAL CONSTANT BEGINNING WITH (up to twelve characters) EXCEEDS INTERNALLY HANDLED PRECISICN. CONSTANT TRUNCATED.

Explanation: A real constant has exceeded capacity limitations regarding precision and has been truncated.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX085I S nnnnn ILLEGAL USE OF LABEL (label).

Explanation: A label defined in a for statement has been used in a gcto statement outside the for statement, or the label occurs in a syntactically illegal position.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX086I S nnnnn TCO MANY CONSTANTS.

Explanation: Number of constants used exceeds capacity limitations.

Programmer Response: Probable user error. Make precompiled procedures of suitable parts of source program. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX087I W nnnnn FULL OPTIMIZATION NOT POSSIBLE DUE TO INTERNAL OVERFLOW.

Explanation: Main storage capacity available prevents for statement optimization by the compiler after the overflow occurs.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX088I W nnnnn IDENTIFIER (identifier) IN BOUND EXPRESSION DECLARED IN SAME PROGRAM BLOCK AS ARRAY. DECLARATION IN SURROUNDING BLOCK SEARCHED FOR.

Explanation: A bound expression can depend only on variables and procedures which are non-local to the block for which the array declaration is valid, because local variables do not have values before entering the statements of the block.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX089I W nnnnn 'GOTO' (identifier) INVALID OUTSIDE FOR STATEMENT CONTAINING THIS LABEL.

Explanation: A switch may have been misused, since a label has been found in a switch declaration outside a for statement containing a definition of the same label.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX160I S nnnnn SEQUENCE (operator) (operator) NOT ALLOWED.

Explanation: In this context, this sequence is not allowed.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX161I S nnnnn SEQUENCE (operator) OPERAND (operator) NOT ALLOWED.

Explanation: In this context, this sequence is not allowed.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX162I S nnnnn OPERAND MISSING BETWEEN (operator) AND (operator).

Explanation: In this context, there must be an operand between two operators.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX163I S nnnnn OPERAND FOLLOWING (operator) MUST BE OF ARITHMETICAL TYPE.

Explanation: An arithmetical operand must follow an arithmetical operator.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX164I S nnnnn NO OPERAND ALLOWED BETWEEN (operator) AND (operator).

Explanation: In this context, no operand is allowed between the two operators.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX165I S nnnnn EXPRESSIONS BEFORE AND AFTER 'ELSE' NOT COMPATIBLE.

Explanation: For example, if an arithmetical expression is specified before 'ELSE', then an arithmetical expression must be specified after.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX166I S nnnnn DECLARATOR IN ILLEGAL POSITION.

Explanation: A declaration has occurred outside the block heading, or, for instance, a label precedes the declaration.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX168I S nnnnn OPERAND PRECEDING (operator) CANNOT POSSESS VALUE.

Explanation: Only quantities that can possess a value can be used in expression. For example, not standard I/O or non-type procedure identifier.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX169I S nnnnn LABEL FOLLOWING (operator) ILLEGAL.

Explanation: In this context, a label is not allowed due, for example, to a semicolon being missing.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX172I S nnnnn DIFFERENT TYPES IN LEFT PART LIST.

Explanation: The identifiers in a left part list must be of similar type.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX173I T nnnnn COMPILATION UNSUCCESSFUL DUE TO COMPILER OR MACHINE ERROR.

Explanation: Self-explanatory.

Programmer Response: Recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX174I S nnnnn PARAMETERS NOT ALLOWED FOR TYPE PROCEDURE CALLED BY VALUE.

Explanation: A type procedure called by value must have an empty parameter part.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX175I S nnnnn OPERAND FOLLOWING (operator) MUST BE LABEL OR SWITCH.

Explanation: For example, 'GOTO' must be followed by a designational expression.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX176I S nnnnn OPERAND MISSING BEFORE (operator).

Explanation: In this context, the operator must be preceded by an operand.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX177I S nnnnn OPERAND NOT ALLOWED BEFORE (operator).

Explanation: In this context, no operand is allowed before the operator.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX178I S nnnnn ILLEGAL OPERAND IN EXPRESSION BEFORE OR AFTER 'ELSE'.

Explanation: For example, only arithmetical operands may be used in an arithmetical expression.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX179I S nnnnn NUMBER OF SUBSCRIPT EXPRESSIONS DIFFERS FROM DIMENSION IN ARRAY DECLARATION FOR VARIABLE.

