TSOS SOFTWARE DESCRIPTION INFORMATION MANUAL

RCA
Information Systems

SPECTRA 70

TIME-SHARING OPERATING SYSTEM (TSOS)

# Software Description Information Manual

RCA
Information
Systems

SPECTRA 70

TIME-SHARING OPERATING SYSTEM (TSOS)

# Software Description Information Manual

~~79-00-516~~
July 1968

First Printing: July 1968

CONTENTS

## INTRODUCTION

This manual presents a brief description of the various programming routines available under the RCA 70/46 Time-Sharing Operating System (TSOS). The system creates an appropriate system environment to satisfy the following user requirements:

- Convenient, direct access to the computer for many users simultaneously, with facilities for dynamic interaction with an executing program.
- Enhanced facilities for efficient background processing in a multiprogramming random access environment.
- Rapid response from the computer to reduce the overall time between problem definition and solution.

The 70/46 operating system is compatible with the RCA Tape Operating System (TOS). This compatibility permits the user to progress easily and naturally from TOS to the more sophisticated TSOS that is oriented toward random access.

The multiprogramming facilities of TSOS are more elaborate than those of TOS, and depend upon the following techniques:

- Virtual storage management.
- Use of a drum to retain frequently used portions of the Supervisor, control programs, and system tables.
- Spooling (buffering) of conventional card input files and files destined for the printer on random access devices (or tape).
- Construction of a simple task queue from multiple input sources, and concurrent execution of many of these tasks as the resources of the system allow. Nonconversational tasks may be entered into the task queue from remote terminals.

The considerably enhanced file management facilities in TSOS provide for:

- Cataloging of all files in the system.

INTRODUCTION
(Cont'd)

- Specification by the user of file names and characteristics at execution time, or, if he chooses, at assembly or compilation time.
- Sharing of files when specified by the user, but protection of the user's private files by the system through passwords.
- Indexed-sequential access for files retained on random access devices.
- Transcribing inactive or infrequently used files to or from the Mass Storage Unit.

The TSOS also offers convenient program preparation and testing based on facilities for:

- On-line preparation and editing of files including symbolic programs and data by means of a powerful file editor.
- Interactive, line-at-a-time compiling and interpretive execution of programs in a FORTRAN IV related program.
- System direction through a flexible and easy-to-use command language that provides for prestored common procedures.
- Program checkout with symbolic debugging commands that permit the user to request data from his program, to modify variables, and to specify locations in the program under which deferred commands are to be performed.

The routines described in this manual have been divided into seven general types:

1. Control Routines
2. Background Compilers
3. Interactive Compilers
4. Program Preparation and Testing
5. Utilities
6. Hardware Maintenance Routines
7. Scientific Routines

The user should refer to the appropriate manual for detailed information on any given routine and before attempting the execution of any program. This manual gives a broad view of the types, variety, and the large number of programs available to the TSOS user.

## CONTROL ROUTINES

### TSOS EXECUTIVE

The Executive system of the Spectra 70/46 Time Sharing Operating System (TSOS) is a modular system that adapts to its environment of programs and hardware devices. The system also supports multiprogramming and general-purpose time sharing. Inquiry programs, interactive programs and batch programs can be run concurrently in the system as user programs.

A control program governs the privileged system operation and includes the functions necessary to support the operating system. Programs other than the control program are treated as user programs.

The Executive controls the execution of tasks and controls the equipment environment in which they operate. The Executive receives interrupts, sorts them as to type and function, and gives control to appropriate routines to respond to each. By means of time slicing, it provides rapid and complete service to a number of users concurrently.

Executive routines control I/O activity and allocate machine resources. The Executive controls system startup, performs error recovery functions, and accumulates user accounting statistics.

The most frequently used components of the Executive are resident in main memory with the others readily available from the drum. It executes largely in the privileged mode and its locations are not addressable by other programs. The Executive is not itself generally time-sliced.

The Executive control functions are invoked through the use of both macros and commands.

### TSOS Executive Macros

#### REQM - Request Memory

This macro requests a contiguous area of memory for a user's program at object time.

#### RELM - Release Memory

This macro releases a contiguous area of memory in a user's program at object time.

#### CSTAT - Change Status

This macro changes the status of a specified page or all pages in a program.

TSOS Executive Macros (Cont'd)

SAVE - Save Register Contents

This macro saves contents of general registers for a calling program.

RETRN - Return to a Program

This macro returns control to the calling program.

RDATA - Read Record from SYSDTA

This macro is used to retrieve the next record from SYSDTA file.

WROUT - Write Records to SYSOUT

This macro sends a message from the program to the SYSOUT file.

WRTRD - Terminal Tandem Write Read

Conversational mode programs use this macro to send a message to the terminal that requires a response.

SETSW - Set Switch

This macro sets and resets the task switches.

GETSW - Get Switch

The Get Switch macro retrieves the task switches and places them in Register 0.

WRLST - Write Print Line

The user program uses this macro to write a record to SYSLST.

TMODE - Task Mode

This macro provides the user with task information in a designated area.

LSTFM - SYSLST Format Macro

This macro indicates editing options for records written to SYSLST file.

PASS - Relinquish Remainder of Time Slice

This macro causes the task to relinquish its current time slice.

TSOS Executive Macros (Cont'd)

SPEXT - Supply Executive Table

This macro enables a Class II program to obtain a status block required by the user's interrupt routines.

GDATE - Get Current Date

This macro acquires the current date.

TOS Macros Supported by TSOS

The TOS macros supported by TSOS provide the same functions as in the Tape Operation System. The TSOS expansions of the macro are the same as for TOS, thus providing object level compatibility. Unless otherwise noted in the following sections, these macros are permitted in either Class I or Class II programs.

| TOS Macro | Description | Class | SVC# |
|---|---|---|---|
| LPOV | Load Program Overlay | both | SVC 2 (0) |
| FLOAD | FCP Load | I | SVC 25 (0) |
| ADEXT | Address Executive Tables | I | SVC 3 (0) |
| STXIT | Set Contingency Routine Address | both | SVC 4 (0) |
| EXIT | Exit from Contingency Routine | both | SVC 18,19,20 (0) |
| ASCII | Set ASCII Mode | both | SVC 16 (0) |
| EBCD | Set EBCDIC Mode | both | SVC 17 (0) |
| TERM | Terminate Program | both | SVC 28 followed by SVC 9 (0) |
| TERMD | Terminate and Dump | both | SVC 28 followed by 22 (0) |
| GETOD | Get Time of Day | both | SVC 23 (0) |
| TERMJ | Terminate Job | both | SVC 63 (0) |
| ERFLG | Set Error Flag | both | SVC 60 (0) |
| STUT1 | Set SYSUT1 Flag | both | SVC 57 (0) |
| STUT2 | Set SYSUT2 Flag | both | SVC 56 (0) |

| TOS Macro | Description | Class | SVC# |
|---|---|---|---|
| MONTB | Address Monitor Table | both | SVC 59 (0) |
| RDCRD | Read System Input | both | SVC 62 (0) |
| PROUT | Write Print Line Image | both | SVC 58 (0) |
| WRTOT | Write System Output | both | SVC 61 (0) |
| TOCOM | Move to Common Data Area | both | SVC 32 (0) |
| EXCOM | Set from Common Data Area | both | SVC 33 (0) |
| GEPRT | Get Program Time | both | SVC 24 (0) |
| SETIC | Set Time Clock | both | SVC 21 (0) |

## Command Language

The command language permits individuals using the system to identify themselves, to specify work for the system, and to monitor that work. It is the principal means of communication between TSOS and individuals who use the system. These may be:

System Administrator

System Operators

Users

When granted access to the system, an individual is assigned one or more command privilege classes that determine the commands he is authorized to issue to the system.

System administrators use the facilities of the command language to authorize user access to the system, and to maintain and retrieve records of system use for accounting and administrative purposes.

**System Administrator Commands**

JOIN - Grants a user access to the system.

SEVER - Prohibits a user subsequent access to the system.

LOGON - Identifies the user to the system.

LOGOFF - Terminates a task.

CANCEL - Cancels a waiting task or terminates a task being executed.

STATUS - Presents the status of tasks being executed.

PRIORITY - Allows priority of a task to be changed.

**System Operator Commands**

BROADCAST - Allows the operator to send a message to all active terminals.

MESSAGE - Allows the operator to send a message to a specific user's terminal.

RCARD - Reads punched cards and transcribes them to a work file.

SHUTDOWN - Terminates task in preparation for physical shutdown of the computer.

BIAS - Schedules the mix of conversational and nonconversational programs.

ENTER - Places a nonconversational task into the job stream.

CANCEL - See System Administrator commands.

STATUS - See System Administrator commands.

PRIORITY - See System Administrator commands.