Explanation: A subscript list must contain the same number of subscript expressions as the dimension in the corresponding array declaration.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX180I S nnnnn INVALID SWITCH DESIGNATCR.

Explanation: More than one subscript expression in switch designator.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX181I S nnnnn SWITCH DESIGNATOR IN ILLEGAL POSITION.

Explanation: A switch designator must follow only 'THEN', 'ELSE', 'GOTC', := or ,.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX182I S nnnnn OPERAND FOLLOWING (operator) MUST BE BOOLEAN.

Explanation: A non-Boolean operand has been specified where a Boolean one was required.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX183I S nnnnn OPERAND PRECEDING (operator) MUST BE A PROCEDURE IDENTIFIER.

Explanation: A non-procedure identifier has been specified where a procedure one was required.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX184I S nnnnn OPERAND PRECEDING (operatcr) MUST BE AN ARRAY OR SWITCH IDENTIFIER.

Explanation: A non-array or nonswitch identifier has been specified where an array or switch one was required.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX185I S nnnnn REAL OPERAND PRECEDING (operator) NOT ALLOWED FOR INTEGER CIVISICN.

Explanation: A real operand has been specified for an integer division.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX186I T nnnnn SYNTACTICAL STRUCTURE TCO COMPLICATED. INTERNAL OVERFICW.

Explanation: The syntactical structure of the program has caused an internal overflow in the compiler. A larger main storage size is required.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings

available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX187I S nnnnn INCORRECT NUMBER OF ACTUAL PARAMETERS.

Explanation: The number of actual parameters does not correspond to the number of formal parameters in a procedure.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX188I S nnnnn INVALID ACTUAL PARAMETER FOR STANDARD PROCEDURE. DSN= (number).

Explanation: An actual parameter has been specified incorrectly in a standard procedure. Either semicolon number or data set number is given. In the case where the data set number is given instead of the semicolon number, the error is due to SYSACT8 having been specified for the data set when SYSACT4, SYSACT13 or an input operation has been specified also. Such a combination is invalid.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX189I S nnnnn DATA SET NUMBER OR FUNCTION OF SYSACT OUT OF ALLOWED RANGE.

Explanation: Data set numbers are 0 - 15. SYSACT functions are 1 - 15.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX190I S nnnnn ASSIGNMENT NOT POSSIBLE.

Explanation: Only variable allowed in for clause. Only variable or type procedure identifier allowed in left part list.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX191I S nnnnn NO OPERAND ALLOWED BETWEEN) AND (operator).

Explanation: When a right parenthesis is used it must be followed by an apostrophe, a semicolon, an arithmetical operator, a comma, or another right parenthesis.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX192I S nnnnn INVALID RIGHT PART IN ASSIGNMENT STATEMENT.

Explanation: The right part must be either an arithmetic or a Boolean expression.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX193I S nnnnn INCOMPATIBLE TYPES IN ASSIGNMENT STATEMENT.

Explanation: Value assigned to right part does not correspond to type of left part list in assignment statement.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX194I S nnnnn (operator) NOT ALLOWED.

Explanation: In this context, the operator is not allowed.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX195I S nnnnn SEQUENCE OPERAND (operator) NOT ALLOWED.

Explanation: In this context, this sequence is not allowed.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX196I S nnnnn ARRAY IDENTIFIER PRECEDING (operator) NOT ALLOWED.

Explanation: In this context, an array identifier is not allowed.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX200I W nnnnn OPTION PARAMETER (parameter) INVALID. PARAMETER IGNORED.

Explanation: An invalid option has been specified in the PARM parameter and ignored by the compiler.

Programmer Response: Probable user error. Make sure all compiler options specified are correct and recompile if necessary. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX201I T nnnnn DD CARD FOR (ddname) INCORRECT OR MISSING.

Explanation: During an ALGOL compilation, the DD statement for the data set named ddname was incorrect or missing. ddname can be SYSIN, SYSPRINT, or SYSUT1, 2, or 3.

This message appears on the console if ddname is SYSPRINT.

Programmer Response: Probable user error. Make sure the DD statement is correct or supply the missing one. Recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX202I W nnnnn DD CARD FOR SYSLIN INCORRECT OR MISSING. OPTION NOLOAD ASSUMED.

Explanation: The SYSLIN data set has been specified incorrectly or not at all when the LOAD option is specified, so an object module is not generated.

Programmer Response: Probable user error. Make sure the DD statement is correct or supply the missing one. Recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX203I W nnnnn DD CARD FOR SYSPUNCH INCORRECT OR MISSING. OPTION NODECK ASSUMED.

Explanation: The SYSPUNCH data set has been specified incorrectly or not at all when the DECK option is specified, so an object deck is not punched.

Programmer Response: Probable user error. Make sure the DD statement is correct or supply the missing one. Recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX204I T nnnnn BLOCKSIZE SPECIFIED FOR SYSIN INCORRECT.

Explanation: The blocksize specified for SYSIN does not correspond to the actual blocksize.

Programmer Response: Probable user error. Make sure the DD statement is correct and recompile. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX205I W nnnnn BLOCKSIZE SPECIFIED FOR (ddname) INCORRECT. UNBLOCKED OUTPUT ASSUMED.

Explanation: One of the output data sets has had an incorrect blocksize specified so unblocked output is generated.

Programmer Response: Probable user error. Make sure the DD statement is correct or supply the missing one. Recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX206I W nnnnn TOO MANY OPTION PARAMETER ERRORS. SUBSEQUENT PARAMETERS IGNORED.

Explanation: Too many incorrect parameters have been specified in the PARM parameter so the rest are ignored.

Programmer Response: Probable user error. Make sure all compiler options specified are correct and recompile if necessary. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX207I W nnnnn POSSIBLE ERROR IN DD NAMES PARAMETER.

Explanation: An incorrect ddname may have been specified in the DD statement.

Programmer Response: Probable user error. Make sure the DD statement is correct or supply the missing one. Recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX208I W nnnnn SIZE PARAMETER INVALID. SIZE 45056 assumed.

Explanation: The main storage size specified as being available to the compiler is less than the minimum required, so the minimum value is assumed.

Programmer Response: Probable user error. Make sure all compiler options specified are correct and recompile if necessary. If the problem recurs, do the following
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX209I T nnnnn COMPILATION UNSUCCESSFUL DUE TO PROGRAM INTERRUPT. PSW (hexadecimal digits).

Explanation: A program interrupt has occurred causing termination of the job step. The program status word when the error occurred is given.

Programmer Response: Recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX210I T nnnnn UNRECOVERABLE I/O ERROR ON DATA SET (ddname).

Explanation: During an ALGOL compilation, an uncorrectable input/output error occurred in using the data set named ddname.

This message appears on the console if ddname is SYSPRINT.

Programmer Response: Make sure that the DD statement is correct and recompile. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX211I T nnnnn PROGRAM INTERRUPT IN ERROR MESSAGE EDITING ROUTINE. PSW (hexadecimal digits).

Explanation: A program interrupt has occurred in the error message editing routine, ending the job.

Programmer Response: Recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX212I T nnnnn TOO MANY ERRORS.

Explanation: The total length of the error message patterns produced exceeds capacity limitations.

Programmer Response: Probable user error. Make sure the source code is correct and recompile if necessary. If the problem recurs, do the following before calling IBM for programming support:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX213I T nnnnn INTERNAL OVERFLOW OF IDENTIFIER TABLE.

Explanation: The number of identifiers declared exceeds capacity limitations.

Programmer Response: Probable user error. Rearrange the structure of the source program to avoid the capacity limitation. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX214I S nnnnn DATA STORAGE AREA EXCEEDED. PROGRAM BLOCK NO. (number).

Explanation: The data storage area required by the program block specified exceeds 4096 bytes.

Programmer Response: Probable user error. Rearrange the structure of the source program to avoid the capacity limitation. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:

- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX215I T nnnnn SOURCE PROGRAM TOO LONG.

Explanation: The source program exceeds capacity limitations.

Programmer Response: Probable user error. Make precompiled procedures of suitable parts of source program. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IEX216I S nnnnn TOO MANY LABELS. LABEL NUMBER RESET.

Explanation: The total number of labels used exceeds capacity limitations, so duplicated numbers are allocated.

Programmer Response: Probable user error. Make precompiled procedures of suitable parts of source program. Make sure the source code is correct and recompile. If the problem recurs, do the following before calling IBM:
- Have source and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

Linkage Editor and Loader Messages

The diagnostic messages produced by the linkage editor and by the loader are listed in the publication _OS Loader and Linkage Editor_.

The diagnostic message consists of one or more printed lines and contains:

- A message key, consisting of the letters IEW, a three digit decimal number identifying the message, and a final digit, 1, 2, 3 or 4, indicating the severity code (see below). Linkage editor message keys read IEW0---; loader message keys read IEW1---.
- The message text describing the error. For severity code 1 the message is preceded by 'WARNING'. For all other severity codes the message is preceded by 'ERROR'.