SUSPEND - To be specified later.

| User Commands | |
|---|---|
| | LOGON - See System Administrator commands. |
| | LOGOFF - See System Administrator commands. |
| | FILENAME - Enables the user to execute a PROC file. |
| | PROCEDURE - Serves as starting delimiter of a PROC file. |
| | ENDP - Returns control from a PROC file to the Primary SYSCMD. |
| | ENTER - Specifies the name of a catalogued file for input of nonconversational tasks. |
| | EXECUTE - Loads object module and initiates program execution. |
| | PARAMETER - Indicates language processor options. |
| | SKIP - Tests the designated task switches. |
| | REMARK - Indicates the user's remarks to the SYSOUT file. |
| | STEP - Resets the Monitor Table and task switches 16-31. |
| | LOAD - Loads a program into storage. |
| | SETSW - Sets the task switches on, off, or to an inverted position. |
| | TYPE - Prints a message on the operator's console typewriter. |
| | SECURE - Reserves resources for task execution. |
| | EOF - Transfers to user's end of file routine. |
| | STATUS - Obtains current status of tasks for a user. |
| | BREAK - Allows transfer of control from a user program to process command statements. |
| | SYSFILE - Allows reassignment of SYSIPT, SYSOPT, or SYSDTA. |
| | CANCEL - See System Administrator commands. |
| | SUSPEND - To be specified later. |

**User Commands (Cont'd)**

PAUSE - Allows operator directions to be printed on the system's console typewriter.

INTR - Causes control to be resumed in the operator's communication routine.

IHBPXP - Loads and initiates the TSOS Basic Processor Unit Exerciser routine.

**Utility Commands**

PRINT - Initiates printing of a specified file on the printer.

PUNCH - Initiates punching of a specified file onto cards.

ACTIVATE - Transcribes a file from mass storage to disc.

DEACTIVATE - Transcribes a file from disc to mass storage.

## DATA MANAGEMENT SYSTEM

TSOS provides comprehensive facilities for systematic and convenient management of the conventional input/output files used by data processing programs, source programs, object programs, and subroutines; and textual information to be organized and processed by the File Editor.

These facilities fall naturally into two categories:

- File cataloging and management
- Problem program input/output

File management facilities provide the means for identifying files; for storing and retrieving them within the system; for sharing them with other users; for copying, modifying, and erasing them; and for defining their existence and use in the system. Problem program input/output facilities provide for the actual transfer of data to and from programs that are in execution.

**File Security**

When a user creates a file, he is recognized as its owner. No other user can obtain access to the file, unless the owner catalogues it with the SHARE=YES parameter specification.

**File Retrieval**

File access to both the owner and the sharer is also controlled by two options. Two passwords can be associated with the file: a read password and a write password. Two passwords facilitate two levels of file sharing. The ACCESS parameter can be used to limit the file to read access.

A file, marked as sharable, can be accessed by any user that can provide the identification code of the owner, the file name, and the appropriate password (if required).

The passwords required to gain access to protected files must be supplied by a PASSWORD command.

Temporary files can be created for the duration of a task (Logon to Logoff); optionally, they may be recatalogued as permanent. Temporary files are system allocated from public volumes.

**Volume Concepts**

Volumes are classified as follows:

| CLASS | CLASSIFICATION | DA | TAPE | MUST BE PERMANENTLY MOUNTED | RESTRICTED TO SINGLE TASK | FILE PROT. | TSOS I/O MACRO ACCESS | TOS EXCP ACCESS |
|---|---|---|---|---|---|---|---|---|
| CLASS II | PRIVATE | YES | YES | NO | YES | YES | YES | NO |
| | PUBLIC | YES | NO | YES | NO | YES | YES | NO |
| | TOS | YES | YES | NO | YES | NO | YES | YES |
| CLASS I | UNIT RECORD | | | | | | NO | YES |

## File Space on Public Volumes

When each user is JOINed to the system, the system administrator specifies the maximum amount of space on public volumes that the user could require. The space is not actually allocated to the user until it is required, and is referred to as the user's public space allotment.

Primary and secondary space requirements can be specified for files occupying more than one volume. Secondary space will be dynamically allocated by the system as required for _public_ volumes.

## _The Access Methods_

1. Sequential Access Method (SAM)
2. Indexed-Sequential Access Method (ISAM)
3. Basic Direct Access Method (BDAM)
4. Basic Tape Access Method (BTAM)

### Sequential Access Method (SAM)

Logical records are retrieved by the use of the GET or GETR macro-instructions, which supply a logical record to the program. The access method anticipates the need for records based on their sequential order (the order in which they are written) and normally will have the desired record in storage, ready for use. Logical records are designated for output by use of the PUT macro. The program can continue as if the data record was written immediately, although the access method's routines may perform blocking with other logical records, and delay the actual writing until the output buffer has been filled. Buffers are automatically scheduled by the system. The sequential access method is, for the most part, device independent and allows files on both magnetic tape and random access devices to be processed.

### Indexed-Sequential Access Method (ISAM)

The Indexed-Sequential Access Method processes logical records in an indexed-sequential file. It may be used to:

- Create an indexed-sequential file in a sequential or nonsequential manner.
- Retrieve the logical records of the file in a sequential or nonsequential manner.

Indexed-Sequential Access Method (ISAM) (Cont'd)

- Update records in a sequential or nonsequential manner.
- Insert new records in their proper logical sequence within the file.
- Delete selected records from the file.

A file is created by use of the PUT macro-instruction which records records in logical sequence of the keys within the records. Logical records are sequentially retrieved from a created file by use of the GET or GETR macro-instruction.

Retrieved logical records are updated and returned to the file by using the PUTX macro-instruction. The INSRT and STORE macro-instructions are used to add new records to the file. A record may be eliminated from the file by using the ELIM macro-instruction. The program can continue as if the data record were written immediately, although the access method's routines may perform blocking with other logical records, and delay the actual writing until an output buffer has completed. Buffers are automatically scheduled by the system when sequential processing is being performed.

Basic Direct Access Method (BDAM)

The Basic Direct Access Method (BDAM) provides the user with the ability to create and process files on random access devices. The user is free to establish his own file organization. Macros are provided to load, read, update, add, or replace records.

Files can be of two types: with a key or without. Nevertheless, when a file is specified as having keys, every record in the file must have a key and all keys must be of the same length. Two record formats are provided: fixed-length and undefined. If the user wishes blocked records, he must provide his own blocking and deblocking routines.

Basic Tape Access Method (BTAM)

The Basic Tape Access Method (BTAM) provides the programmer with an efficient and flexible means for storing and retrieving the blocks of a sequentially organized tape file.

A major feature of BTAM is that it permits the programmer to transfer data from a magnetic tape device directly to a specific area of main storage or, conversely, to transfer data from a specific area of

Basic Tape Access Method (BTAM) (Cont'd)

main storage to the device, without first moving it to or from a buffer. This feature is particularly useful, if, owing to the large size of the records to be processed, no main storage space is available for buffers.

The user may supply his own I/O areas. When supplying his own I/O areas, the user must ensure that the buffers do not cross page boundaries. I/O requests referencing buffers that cross page boundaries are not processed by the DMS.

Because of the additional modes of the OPEN macro-instruction provided for the physical access method, it is effective in applications where records are to be alternately read and written, in rapid succession, from and to a file used as a temporary extension of main storage.

<u>Data Management Commands</u>

| | |
|---|---|
| <u>ALLOCATE</u> | - Performs storage allocation for direct access devices. |
| <u>CATALOG</u> | - Creates or alters a catalog entry for a file. |
| <u>COPY</u> | - Copies a file. |
| <u>DELETE</u> | - Deletes one or more entries from the catalog. Does not disturb the file. |
| <u>DROP</u> | - Removes the Hold status of a Link name. |
| <u>ERASE</u> | - Deallocates space assigned to one or more files and deletes catalog entries. |
| <u>FILE</u> | - Allows the user to catalog a temporary file, allocate space for a file, preassign devices, complete or modify the FCB of a file at execution time, and supply the symbolic device names and device types for TOS programs. |

Data Management Commands (Cont'd)

FSTATUS - Furnishes the status of one or more files in hard copy form.

HOLD - Used to override normal releasing of devices and file commands.

PASSWORD - Supplies passwords to a task.

RELEASE - Releases the definition of a previous FILE command.

RENAME - Changes the name of a file.

TRANSFER - Transfers the catalog entry of a file on a public volume from one user to another.