The severity codes have the following meaning:

1 indicates a condition that may couse an error during execution of the load module. A module map or cross-reference table is produced if it was required by the programmer. The output load module is marked as executable.

2 Indicates an error that could make execution of the load module impossible. Processing continues. When possible, a module map or cross-reference table is produced if it was required. The load module is marked as not executable unless the IET option has been specified.

3 indicates an error that will make execution of the load module impossible. Processing continues. If possible a module map or cross-reference table is produced if it was required. The load module is marked as not executable.

4 indicates an error condition from which no recovery is possible. Processing terminates. The only output is diagnostic messages.

Execution Time Messages

The following table describes the format and gives other pertinent information about the ALGOL object program messages

Component Name	IHI
Program Producing Message	Object program originally coded in ALGOL language.
Audience and Where Produced	For programmer: SYSPRINT data set. For operator: console.
Message Format	IHInnnI SC nnnnn text nnn Message serial number. nnnnn Semicolon number, right-adjusted, and in decimal. text Message text. Where appropriate, begin with: DSN=nn or DSN=ddname Indicates the number (nn) or name (ddname) of the data set involved in the error. PSW=nnnn nnnn Contents of the program status word (PSW) held by the system when the error occurred. ** Indicates that the program does not correspond to the parameters specified in the job control statements.
Associated Publications	_OS ALGOL Language_, GC28-6615

IHI000I SC=nnnnn DATA SET NUMBER OUT OF RANGE

Explanation: A data set number must be in the range 0 to 15.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI001I SC=nnnnn DSN=nn. REAL NUMBER TO BE CONVERTED OUT OF INTEGER RANGE

Explanation: A real number has been included which exceeds capacity limitations when converted to integer. This message applies for input/output operations.

Programmer Response: Probable user error. Make sure the source code is correct before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI002I SC=nnnnn DSN=nn. INCOMPATIBLE ACTIONS ON DATA SET

Explanation: The I/O procedure requested is not defined for this data set. For example, procedure SYSACT8 specifying data set number 0 is not allowed.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI003I SC=nnnnn DSN=nn. INPUT BEYOND LAST OUTPUT

Explanation: Before reading data which has just been written on the same data set, backward repositioning must be specified.

Probable user error. Make sure the source code is correct. Modify the source to avoid the capacity limition and rerun the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI004I SC=nnnnn TOO MANY REPOSITIONINGS IN DATA SETS. INTERNAL OVERFLOW

Explanation: Too many repositionings have caused an internal overflow of the Note Table.

Programmer Response: Probable user error. Make sure the source code is correct. Modify the source to avoid the capacity limitation and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement

IHI005I SC=nnnnn DSN=nn. INPUT REQUEST BEYOND END OF DATA SET

Explanation: Input has been requested to start beyond the end of the data set. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI006I SC=nnnnn DSN=nn. EXPONENT PART OF INPUT NUMBER CONSISTS OF MORE THAN TWO SIGNIFICANT DIGITS

Explanation: The length of the exponent part of an input number exceeds capacity limitations.

Programmer Response: Probable user error. Make sure the source code is correct. Modify the input data to avoid the capacity limitation and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI007I SC=nnnnn DS=nn. **NO CONTROL CHARACTER SPECIFIED IN RECORD FORMAT OF DATA SET. SPLITTING INTO SECTIONS IMPOSSIBLE

Explanation: A control character is required to define printing format.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement. .

IHI008I SC=nnnnn DSN=nn. SOURCE IN PROCEDURE OUTSYMBOL DOES NOT MATCH STRING

Explanation: The symbol specified by the third parameter of the OUTSYMBOL procedure does not correspond to any symbol in the string specified by the second parameter.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI009I SC=nnnnn DSN=nn. UNDEFINED FUNCTION NUMBER IN SYSACT PROCEDURE

Explanation: A function number has not been defined for a SYSACT procedure. The function number range is 1 to 15.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI010I SC=nnnnn DSN=nn. DATA SET CLOSED

Explanation: The data set is closed but a SYSACT procedure has been specified which requires it to be open.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI011I SC=nnnnn DSN=nn. DATA SET OPEN

Explanation: The data set is open but a SYSACT procedure has been specified which requires it to be closed.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI012I SC=nnnnn DSN=nn. QUANTITY IN SYSACT PROCEDURE MUST BE VARIABLE

Explanation: The third parameter of the SYSACT procedure must be a variable.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI013I SC=nnnnn DSN=nn. QUANTITY IN SYSACT PROCEDURE OUT OF RANGE

Explanation: The variable specified in the third parameter of the SYSACT procedure exceeds capacity limitations.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI014I SC=nnnnn DSN=nn. BACKWARD REPOSITIONING NOT DEFINED

Explanation: Backward repositioning is defined using SYSACT 13.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and

associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI015I SC=nnnnn UPPER BOUND LESS THAN LOWER BOUND IN ARRAY DECLARATION

Explanation: The upper subscript bound specified in an array declaration must not be less than the lower subscript bound.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI016I SC=nnnnn VALUE OF SUBSCRIPT EXPRESSION NOT WITHIN DECLARED BOUNDS

Explanation: This error is detected only when the subscripted variable address falls outside the area reserved by the compiler for the array identifier.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI017I SC=nnnnn ENDLESS LOOP IN FOR STATEMENT

Explanation: The expressions used in the for statement result in an endless loop.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI018I SC=nnnnn MAIN STORAGE REQUESTED NOT AVAILABLE

Explanation: The storage space required by an array exceeds capacity available.

Programmer Response: Make sure the source code is correct. Either specify a larger partition or region or modify the source to avoid the capacity limitation and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI019I SC=nnnnn UNEQUAL NUMBER OF DIMENSIONS FOR ACTUAL AND FORMAL PARAMETER

Explanation: An array identifier being used as a parameter in a procedure has had a different number of dimensions assigned in the formal and actual positions.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement. PARAMETER OF DIFFERENT TYPE OR KIND

IHI020I SC=nnnnn ACTUAL AND CORRESPONDING FORMAL PARAMETER OF DIFFERENT TYPE OR KIND.

Explanation: An actual parameter has been assigned which does not have the type or kind declared for the corresponding formal parameter.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI021I SC=nnnnn UNEQUAL NUMBER OF PARAMETERS IN PROCEDURE DECLARATION AND PROCEDURE STATEMENT/FUNCTION DESIGNATOR

Explanation: Either not all, or more than, the formal parameters used in a procedure have been assigned in a procedure call.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI022I SC=nnnnn ASSIGNMENT TO A FORMAL PARAMETER NOT POSSIBLE

Explanation: A value cannot be assigned to an expression used in a standard input procedure, assignment statement, or for clause.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI023I SC=nnnnn ARGUMENT OF SQRT LESS THAN ZERO

Explanation: The ALGOL library SQRT routine cannot handle arguments with a value less than zero.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Have source, including source for precompiled procedures, input data and associated listings available.

IHI024I SC=nnnnn ARGUMENT OF EXP GREATER THAN 174,673

Explanation: The argument of EXP exceeds capacity limitations.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI025I SC=nnnnn ARGUMENT OF LN NOT GREATER THAN ZERO

Explanation: A number not greater than zero cannot have a natural logarithm.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI026I SC=nnnnn ABS VALUE OF ARGUMENT OF SIN OR COS NOT LESS THAN PI*2**18

Explanation: The argument exceeds capacity limitations for a short precision real value.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI027I SC=nnnnn ABS VALUE OF ARGUMENT OF SIN OR COS NOT LESS THAN PI*2**50

Explanation: The argument exceeds capacity limitations for a long precision real value.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI028I SC=nnnnn PSW=xxxxxxxx xxxxxxxx. FIXED POINT OVERFLOW INTERRUPT

Explanation: An interrupt has occurred due to an overflow of a fixed point number.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI029I SC=nnnnn PSW=xxxxxxxx xxxxxxxx. FLOATING POINT EXPONENT OVERFLOW INTERRUPT

Explanation: An interrupt has occurred due to an overflow of a floating point exponent.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI030I SC=nnnnn PSW=xxxxxxxx xxxxxxxx. DIVISION BY ZERO. FIXED POINT

Explanation: An attempt has been made to divide a fixed point number by zero.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI031I SC=nnnnn PSW=xxxxxxxx xxxxxxxx. DIVISICN BY ZERO. FLOATING POINT

Explanation: An attempt has been made to divide a floating point number by zero.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI032I SC=nnnnn DSN=nn. UNRECOVERABLE I/O ERRCR

Explanation: During execution of an object program originally written in the ALGOL language, an uncorrectable input/output error occurred in using the data set indicated by DSN=nn.

This message appears on the console if the data set is SYSPRINT.

Programmer Response: Make sure that the DD statement and source are correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.