Data Management Macrcs

The following TSOS file management macros perform essentially the same functions as the correspondingly named commands:

| MACROS | COMMANDS |
|---|---|
| ALLOC | ALLOCATE |
| CATAL | CATALOG |
| DEL | DELETE |
| ERASE | ERASE |
| FILE | FILE |
| FSTAT | FSTATUS |
| REL | RELEASE |
| RENAM | RENAME |

The following is a list of the various TOS macros that are fully supported or partially supported under TSOS:

## Executive Macros

| TOS | TSOS SUPPORT | | |
|---|---|---|---|
| MACROS | NO SUPPORT | FULL SUPPORT | PARTIAL SUPPORT |
| TYPE | | X | |
| LPOV | | X | |
| ADEXT | | | X |
| STXIT | | X | |
| DDEV | | X | |
| TERM | | X | |
| TERMS | X | | |
| EXCPW | | X | |
| EXCP | | X | |
| WAIT | | X | |
| CHECK | | X | |
| AID | X | | |
| ASCII | | X | |
| EBCD | | X | |
| EXIT | | X | |
| CPCI | | X | |
| TERMD | | X | |
| DMODE | | X | |
| DTYPE | | X | |
| GETOD | | X | |
| COMTY | | X | |
| CCB | | X | |
| CKPT | X | | |
| FLOAD | | X | |
| SMODE | | X | |
| SPRG | | X | |
| QUIET | | X | |
| ASSGN | | X | |

## Monitor Macros

| MACRO | NO SUPPORT | FULL SUPPORT | ACTION TAKEN BY TSOS CONTROL PROGRAM |
|---|---|---|---|
| TERMJ* | | X | Similar to TOS action. |
| RDCRD | | X | Supports TOS function of reading one line from SYSIPT. |
| WRTOT | | X | Cataloged file is created. User can explicitly request punching of Object Module. |
| ERFLG | | X | Set Error Flag. |
| MONTB | | X | Address Monitor Table. |
| PROUT | | X | Print line images are spooled for listing on printer. |
| STUT1 | | X | Similar to TOS action. Set SYSUT1 Flag. |
| STUT2 | | X | Similar to TOS action. Set SYSUT2 Flag. |
| MNSNP | X | | |

*All except TERMJ are supported for Class I programs only.

## DYNAMIC LINKING LOADER

The TSOS Linking Loader provides an alternate method of loading Class II programs. Class II programs containing tree structured overlays must be bound by the Linkage Editor. All other Class II programs may be loaded by the Linking Loader. It will be part of the TSOS control system.

The TSOS Linking Loader accepts three types of input:

1. Object Module Files consisting of one or more object modules (For example, Language Processor output).

2. Object Module Libraries.

3. INCLUDE Control statements to allow selective loading of object modules.

### Linking Loader Functions

The Linking Loader can perform the following functions:

- Load one or more object modules directly into memory.

- Obtain and load object modules from secondary inputs.

- Search one or more object module libraries to obtain modules to be loaded into memory (explicit calls).

- Collect control sections with COMMON attribute and reserve space for them. Both named and "blank" COMMON are supported.

- Satisfy unresolved external references (implicit calls) by an automatic search of a user's TASKLIB and the System Library.

- Adjust CSECT origins according to attributes specified in the ESD card defining the CSECT.

- Allow programs in load module format (that is, Linkage Editor output) to include object modules.

- Allow conversational TSOS languages to include subprograms compiled by other languages to be included at execution time.

<u>Linking Loader Functions</u> (Cont'd)

- List all outstanding, unresolved, external references at the completion of the load process.
- Construction of an IDA file.

## BACKGROUND COMPILERS

### COBOL VERSION 1

The TSOS COBOL Background Compiler (CLASS I program) is a modification of the existing TOS COBOL Compiler. It provides the TSOS COBOL user the basic TSOS facilities. The object code produced by the COBOL Version 1 Compiler is CLASS I object code.

The user can specify the following options:

1. Alternate input files.

2. That object modules be written to disc files.

3. That a diagnostic file be generated and catalogued for subsequent query through a post compilation routine.

4. That an internal symbol dictionary be generated and catalogued for subsequent use by the Interactive Debugging Aid (IDA).

#### Alternate Input Files

The TSOS COBOL Compiler normally retrieves the next data record from the SYSIPT file, a card file which has been spooled to a temporary random access file. The user may optionally redirect SYSIPT to a catalogued data file.

#### Disc Files

The Compiler normally writes an output record (object code) to SYSOPT (card output). When it is desired to catalogue an object module, the user redirects the Compiler by specifying the parameter CARD=YES. This causes an indicator to be set in the monitor table to effect the cataloguing of the object module to disc. Note that //PARA CARD=YES is used to maintain TOS compatibility.

#### Diagnostic File

The user may request that a diagnostic file be produced and stored on disc by the parameter //PARA DIAGF=YES.

The TSOS COBOL Diagnostic routine is a post compilation routine providing diagnostic information concerning a compilation at a remote terminal. The routine will be used as a TSOS CLASS II Conversational program.

The diagnostic routine is activated by any of the following:

Diagnostic File (Cont'd)

STATUS - Provides a brief summary of the results of the compilation. Included in the summary are the total number of errors detected, and the flag errors.

PRINT - Lists all the error messages with their explanations or the error message identification and the associated statement number.

HELP - Lists the entire set of command names if the operand field is omitted, or the functional description of those command names specified in the operand field.

STOP - Permits the user to interrupt and terminate execution of any diagnostic command.

Internal Symbol Dictionary

The user, by specifying //PARAM SYMD=YES causes the Compiler to produce and catalogue an internal symbolic dictionary.

This dictionary is used when the user is debugging his program through IDA.

COBOL VERSION 2

The TSOS COBOL Background Compiler (CLASS I program) produces object code that allows the user to run his program as a CLASS II program. All source input statements (exclusive of the COBOL library functions) will be read from the SYSDTA file. The SYSDTA file can be catalogued as a data file, a system card reader, or a terminal.

Object modules are saved if the user specifies CARD=YES or DISC=YES. DISC=YES is used for LOAD and GO situations.

Source and object listing will be provided if the user specifies LIST=YES and/or OBJECT=YES. The Data Management Easy Access Method (EAM) is used to write the listing.

The compile-time facilities provided in the TSOS COBOL Background Compiler Version 1 are also available in Version 2.

The COBOL language elements implemented by the Version 2 Background Compiler are the same as those provided in the TOS/TDOS COBOL Compiler. In addition, a new optional clause is added in the SPECIAL-NAMES

COBOL VERSION 2 (Cont'd)

paragraph to permit the use of a terminal device in the ACCEPT and DISPLAY statements. Also, the implementation of the SELECT sentence will be modified to conform with the Data Management System (DMS) requirements.

COBOL object programs will utilize the following Data Management System Access Methods:

SAM (Sequential Access Method)

ISAM (Indexed-Sequential Access Method)

BDAM (Basic Direct Access Method)

The selection of an access method is based on the ORGANIZATION and ACCESS clauses in the ENVIRONMENT DIVISION.

The SORT/MERGE System is not capable of running as a TSOS Class II program. COBOL source programs containing the SORT feature must be compiled by the TSOS COBOL Version I Compiler.

FORTRAN BACKGROUND COMPILER VERSION 1

The TSOS FORTRAN Background Compiler Version 1 is a modification of the TOS/TDOS FORTRAN Compiler. The Compiler operates in the nonconversational mode as a Class I type program under the Spectra 70/46 Time Sharing Operating System (TSOS). The TSOS FORTRAN Background Compiler generates object modules that are executed as Class I programs under TSOS. The TSOS FORTRAN user can specify certain basic facilities:

1. Source input can be retrieved from a catalogued data file on disc.

2. A generated Object Module can be written to disc, to be automatically punched on cards at task termination.

3. Requested listings will be written to disc, to be automatically printed at task termination.

4. An Internal Symbol Dictionary (ISD) can be generated, for subsequent use under the TSOS IDA System when debugging at program execution time.

FORTRAN BACKGROUND COMPILER VERSION 1 (Cont'd)

5. A Diagnostic File can be catalogued and generated to the disc for subsequent query through the post-compilation TSOS FORTRAN Interactive Diagnostic routine.

Prior to compilation with the TSOS FORTRAN Background Compiler, the TSOS FORTRAN user may compose, syntax check, and correct source programs using the TSOS FORTRAN Interactive System. The user can then direct the Background Compiler to process the resulting catalogued indexed-sequential file containing the corrected source program.

The Compiler maintains a total count of error messages, and also counts according to severity code types. When all detail records have been processed, a summary record containing the three total values is generated to the disc. This Summary Record will be written in Update mode and will overlay the dummy Summary Record written at file initialization. The Diagnostic File will be subsequently closed.

## Internal Symbol Dictionary (ISD) Generation

The Version 1 Compiler generates Internal Symbol Dictionary (ISD) if the user specifies the Compile-time parameter SYMDIC = YES.