IHI033I SC=nnnnn PSW=xxxxxxxx xxxxxxxx. PROGRAM INTERRUPT

Explanation: A program interrupt has occurred.

Programmer Response: Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI034I SC=nnnnn VALUE OF SWITCH DESIGNATOR NCT DEFINED IN DECLARATION CF SWITCH

Explanation: The designational expressions in the switch list of a switch declaration must define the values of all the corresponding switch designators.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI035I SC=nnnnn BASE NOT GREATER THAN ZERC

Explanation: Exponentiation is not defined in this case, because the base is zero or negative.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI036I SC=nnnnn TOO MANY NESTED BLOCKS ANC CALLS OF PROCEDURES, SWITCHES, AND PARAMETERS. INTERNAL OVERFLOW

Explanation: Structure of program causes it to exceed the internal capacity limitations.

Programmer Response: Probable user error. Make sure the source code is correct. Modify the source to avoid the capacity limitiation and run the job again. If the problem recurs, do the following calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI037I SC=nnnnn DSN=nn. **BLOCKSIZE NOT A MULTIPLE OF LOGICAL RECORD LENGTH

Explanation: Blocksize must be an exact multiple of logical record length.

Programmer Response: Make sure that the DD statement and source are correct and run the job again. If the problem recurs, do the following before calling IBM for probramming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.

IHI038I SC=nnnnn DSN=nn TOO LONG RECORD

Explanation: Record is longer than specified.

Programmer Response: Make sure that the DD statement and source are correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL (1,1) was specified in the JOB statement.
- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.

IHI039I SC=nnnnn GET/PUT IDENTIFICATION OUT OF RANGE

Explanation: The identification number specified for a GET/PUT operation is out of range.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that the DUMP option was apecified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

IHI040I SC=nnnnn REAL NUMBER TO BE CONVERTED OUT OF INTEGER RANGE

Explanation: A real number has been included which exceeds capacity limitations when converted to integer. This message applies to internal operations.

Programmer Response: Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB sstatement.
- Have source including source for precompiled procedures input data, and associated listings available.
- Make sure that the DUMP option was specified.

IHI041I SC=nnnnn DSN=nn. DD CARD INCORRECT OR MISSING

Explanation: During execution of an object program originally written in the ALGOL language, the DD statement for the data set indicated by DSN=nn was incorrect or missing.

This message appears on the console if the data set is SYSPRINT.

Operator Response: Correct the SYSPRINT DD statement, or supply the missing one. Then execute the job step again.

Programmer Response: Make sure that the DD statement is correct or supply the missing one. Execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Have source, including source for precompiled procedures input data, and associated listings available.

IHI042I SC=nnnnn INVALID OPTION PARAMETER

Explanation: An invalid option parameter has been specified in the PARM parameter.

Programmer Response: Make sure all options specified are correct and execute the job step again. If the problem recurs, do the following before calling IBM for programming support:
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.
- Have the source and associated listings available.

IHI043I SC=nnnnn ILLEGAL CALL OF GET/PUT OR LIST PROCEDURE

Explanation: Recursive calls of GET/PUT or list procedures are not allowed.

Programmer Response: Probable user error. Make sure the source code is correct and run the job again. If the problem recurs, do the following before calling IBM for programming support:

- Make sure that the DUMP option was specified.
- Have source, including source for precompiled procedures, input data, and associated listings available.
- Make sure that MSGLEVEL=(1,1) was specified in the JOB statement.

APPENDIX G: INDEPENDENT COMPONENT RELEASE (ICR)

DESCRIPTION OF OS ALGOL F INDEPENDENT COMPONENT RELEASE

The Independent Component Release (ICR) is distributed on a non-labeled, 9-track, 800 bpi, reel of magnetic tape (BLKSIZE=3440).

The Distribution Tape Reel (DTR) contains the job DTRALGOL, which consists of 8 steps, STEP1 to STEP8, as described below.

- STEP1 link-edits ALGOL library modules into SYS1.ALGLIB.
- STEP2 link-edits ALGOL compiler modules into SYS1.LINKLIB.
- STEP3 link-edits ALGOL message-editing modules into SYS1.LINKLIB.
- STEP4 places into a PDS the macro DTRALGOL, which is used in STEP6 to specify the ALGOL compiler options.
- STEP5 adds to SYS1.PROCLIB a member, OPTIONS, which contains data on default options.
- STEP6 assembles OPTIONS from STEP5, using the macro DTRALGOL from STEP4, and receives a CSECT of IEX00 (IEX00001) with the compiler options specified.
- STEP7 link-edits IEX00 into SYS1.LINKLIB.
- STEP8 adds the ALGOL cataloged procedures (ALGOFC, ALGOFCG, ALGOFCL, ALGOFCLG) to SYS1.PROCLIB or the user's procedure library.

The DTR ends with a library trailer label, 80 bytes long, with control information about the DTR.

INFORMATION ABOUT THE OS ALGOL F INDEPENDENT COMPONENT RELEASE

The Independent Component Release (ICR) contains components 360S-AL-531 and 360S-LM-532 on the level of OS release 21.0. They are distributed on magnetic tape (DTR).

The ICR can be installed under any IBM OS release. Any earlier version of OS ALGOL F that the user may have installed will be replaced.

By installation of this ICR, the system data sets SYS1.LINKLIB and SYS1.PROCLIB are referenced and modified, i.e., they must accommodate new ALGOL modules. Therefore, these data sets must have the required free space available. (Refer to the _Storage Estimates Manual_, Form GC28-6551.) SYS1.MACLIB is referenced only.

It is advisable that each installation list the DTR to determine whether any JCL cards require modification. If so, the contents of the DTR should be punched out. Then the modification can be made, and DTRALGOL can be executed as a batch job.

INSTALLATION_OF_OS_ALGOL_F_INDEPENDENT_COMPONENT_RELEASE

Before starting the reader to process the DTR that contains the independent component release, the user must:

(a) define and catalog SYS1.ALGLIB -- if it does not already exist; (refer to the section Define_and_Catalog_SYS1.ALGLIB);
(b) add a procedure CRSRC to SYS1.PROCLIB (refer to the section Add_a_Procedure CRSRC_to_SYS1.PROCLIB); and
(c) verify that SYS1.PROCLIB does not contain a member named OPTIONS.

If the user wishes to change the default compiler options, he must

(d) add a member, OPTIONS, to SYS1.PROCLIB (refer to the section Add_a_Member OPTIONS_to_SYS1.PROCLIB).

After steps (a) to (c), or (d), the user issues a Start Reader command to process the DTR (refer to the section Sample_of_Starting_Reader_to_Process_DTR).

DEFINE AND CATALOG SYS1.ALGLIB

The following is a sample set of JCL statements for the definition and cataloging of SYS1.ALGLIB:

```
//CATAL    JOB ...
//STEP    EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=A
//CATLOG    DD DISP=OLD,UNIT=3330,VOL=SER=serial1
//ALGLIB    DD DSNAME=SYS1.ALGLIB,DISP=(,KEEP),
//              VOLUME=(,RETAIN,SER=serial2),
//              UNIT=2311,
//              LABEL=EXPDT=99350,
//              SPACE=(TRK,(14,5,14)),
//              DCB=(RECFM=U,BLKSIZE=3625)
//SYSIN     DD *
  CATLG DSNAME=SYS1.ALGLIB,CVOL=3330=serial1,VOL=2311=serial2
/*
```

For more details, refer to the System Generation manual (GC28-6554), section Initializing_New_System_Data_Sets.

ADD A PROCEDURE CRSRC TO SYS1.PROCLIB

The job DTRALGOL uses CRSRC to place the ALGOL cataloged procedures into the user's procedure library. This is normally SYS1.PROCLIB but the user may choose to specify and, perhaps, modify another data set so that he can examine the cataloged procedures before including them in the system.

```
//CRSRCPRC JOB ...
//STEP    EXEC PGM=IEBUPDTE,PARM=MOD
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD DSNAME=SYS1.PROCLIB,DISP=SHR
//SYSUT2    DD DSNAME=SYS1.PROCLIB,DISP=SHR
//SYSIN     DD DATA
./ ADD LIST=ALL,NAME=CRSRC,LEVEL=01,SOURCE=0
./ NUMBER NEW1=10,INCR=10
//CRSRC   EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT  DD SYSOUT=A
//SYSUT2    DD DDNAME=PROCLIB
//PROCLIB   DD [user's procedure library]
//SYSIN     DD DUMMY
./ ENDUP
/*
```