The ISD will contain the given names of symbols defined in the FORTRAN source program, together with the necessary information required to define each symbol's object program location, length and type attributes. The symbols are simple variables, dimensioned variables, and statement numbers.

The ISD is used when on-line debugging is desired through the use of the Interactive Debugging Aid (IDA).

## FORTRAN BACKGROUND COMPILER VERSION 2

The TSOS FORTRAN Background Compiler (Class I program) produces object code that allows the user to run his program as a Class II program. All source input statements will be used from the SYSDTA file. The SYSDTA file can be catalogued as a data file, as a system card reader, or as a terminal.

Object modules are saved if the user specifies CARD=YES or DISC=YES.

Source and object listings will be provided if the user specifies LIST=YES and/or OBJECT=YES. The Data Management Easy Access Method (EAM) is used to write the listing.

The compile-time facilities provided in the TSOS FORTRAN Background Compiler Version 1 also are available in Version 2.

## REPORT PROGRAM GENERATOR (RPG)

The RCA Report Program Generator (RPG) for the Spectra 70 System is a simplified programming language that produces a printed report without requiring a detailed knowledge of machine coding. The source program specifications are written on a series of preprinted tabular forms that describe the input data, output forms, and calculations to be performed.

The principal function of the RPG is program generation. A machine language program is generated in accordance with specifications furnished by the programmer. RPG automatically allocates the necessary storage locations, provides linkage to input/output operations, and includes constants and other designated information. During the data processing state, the object program coding processes the user's input data files and produces the output files and/or printed reports.

RPG is a compiler which is itself a Class I program and produces object programs not capable of paging. RPG offers such report features as input data selection, editing, calculation, summarizing, control breaks, and file updating.

## TSOS ASSEMBLER

The TSOS Assembler is organized to work efficiently and conveniently with the TSOS Remote Terminal user by (1) using the DMS I/O system, (2) providing an optional assembly diagnostic file that can be interrogated remotely (by an Assembly Diagnostic routine described in this document) and (3) providing an optional Internal Symbol Dictionary for use with the Interactive Debugging Aid. This Assembler supports a larger Assembly language that contains many of the language elements currently in the larger IBM Assembly Systems (such as IBM DOS, OS-360, etc.). Some of these language features are: (1) a greater flexibility in naming macro set symbols, (2) availability of type and length attributes at source code generation time, (3) additional conditional assembly statements, and (4) an increased number of macro operands.

The TSOS Assembler is designated to run under control of the TSOS Executive System on a Spectra 70/46. The Assembler is a Class II sharable program.

The Assembler accepts programs and macro definitions written in the TSOS Assembly Language. The output from the Assembler includes (1) a listing file that contains the source program, object code, and diagnostic information; (2) an object file that can be loaded and executed; and (3) a diagnostic file that can be tested from a remote terminal.

### Input

Input to the TSOS Assembler is, of course, the User's Source Program.

The TSOS Assembler assumes that the user's source program is on the system SYSDTA file and that it is accessed with the RDATA executive macro. The RDATA macro can support the following devices:

1. Card reader (the card reader is supported by means of a temporary file that is spooled from the card reader to the disk).

**Input** (Cont'd)

2. Catalogued SAM file.

3. Catalogued ISAM file.

4. Terminal.

The Assembler will test General Register 15 upon return from the RDATA macro to determine the input device type.

**Sequential Files**

Card reader files and SAM files are treated as 80-character card images. The standard columns are column 1 for start, column 71 for end, and column 16 for continuation. Standard columns may be changed by the Assembler ICTL statement. (See the Assembly Reference Manual for complete description of standard columns and the ICTL statements).

**Index-Sequential Input Files**

Index-sequential files will be treated as variable-length card images. Images that are less than 80 characters will be space filled to the right, and card images that are longer than 80 characters will be truncated. Since ISAM input files can be in File Editor format, an eight-byte key will be assumed to be at the beginning of each record. The Assembler will adjust the start column to 9, the end column to 79 and the continue column to 24. If a user uses an ISAM input file that is not in File Editor format, then an ICTL Assembler instruction should be used to reset the standard columns.

**Terminal Input**

Input from a remote terminal will be treated as variable-length card images. Records that are not equal to 80 bytes will be space-filled or truncated as in Index-Sequential input. Columns 1, 71, and 16 will be used as the start, end, and continue columns. Column 1 means the first position at which the terminal I/O routine released the keyboard to the user. Horizontal tab characters are treated as one blank character.

## Macro Libraries

A collection of macro definitions can be made available to more than one source program by placing the macros in a macro library. When this is done, the macro definition can be referenced by writing only the macro instruction call line.

Two kinds of macro libraries exist in the system: one is a system macro library, which is available to all users, and the second is a private macro library, which is catalogued as a user's file. Both the system macro library and a private macro file can be referenced in the same assembly. If a macro instruction references a macro definition that is in both libraries, the definition in the private library will be used. Definitions in a private macro library can call on other macros in the same private file, or they can call on system macros. System macros can call on other system macros; however, they cannot call macros that are in a private library unless the macro from the private library is also referenced in the source program.

## External References to Macro Libraries

The system macro library is specified with a File Control Block that has LINK=SYSLIB. When a user wishes to supply his own file as the system macro library, he may supply a /FILE card for the new macro library that has SYSLIB as the link.

The alternate macro library has a LNIK=ALTLIB specification. When a user wishes to use a private macro library, a /FILE card and a /PARAM card specifying LATLIB=YES must precede his job. If no /FILE card is found with LINK=ALTLIB, the Assembly will proceed using only the system macro library.

## Monitor Parameter Table Indicators

Alternate Macro Library

The alternate macro library bit is set when:

```
/PARAM     ALTLIB=YES
```

is found in the job control language statement. The user also must have a /FILE card with LINK=ALTLIB, which gives the name of his macro library file.

Monitor Parameter Table Indicators (Cont'd)

## Cross-Reference Indicator

The cross-reference indicator is set with the following job control language card:

```
/PARAM XREF= YES
             NO
```

YES is the default case. When XREF=YES is in effect, the Assembler prints a combined symbol table map and cross-reference listing on the output file.

## Error File Indicator

The error file indicator is set with the following command:

```
/PARAM     ERFILE= NO
                   YES
```

NO is the default case. When ERFILE=YES is specified, the Assembler creates a file that can be used at a later time with the Assembler Diagnostic routine.

## Listing Indicator

The listing indicator is set with the following command:

```
/PARAM   ASMLST= YES
                 NO
```

YES is the default case. The Assembler will produce a program listing on the output file when ASMLST=YES is in effect.

Note: When ERFILE=YES is specified, it is strongly suggested that the listing be suppressed. The listing can be obtained by issuing a PRINT command for the Diagnostic Error File.

## Internal Symbol Dictionary (ISD) Indicator

The ISD indicator is set with the following command:

<u>Monitor Parameter Table Indicators</u> (Cont'd)

```
/PARAM   SYMTAB= NO
                 YES
```

When SYMTAB=YES is specified, the Assembler produces ISD records that allow symbolic debugging facilities with the system debugging program IDA.

<u>Outputs</u>

<u>Program Listing</u>

The program listing will contain a listing of the External Symbol Dictionary (ESD), the source program with the corresponding object code and associated flags, cross reference data, and diagnostic executive WRLST macro. The listing will not be saved after it is spooled to the printer, that is, WRLST will not create a catalogued file).

<u>ASMDOS</u>

ASMDOS is an enhancement of the TOS Assembler and runs in the 70/45 mode (a 65K processor with at least 40K assigned to the Assembler) as a Class I program. ASMDOS operates within the TOS Executive and Monitor.

ASMDOS source language is identical to that of the TOS Assembler; however, the TSOS language has some features that are not available in ASMDOS:

1. DC, DS Enhancements
2. Operation Code Enhancement
3. Operation Synonym
4. Continuation Line Enhancement
5. Copy Predefined Source Code

Summary of TOS, ASMDOS, and TSOS Assemblers

| Feature | TOS | ASMDOS | TSOS |
|---|---|---|---|
| 1. FCALL and TS operations | No | No | No |
| 2. Hexadecimal Self-defining term<br>Binary Self-defining term<br>Character Self-defining term | 1-6 digits<br>1-24 bits<br>1-3 chars. | 1-8 digits<br>1-32 bits<br>1-4 chars. | 1-8 digits<br>1-32 bits<br>1-4 chars. |
| 3. DC and DS-<br>Duplication factor and Modifiers expressed as: | Self-Defining term | Self-defining term or a Macro variable Symbol | Self-defining term or Macro Variable Symbol or Expression enclosed in parentheses |
| 4. Macro name same as operation code | No | No | Yes |
| 5. OPSYN | No | No | Yes |
| 6. Control Section Boundary | Double Word | Page | Page |
| 7. PSECT | No | Yes | Yes |
| 8. Control Section Attributes Variable Read, Public, Privileged | No | Yes | Yes |
| 9. COPY | No | No | Yes |
| 10. Continuation lines | 2 | 2 | No real limit |
| 11. Source Language Correction | Yes | Yes | No |
| 12. Number of symbols | $\leq 4096$ | $\leq 4096$ | $\leq 8192$ |
| 13. Output formats | -- | Same as TOS except attributes may be output in ESD card | Not yet specified |

Macro Language

| Feature | TOS | ASMDOS | TSOS |
|---|---|---|---|
| 1. Operand Sublists | No | Yes | Yes |
| 2. Nesting level | 3 | 6 | Limited only by Storage |
| 3. Use of macro variables outside of macro definitions | No | Yes | Yes |
| 4. Number of Macro-Instruction Operands | 50 | 50 | 50 |
| 5. Number of Macro Set Symbols | | | |
| &AGn | 25 | 25 | 100 |
| &CGn | 20 | 50 | 100 |
| &BGn | 480 | 480 | 480 |
| &ALn | 16 | 16 | 50 |
| &CLn | 0 | 0 | 50 |
| &BLn | 128 | 128 | 128 |
| 6. Generated Sequence Symbols | No | Yes | Yes |
| 7. Conditional Assembly | No | Yes | Yes |
| 8. Prototype relaxation | No | Yes | Yes |
| 9. SET operation code | No | Yes | Yes |
| 10. Type Attribute | No | No | Yes |
| 11. Length Attribute (macro) | No | No | Yes |
| 12. Count Attribute | No | Yes | Yes |
| 13. Number Attribute | No | Yes | Yes |
| 14. Size of macro operand | 8 | 127 | 127 |
| 15. Size of SETC variable | 8 | 127 | 127 |
| 16. Size of Substring | 16 | 255 | 255 |
| 17. Statements not allowed in a macro | COPY<br>END<br>ICTL<br>ISEQ<br>START | COPY<br>END<br>ICTL | COPY<br>END<br>ICTL |

## INTERACTIVE COMPILERS

### BASIC

The Beginner's All-purpose Symbolic Instruction Code provides the ability for the creation, modification, and execution of BASIC programs in an interactive mode.

The TSOS BASIC System includes comprehensive language facilities for editing source programs. There are commands for editing a program (deleting or extracting part of a program, renumbering a sequence of lines, etc.), listing a program, executing a program, saving and reloading programs on a user's disc file space, and interrogating the BASIC system. The intent of these language facilities is to combine TSOS BASIC source statements and editing commands into a homogeneous language for conversational program preparation and execution.

Therefore the user does not have to manually switch between edit and program construction modes.

BASIC validates source language statements and prompts the user when an error is detected. If a source error occurs, the user corrects only the part of the statement in error and not the entire statement line.

The BASIC file is not saved unless the user specifies the SAVE command. Edit commands are executed immediately and therefore do not become part of the BASIC file.

The BASIC Compiler is essentially a one-pass, load-and-go subsystem. The Compiler generates CLASS II nonreentrant object code. The Compiler itself is CLASS II reentered code.

### BASIC Control Statements

GOTO

This statement causes an unconditional transfer of control to the statement whose line number is referenced.

ON

When used in connection with the GOTO statement, this statement provides a conditional transfer of control.

<u>BASIC Control Statements</u> (Cont'd)

<u>STOP</u>

This statement terminates program execution.

<u>END</u>

This statement denotes end of compilation. If an END is missing, the Compiler will supply one automatically.

<u>GOSUB</u>

This statement transfers control to a subroutine.

<u>RETURN</u>

This statement returns control to the statement immediately following a GOSUB command.

<u>IF</u>

This statement denotes a conditional branch.

IF (an arithmetic expression is true) THEN (go to this statement number). If an expression is not true proceed to next statement.

<u>FOR</u>

This statement initiates a program LOOP. The LOOP is terminated by a NEXT command. Loops may be nested.

<u>NEXT</u>

This statement terminates a loop. It serves as a delimiter for the extent of a loop.

<u>BASIC Input/ Output Statements</u>

<u>READ</u>

This statement reads in data defined by a DATA statement and assigns the data to the variables listed.

<u>DATA</u>

DATA statements are an ordered set of nonexecutable statements, that in effect, constitute a data file. This set can be placed anywhere in a program.

## BASIC Input/Output Statements (Cont'd)

RESTORE

This statement resets the DATA list so that it points to the first item in the list.

INPUT

This statement is similar to the READ statement. Unlike the READ statement, however, the INPUT statement is dynamic (it acquires input from the user's terminal).

PRINT

This statement causes data values to be written to the user's terminal.

## BASIC Declaration Statements

DIM

This statement reserves sufficient room for a list or a table. (It is usually used to specify an array.)

DEF

This statement defines a function within the BASIC program. This ability will be a time saver for functions used repeatedly. If a function is defined as ABC, the user merely supplies the values and the function is performed immediately.

LET

This statement assigns a value to a variable.

## BASIC Matrix Operations

The BASIC user could perform matrix operations by using the statements described thus far. However, the following is a special set of instructions for matrix computations of a numeric array. Each matrix instruction must start with the word "MAT".

MAT READ

This operation reads the matrices named in the Read statement; their dimensions having been previously supplied.

MAT PRINT

This operation prints the matrices named in the Print statement.

BASIC Matrix Operations (Cont'd)

MAT ADD

MATC=A+B; place in the matrix c, the sum of A plus B.

MAT SUBTRACT

MATC=A-B; place in the matrix c, the result of A minus B.

MAT MULTIPLY

MATC=A*B; multiply the matrix A by the matrix B, placing the product in the matrix c.

MAT INVERT

MATC=INV(A); invert the matrix A.

MAT TRANSPOSE

MATC=TRN(A); transpose the matrix A.

MAT SCALAR-MULTIPLY

MATC=(A)*B; multiply the matrix B by the number A. The number A, which must be in parentheses, also can be a formula.

MAT ZEROS

MATM=ZER; fill out m with zeros.

MAT ONES

MATM=CON; fill out m with ones.

MAT IDENTITY

MATM=IDN; matrix m is defined as an identity matrix.

<u>BASIC Edit Commands</u>

<u>NEW</u>

This statement informs the BASIC Compiler that the user is creating a new program.

<u>OLD</u>

This statement informs the BASIC Compiler that the user requests the loading of an old program; (that is, a program permanently stored by a SAVE command).

<u>RENAME</u>

With this statement the user can change the name of the program currently in his work space.

<u>SCRATCH</u>

The user can erase the current contents of his work space by using this statement. The program name, however, is retained.

<u>LENGTH</u>

This statement causes the amount of space occupied by the user's program to be printed on the user's terminal.

<u>STATUS</u>

This statement prints the program name and the current time of day onto the user's terminal.

<u>SAVE</u>

This statement causes the user's current program to be cataloged as a permanent file in the user's public space.

<u>UNSAVE</u>

This statement erases BASIC programs that have been previously SAVED.

<u>CATALOG</u>

This statement prints a list of the user's BASIC programs previously SAVED.

BASIC Edit Commands (Cont'd)

SYSTEM

This statement is used to return control to the TSOS.

RUN

This statement directs the system to compile and execute the BASIC program in the user's work space.

LIST

This statement directs the system to print, on the user's terminal, the sequence of lines referenced. When no line number is given, the entire program is printed.

DELETE

This statement directs the system to delete from the user's work space, the sequence of lines referenced. When no line number is given, the entire program is deleted (same effect as the SCRATCH command).

EXTRACT

This statement is used to extract, from the user's work space, the sequence of lines referenced. All other lines of source text are automatically deleted.

RESEQUENCE

This statement is used to renumber the sequence of lines referenced, using the range of line numbers specified.

DUPLICATE

This statement duplicates the sequence of lines referenced, using the range of line numbers specified.

MERGE

This statement directs the system to merge a specified sequence of programs that has been previously SAVED, into the program currently being worked upon.

## DESK CALCULATOR

The TSOS Desk Calculator is a conversational type routine that is designed to simulate a commercial desk calculator by the use of a remote terminal. Both arithmetic and elementary functions are provided. A general accumulator and 10 special accumulators can be accessed or displayed.

### Add Function (+)

This function adds a given constant or the contents of a specified accumulator to the general accumulator.

### Subtract Function (-)

This function subtracts a given constant or the contents of a specified accumulator from the general accumulator.

### Multiply Function (*)

This function multiplies the general accumulator by either a given constant or the contents of a specified accumulator.

### Divide Function (/)

This function divides the general accumulator by either a given constant or the contents of a specified accumulator.