ADD A MEMBER OPTIONS TO SYS1.PROCLIB

The default options are identical to those specified for the ALGOL macro in the System Generation manual (GC28-6554). The user may change the default options by adding a member named OPTIONS to SYS1.PROCLIB.

```
//OPTIONS  JOB ...
//STEP     EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT  DD SYSOUT=A
//SYSUT2    DD DSNAME=SYS1.PROCLIB,DISP=SHR
//SYSIN     DD DATA
./ ADD LIST=ALL.NAME=OPTIONS
./ NUMBER NEW1=10,INCR=10
 PRINT ON,NODATA
 DTRALGOL [user-specified options]
 END
./ ENDUP
/*
```

SAMPLE OF STARTING READER TO PROCESS DTR

If, for instance, the DTR is mounted on unit 182, the Start Reader command is:

```
S  RDR,182,DCB=(LRECL=80,BLKSIZE=3440,RECFM=FB),LABEL=(,NL),REGION=200K
```

The region parameter may be different for each installation.

DESCRIPTION OF OPTIONAL MATERIAL OF OS ALGOL F INDEPENDENT COMPONENT RELEASE

The optional material of the Independent Component Release (ICR) is distributed on a non-labeled, 9-track, 800 bpi, reel of magnetic tape (BLKSIZE=800,LRECL=80).

The Distribution Tape Reel (DTR) contains the source modules of the OS ALGOL F compiler (component 360S-AL-531) and the OS ALGOL F Library (component 360S-LM-532) as an unloaded version from the partitioned data set named AE01.KARAF20S.

The DTR ends with an 80-byte library trailer label that contains control information about the DTR.

The contents of the DTR can be loaded by means of the following set of JCL statements:

```
                                                                column
                                                                   72


//JOB1      JOB ...
//STEP1    EXEC PGM=IEHMOVE
//SYSPRINT  DD SYSOUT=A
//SYSUT1    DD UNIT=2314,VOL=SER=xxxxxx,DISP=OLD
//FROM      DD UNIT=2400,DISP=OLD,LABEL=(,NL),
//             VOL=(PRIVATE,RETAIN,SER=ALGOLF),
//             DCB=(BLKSIZE=800,LRECL=80,RECFM=FB)
//TO        DD UNIT=2314,VOL=SER=xxxxxx,DISP=OLD
//SYSIN     DD *
 COPY PDS=AE01.KARAF20S,RENAME=yyyyyy,                             C
               FROM=2400=(ALGOLF,1),                               C
               FROMDD=FROM,                                        C
               TO=2314=xxxxxx
```

The string xxxxxx stands for the serial number of the volume on which the optional material is to reside. The string yyyyyy must be replaced by the name to be assigned to this data set.

Index

Index to systems reference library manuals are consolidated in the publication OS Master Index to Reference Manuals, Order No. GC28-6644. For additional information about any of the subjects listed below, refer to other publications listed for the same subject in the Master Index.

Reader's Comment Form

OS ALGOL Programmer's Guide GC33-4000-3

Your comments about this publication will help us produce better publications for your use. If you wish to comment, please use the space provided below, giving specific page and paragraph references.

Please do not use this form to ask technical questions about the system or equipment or to make requests for copies of publications. Instead, make such inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Reply requested

Yes □

No □

Name ______________________________

Job Title ______________________________

Address ______________________________

______________________ Zip __________

No postage necessary if mailed in the U.S.A.

GC33-4000-3

YOUR COMMENTS, PLEASE . . .

This SRL manual is part of a library that serves as a reference source for system analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

CUT ALONG THIS LINE

Fold

Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Department 813 L

Fold

Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM / **Technical Newsletter**

IBM World Trade Corporation

This Newsletter No.	GN12 - 5900
Date	September 1, 1972
Base Publication No.	GC33 - 4000 - 3
File No.	S360 - 26
Previous Newsletters	None

OS ALGOL Programmer's Guide

This Technical Newsletter, a part of OS Release 21, ALGOL Compiler (360S-AL-531) and ALGOL Library (360S-LM-532), provides replacement and supplemental pages for the subject publication. These pages remain in effect until specifically altered.

Pages to be inserted and/or removed are listed below.

Replace:	Cover, 2 5, 6
Remove:	95
Add:	95, 96.1 96.2, 96.3

A change to the existing text is indicated by a vertical line to the left of the change.

Summary of Amendments

The replacement pages contain minor changes. Supplemental pages provide information about the OS ALGOL F Independent Component Release (ICR).

Note: *Please file this cover letter at the back of the publication to provide a record of changes.*

IBM World Trade Corporation, 821 United Nations Plaza, New York, New York 10017

Printed in U.S.A.

GC33-4000-3

OS ALGOL Programmer's Guide Printed in U. S. A. GC33-4000-3

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]