### Load Function (@)

This function loads the general accumulator with either a specified constant or the contents of a specified accumulator.

### Store Function (=)

This function stores the general accumulator in the special accumulator specified. The special accumulators are named A0 thru A9.

### Restart Function ($)

This function resets conditions as they were before execution of the last line entered.

### Square Root Function (SQR)

This function computes the positive square root of the quantity in the general accumulator and stores the result in the general accumulator.

### Exponential Function (EXP)

This function raises 'e' to the power specified by the general accumulator and stores the result in the general accumulator.

### Cosine Function, Radians (COS)

This function computes the cosine of the quantity in the general accumulator and stores the result in the general accumulator.

| | |
|---|---|
| Cosine Function, Degrees (COD) | This function converts the quantity in the general accumulator from degrees to radians; computes the cosine of the converted quantity; and stores the result in the general accumulator. |
| Sine Function, Radians (SIN) | This function computes the sine of the quantity in the general accumulator and stores the result in the general accumulator. |
| Sine Function, Degrees (SID) | This function converts the quantity in the general accumulator from degrees to radians; computes the sine of the converted quantity; and stores the result in the general accumulator. |
| Tangent Function, Radians (TAN) | This function computes the tangent of the quantity in the general accumulator and stores the result in the general accumulator. |
| Tangent Function, Degrees (TAD) | This function converts the quantity in the general accumulator from degrees to radians, computes the tangent of the converted quantity and stores the result in the general accumulator. |
| Cosine Hyperbolic Function (COH) | This function computes the hyperbolic cosine of the quantity in the general accumulator and stores the result in the general accumulator. |
| Sine Hyperbolic Function (SIH) | This function computes the hyperbolic sine of the quantity in the general accumulator and stores the result in the general accumulator. |
| Tangent Hyperbolic Function (TAH) | This function computes the hyperbolic tangent of the quantity in the general accumulator and stores the result in the general accumulator. |
| Natural Logarithm Function (LOG) | This function computes the logarithm to the base 'e' of the positive quantity in the general accumulator. The result is stored in the general accumulator. |
| Log Base Ten Function (LGT) | This function computes the logarithm to the base 10 of the positive quantity in the general accumulator. The result is stored in the general accumulator. |
| Power Function (PWR N~N) | This function raises the positive quantity in the general accumulator to a power specified by either a given constant or a specified accumulator. The result is stored in the general accumulator. N may be from 1 to 15 digits. |

| | |
|---|---|
| Inverse Tangent Function, Radians (ATN) | This function computes the principal value of the inverse tangent of the quantity in the general accumulator. The result, in radians, is stored in the general accumulator. |
| Inverse Tangent Function, Degrees (ATD) | This function computes the principal value of the inverse tangent of the quantity in the general accumulator. The result, in degrees, is stored in the general accumulator. |
| Inverse Sine Function, Radians (ASN) | This function computes the principal value of the inverse sine of the quantity in the general accumulator. The result, in radians, is stored in the general accumulator. |
| Inverse Sine Function, Degrees (ASD) | This function computes the principal value of the inverse sine of the quantity in the general accumulator. The result, in degrees, is stored in the general accumulator. |
| Square Function (SQA) | This function squares the quantity in the general accumulator and stores the result in the general accumulator. |
| Reciprocal Function (REC) | This function computes the reciprocal of the quantity in the general accumulator and stores the result in the general accumulator. |
| Factorial Function (FAC) | This function computes the factorial of the positive quantity in the general accumulator and stores the result in the general accumulator. If the input 'n' is not an integer, the quantity n(n-1) (n-2)----P is computed and stored in the general accumulator; where 1<P<2. If n is less than one it is ignored. |
| Absolute Value Function (ABS) | This function computes the absolute value of the quantity in the general accumulator and stores the result in the general accumulator. |
| Repeat Function (REP N~N) | This function repeats the current entry, a specified number of times, beginning with the first character of the line. The repeat function must be the last entry on the input line. If a negative input for n  n is given, the function is ignored. The number may be from 1 to 15 digits. |

**Display Exponent Short (#ES)**

This function displays the general accumulator and any special accumulators requested in short floating-point format with 10 significant digits. The accumulators to be displayed are separated by commas. A display of consecutive accumulators is indicated by a dash. Accumulators must be requested in ascending sequence. The general accumulator is always displayed. Ex.: #ES, A0, A3-A5, A7, A8, A9.

**Display Exponent Long (#EL)**

This function is similar to the 'Display Exponent Short' function with the exception that 16 significant digits are displayed.

**Display Fixed Point (#NN)**

This function is similar to the 'Display Exponent Short' and 'Display Exponent Long' functions, with the exception that all displays are made in fixed point. 'NN' is the number of digits to be displayed from 01 to 16 rounded.

**Terminate (END)**

This function terminates the Desk Calculator routine.

## INTERACTIVE FORTRAN IV

**FORTRAN (Conversational Editor and Checker)**

The FORTRAN Editor and Checker provides syntax checking for the full FORTRAN IV language; the same language accepted by the Background FORTRAN IV Compiler. The editor and checker provide all the necessary interactive facilities to allow FORTRAN program construction in an interactive mode to allow subsequent compilation with the Background Compiler. The editing language also can be interspersed with FORTRAN statements in the interactive mode and can be executed interpretively. The facilities are reentrant, CLASS II programs.

**FORTRAN (Conversational Interpreter)**

The FORTRAN Interpreter provides terminal users with an immediate execution capability by interpretively executing any FORTRAN statements, debugging commands, and editing commands as they are retrieved from a file created by the interactive systems. Files to be processed interpretively must have been ordered by editing facilities.

The TSOS Interactive FORTRAN System combines the features of a conversational FORTRAN Compiler, a special FORTRAN text editing facility, a debugging language, and a desk calculator capability. This provides the FORTRAN user of the Spectra 70/46 TSOS with a comprehensive program preparation tool.

FORTRAN (Conversational Interpreter) (Cont'd)

The system consists of a single interpreter that accepts FORTRAN programs written using the full FORTRAN IV language, the same language accepted by the TSOS FORTRAN IV Background Compiler, and provides the following functions:

Complete syntax checking of the full FORTRAN IV language.

Comprehensive text editing of FORTRAN programs, which takes into account the structure of the FORTRAN language.

Immediate execution of FORTRAN programs from a remote terminal.

Powerful debugging commands.

A desk calculator facility for evaluating FORTRAN expressions entered from a remote terminal.

These facilities are obtained by combining the text editing commands, the debugging commands, and the FORTRAN language, itself, into an extended language, that is interpreted by the interactive FORTRAN System. It is envisioned that by means of these facilities a user will ordinarily construct his FORTRAN program, syntax check it, modify, test, and debug it, all in the interactive mode, and subsequently compile it using the Background Compiler.

## PROGRAM PREPARATION AND TESTING

### TSOS ASSEMBLY DIAGNOSTIC ROUTINE

The Assembly Diagnostic routine provides the user with post-assembly diagnostic information at a remote terminal. Thus, with this routine and the TSOS Text Editor facilities, a user is able to do all of his assembly work (that is, correct source code, reassemble and receive diagnostic data) from a remote terminal without waiting for any cards or printouts to be listed or punched and physically returned to him. To use the diagnostic routine, the user must use the TSOS Assembler and request that a diagnostic file be created during the assembly. When the assembly is completed, the user may invoke the diagnostic routine through the Executive System.

The diagnostic routine allows the user to request the following information:

Status - This is a summary giving the total number of lines in error and the number of errors associated with each error flag.

Line errors - This is a list of the statement numbers of all statements that were flagged in conjunction with their associated flags.

Flags - This is a list of all statement numbers that contain a given error flag.

Tags - This is a list of all symbols that are undefined and/or multiply defined; and, optionally, the cross-reference data for each symbol.

Cross reference - This is a list of all cross-reference data for each symbol supplied in the operand field.

Print - This is a list of statement numbers exactly as they appear on the Assembly listing.

Define - This is the definition of a specified error flag.

Help - This is a description of the diagnostic routine commands.

The Assembly Diagnostic routine is written as a conversational, shared program.

## COBOL SYNTAX CHECKER

The COBOL Syntax Checker provides the user with the ability to enter and check COBOL statements for syntax, a line at a time, from a terminal. Also, the statements can be retrieved and checked from a catalogued file on disc. This provides an error-free COBOL source program for presentation to the Background Compiler to produce object code.

Facilities are provided to correct (or ignore) each line as errors are displayed. Diagnostics are given for incorrect syntax. The proper syntax can be supplied and it is rechecked. Complete sentences can be reconstructed upon detection of error.

A complete COBOL source program or a division may be syntax checked.

### COBOL Syntax Checker Commands

ENTER

This command informs the Syntax Checker that the source program is comming from a terminal.

CHECK

This command informs the Syntax Checker that the source program is on an ISAM file on disc.

CORRECT

This command corrects lines and syntax checks in a division of a COBOL file existing on disc.

CEND

This command terminates the correct command.

MODIFY

This command modifies the contents of the current line number, increment, or interval counters.

IGNORE

This command ignores the current diagnostic (postpone correction).

COBOL Syntax Checker Commands (Cont'd)

WHY

This command is used to expand the diagnostic message.

HELP

This command supplies information about the syntax checker command(s).

STOP

This command is used to stop the session.

FILE

This command specifies the catalog name of the COBOL source file.

FILE EDITOR

The TSOS File Editor is a routine that enables the user to create, modify, and display catalogued program files interactively. It also can be used for other types of catalogued files.

The File Editor is designed to be used conversationally from a terminal so that the user may control processing a command at a time; it also can be used nonconversationally from a sequential file or system card reader.

The File Editor is comprised of twenty basic verbs, which, when invoked, perform editing functions on the contents of a file. These verbs also can be parameterized so that a series of records can be processed by one verb. A verb with its optional parameters can be referred to as a Command Statement.

A command statement can process a record or a series of records of one file. This file must be catalogued as an Index-Sequential File and must be available to the user in the Update mode. Hence, at the beginning of a File Editor session, the user must designate and open an existing ISAM file or request that a temporary ISAM file be catalogued and made available for the session. This file is called the Principal File. Moreover, the File Editor can read in another file sequentially and insert it into the principal file if the input records are of permissible length. Also, it can write a part of the principal file sequentially onto a suitable output file. These input and output files are referred to as Secondary Files. These files must be catalogued and made available through the Data Management System.

## File Editor Commands

ALTER

This command replaces a character string with another character string.

CHANGE

This command performs character by character editing.

DELETE

This command is used to delete a character string.

FIND

This command is used to locate a character string.

GET

This command copies part of a catalogued file into the principal file.

HALT

This command terminates the File Editor and returns control to the TSOS Executive command.

INPUT

This command is used to create new lines in the principal file from SYSDTA.

JUMP

This command causes a conditional transfer to another command.

LOOP

This command causes the repetitive execution of a command series.

MOVE

This command copies one part of the principal file to another part of the principal file.

NOTE

This command causes a SYSOUT display message that is used within procedures.

## File Editor Commands (Cont'd)

OPEN

This command opens a new file as the principal file and closes the current principal file.

PRINT

This command is used to write the principal file to SYSOUT.

QUALIFY

This command is used to open and identify a procedure file as input file.

RESET

This command allows the user to change File Editor variables.

SET

This command sets the value of the symbolic line address, the current line pointer, or a line content parameter.

TEXT

This command permits the user to scan a file and make changes or create new lines.

UPDATE

This command enables the user to insert a character string before or after a specific character in an existing line.

VERIFY

This command displayed the File Editor variables on SYSOUT.

WRITE

This command saves the principal file on a secondary file.

## INTERACTIVE DEBUGGING AID (IDA)

The IDA language of the Spectra 70/46 TSOS facilitates the checkout of new or revised programs. The IDA permits the user to check the progress of a program during its execution to localize the cause of trouble in a program, and to modify the program in the course of its execution. The user does not include source language debugging instructions in his program nor does he recompile his programs in order to utilize IDA.

Once the user has loaded his program, he can input IDA statements that refer to his program, then initiate execution. When the user is operating in the conversational mode, he can interrupt execution by pressing the break key at his terminal and then input further IDA commands and statements. He can then cause program execution to resume from its point of interruption by entering the RESUME command. Alternatively, the user can input dynamic IDA statements, prior to program initiation, specifying control points at which execution is to be stopped.

### IDA Commands

AT

This command defines a statement in the program to be debugged at which some command is to be executed at a later point in time.

DISPLAY

This command displays at the terminal the contents of a data field.

DUMP

This command displays on the debug file a large area of memory.

PROPAGATE

This command is used to fill a receiving data field with the contents of a sending data field.

IDA Commands (Cont'd)

IF

This command causes the conditional execution of the command used with AT and any other command.

MOVE

This command moves the contents of a data field to a receiving data field.

QUALIFY

This command provides for implicitly qualified symbols by specifying the LOAD MODULE and CSECT to which the symbols refer.

REMOVE

This command nullifies the dynamic delayed execution of AT statements.

RESUME

This command resumes execution of the object program.

SET

This command is used to change the contents of the data field to binary numeric value.

STOP

This command stops execution of the object program.

STATIC LINKAGE EDIT (MODIFIED TOS)

The Static Linkage Edit is an enhanced TOS routine that includes parameters for CLASS, PAGE, READ ONLY, and PUBLIC attributes; and a transcriber that produces a bound program on disc as well as tape. This routine operates as a Class I program.

Class

The Class control card indicates the class of the program to be recognized. The format of the Class card is as follows:

$$\triangle\text{CLASS}\triangle\begin{Bmatrix}1\\2\\E\end{Bmatrix}$$

Class 1 indicates a TOS program, Class 2 indicates a TSOS program, and Class E indicates an Executive overlay.

### ESD Card Information

The first card encountered in a module (that is, the ESD card containing the CSECT name that will become the name of the module) will be examined for information concerning PUBLIC, READ ONLY, and PAGE.

### Page

The Page control card indicates that a module is to start on a page boundary.

```
△PAGE△MODULE-name
```

### Read Only

The READ ONLY control card indicates to the Linkage Editor that the specified module is to be in pages marked as READ ONLY at program execution time.

```
△READONLY△MODULE-name
```

### Public

The Public control card indicates that a particular load is public (that is, shared) and is recognized and processed by the Linkage Editor.

```
△PUBLIC△LOAD-name
```

### Reset of Attributes

The previously described attributes can be reset by cards with the following format:

```
NOT   PAGE△module-name

      READONLY△module-name

      PUBLIC△Load-name
```

### V-CONS in TSOS

In TSOS, programs containing V-CONS will cause generated overlay control modules that will become pages separate from the main program.

### Transcriber

The Transcriber is the last pass of the Linkage Editor. Each program is automatically written to both tape and disc; both may be loaded and executed.

## TSOS LINKAGE EDITOR

The Linkage Editor is one of the TSOS service routines. It converts object modules to load module format, links object modules that have been compiled or assembled separately, and produces a loadable program consisting of one or more load modules.

Class I programs must be bound by the Linkage Editor before they can be loaded for execution. Class II programs, alternatively, can be loaded directly from object module format by the Linking Loader, if they are nontree structured in nature. Class II programs tree-structured overlays must be bound by the Linkage Editor prior to execution.

The TSOS Linkage Editor accepts three types of input:

1. Linkage Editor Control Statements.
2. Object Module Files (for example, Language Processor output).
3. Object Module Libraries (for example, System Library).

The TSOS Linkage Editor includes all of the functional capability of the TOS Linkage Editor.

### Linkage Editor Functions

The Linkage Editor can perform the following functions:

- Translate one or more object modules into load module format.
- Link two or more object modules to form one load module.
- Link two or more object modules in an overlay structure consisting of two or more load modules.
- Obtain object modules from secondary inputs.
- Search an object module library file to obtain modules that are then translated and linked into the load modules being produced. (This is an explicit search.)
- Rename entries and external references within an object module.

<u>Linkage Editor Functions</u> (Cont'd)

- Define entry names from an object module.
- Define a new standard entry point for an object module.
- Collect control sections with COMMON attribute and reserve space for them in the load modules. Both named and "blank" COMMON control sections are supported.
- Satisfy those unresolved external references, which are not explicitly excluded, by an automatic search of object module libraries, including the system library (SYSLIB).
- Replace, delete, or rename control sections within object modules.
- Change the attributes of control sections.
- Produce at the user's option, a load module map and/or a cross-reference listing of external definitions and external references.
- List all outstanding, unresolved, external references at the completion of linkage processing.
- Furnish diagnostic messages relating to errors and inconsistencies in the user's input.

## UTILITIES

### ACTIVATE/ DEACTIVATE

Class II programs have access to files stored on Model 70/568 Mass Storage Units. Access is provided through the Activate and Deactivate commands. It should be noted that Class I programs can also access 70/568 files directly by using the TOS FCP facilities supported by the TSOS.

The ACTIVATE command transcribes a file from a mass storage unit to a public or private volume on a direct access unit (disc or drum). After a file has been transcribed, it can be accessed using the TSOS Data Management System. Direct access files created by any of the access methods (including indexed sequential) can be transcribed using this command. The mass storage file specified in the Activate command must be catalogued and must have been transcribed to the 70/568 using the Deactivate command.

The Deactivate command transcribes a file from a direct access unit to a mass storage unit. The file to be transcribed, therefore, must be catalogued and a mass storage file must have been created and catalogued prior to issuing the Deactivate command.

These commands provide a flexible method for managing file space on public as well as private volumes. For example, when the file storage capacity on a direct access unit is exhausted, a user can issue the Deactivate command to transcribe a file to a mass storage unit. He can then issue an Erase command to release the space occupied by that file on the direct access device, and reclaim it for other purposes. T Thus, these commands provide a means for using the Spectra 70/568 Mass Storage Device as a lower-level storage medium for "inactive" or infrequently used files. At the user's option, the Deactivate command will create a magnetic tape "backup" copy of the file being transcribed. In fact, the user may specify that tape creation is the only transcription desired. If a file cannot be activated from the mass storage unit, a message will be written to SYSOUT. The conversational user could then reissue the Activate command, specifying an input file that resides on magnetic tape (backup volume).

**LIBRARY MAINTENANCE ROUTINE**

This routine can be used to create or update object module libraries. The routine maintains the directory that is included in every library to speed searches made by the Linkage Editor and the Dynamic Linking Loader.

**SORT/MERGE ROUTINE**

The TOS/TDOS Sort Merge Generator runs under TSOS as a Class I program routine. This includes the Tape Sort and the Disk Sort routines.

The same capabilities afforded under TOS/TDOS are available under TSOS. They include:

1) Generation of a tailored Sort Merge program based on user-supplied parameters.

2) Inclusion of own-code routines.

**PERIPHERAL ROUTINES**

The various TOS/TDOS Peripheral routines can be run under TSOS as Class I program routines.

**SELF-LOADING DEVICE EDIT**

The 70/46 Self-Loading Device Edit routine is primarily a debugging aid and a program testing tool. It gives the user an easy method of examining information read from, or written to, a random access device or a magnetic tape by providing the ability to edit and print all of, or selected portions of, the information contained on these devices. It is self-loading and completely self-contained, including all necessary routines to perform input/output operations at the hardware level. Thus, the Self-Loading Device Edit routine does not require services from the Executive or Data Management routines, and can be run independently of all other software. Furnished as a card deck, it is loaded from the card reader and is controlled by parameters entered from the console typewriter or the card reader. The output may be to the on-line printer or to a magnetic tape for later printing.

The Self-Loading Device Edit is supplied as an Assembly Language source deck for assembly with the TSOS Assembler or with 70/45 ASMDOS. The source deck contains a two-card Absolute Loader that will be reproduced by the Assembler under control of a PUNCH pseudo-op; the balance of the program will be produced in the form of a text card object deck. The program

SELF-LOADING DEVICE EDIT (Cont'd)

occupies approximately 7,850 decimal memory locations. All of the remaining high-speed memory is used as the input area.

The function of the 70/46 Self-Loading Device Edit is to edit the contents of a random access device or a magnetic tape and to output this edited data to a printer or to a magnetic tape, as indicated by input parameters. The program contains the following options:

- Output to printer or output to tape.
- Input from random access device (70/564 Disk Unit, 70/565 Drum, 70/567 Paging Drum, or 70/568 Mass Storage Unit) or input from magnetic tape (seven- or nine-level).
- Graphic representation of the data, hexadecimal representation of the data, or both graphic and hexadecimal (combination) representation of the data.
- Rewind or No Rewind of the input tape.

Because this routine loads itself into low memory that may have been occupied by the control program, the control program must be reloaded at the end of a Self-Loading Device Edit run.

To terminate the edit before the entire requested area has been printed, press COIN: the program will then ask for the next edit parameter.

For random access input, the buffer size (5,193 bytes) is large enough to handle the largest paging drum record. For magnetic tape input, the routine creates two buffers by dividing the first 131K of available memory into two equal-size buffers for double-buffering purposes.

The random access portion of the Device Edit allows the editing of up to forty records on a single track. Any records after the fortieth are not printed and the program proceeds to the next track, parameter permitting.

## SELF-LOADING MEMORY EDIT

The 70/46 Self-Loading Memory Edit routine is intended for use by the operator as an emergency measure when, for example, an unexpected program stop occurs. The routine is self-loading and completely self-contained, including all necessary routines to perform input/output operations at the hardware level. Thus, the Self-Loading Memory Edit does not require services from the Executive or Data Management routines, and can be run independently of all other software. Furnished as a card deck, the routine is loaded from the card reader and initiated from the console typewriter. It can be floated and loaded into any part of memory. It dumps all or selected portions of memory to the on-line printer or to a tape for later off-line printing.

The 70/46 Self-Loading Memory Edit will provide a listing of scratchpad memory, translation memory, and high-speed memory. Listings are in hexadecimal with graphic equivalents.

The routine is provided to the user as a condensed five-card Relocating (floating) Loader, followed by an object deck in the form of standard TSOS TXT cards. The first $160_{10}$ bytes of memory are occupied by the loader. The object program is relocatable (floatable) and occupies approximately one page (4,096 bytes) of memory. All parameters are entered through the console typewriter.

### Scratchpad Memory

The program provides an edited listing of the following sections of scratchpad memory:

1. General Registers: 0-15 for program state P1.

2. General Registers: 0-15 for program state P2.

3. General Registers: 0-15 for program state P3.

4. General Registers: 0-15 for program state P4.

5. I/O Channel Registers for selector channels 1, 2, and 3.

Scratchpad Memory (Cont'd)

6. Floating-Point Registers 0, 2, 4, 6.

7. Storage Keys for 262K memory.

Note: Alphabetic X's are printed instead of those P3 state General Registers (0, 1, 2, 3, and 15) that were destroyed by the load function.

Translation Memory

The program edits all 512 halfwords of translation memory, breaking each halfword into its components - the W, G, U. S, E, M bits, the Real Page Address, and the H bit.

High-Speed Memory

The routine edits and prints the contents of high-speed memory as specified by the user. The print-out is provided in fullword hexadecimal with graphic equivalents, 48 bytes on each print line.

A print line is separated into three groups of four words. Each group is preceded by the address of the first byte of that group. All duplicate lines are suppressed, and an asterisk is placed in the preceding line.

Output Device

The output of the 70/46 Self-Loading Memory Edit can be assigned to the on-line printer or to a magnetic tape (seven- or nine-level). If used, the tape must be manually rewound before loading the program and will not be rewound at program termination.

Output on magnetic tape can be printed by using the pre-edit option of the 70/25 or 70/35-45-55 Tape Edit Program.

When output is to magnetic tape, one tape mark is written after the program has satisfied each input parameter. This facilitates the printing of multiple edits on one tape. The terminate parameter causes a double tape mark to be written to end the tape.

TSOS SYSTEMS GENERATOR/ INITIATOR (SG/I)

The TSOS SG/I is an RCA supplied program routine. It is used to adapt or tailor TSOS to the specific user equipment configuration. Also, it selects those programming components that are required for the user's particular processing requirements by generating a Control Program that is maintained on a system resident disk.

**TSOS SYSTEMS GENERATOR/ INITIATOR (SG/I) (Cont'd)**

There are three distinct phases to SG/I:

1. BASIC System Preparation - This phase creates a Basic TSOS on a user-specified disk. This phase is optional if there is a pre-existing system. During this phase the Basic System is transcribed to the user disk from the RCA-supplied tape by means of the Hardware Load function and a special program called INALTRAN producing the disk known as SYSRES.

2. User System Generation - This phase performs the following primary functions:

   a. Constructs the table of physical device addresses that describe the particular configuration.

   b. Constructs a parameter table used in loading the CCM.

   c. Organizes a file consisting of the required systems programming components into a bound control program on SYSRES.

3. User System Loading and Initialization -

   This phase actually loads the control program into physical and virtual memory from SYSRES.

   a. Paging drum is initialized here.

   b. Physical page table and translation memory initialized.

   c. CCM memory loaded terminal lines put in service.

**VOLUME INITIALIZER**

The Random Access Volume Initializer is a routine that prepares random access volumes for use with the TSOS. A volume is a 70/564 disc pack, a 70/565 drum, or a 70/568 magazine.

Only the systems operator is permitted to use this routine. The Volume Initializer is a Class I program routine that runs under the TSOS Control routines.

VOLUME INITIALIZER (Cont'd)

A surface analysis is made of each volume and alternate tracks are assigned for any defective tracks found. The testing is accomplished by checking sense bytes returned when the track is written and immediately read back. In the case of the 70/568, the read back is done by hardware.

Home Address (HA) records and Track Descriptor records (RØ) are written on each track. The Home Address provides a safeguard against potential improper head selection by the hardware. Also, a flag byte is contained within the Home Address record. It identifies the track as defective/good and as primary/alternate.

A preformatted Volume Table of Contents (VTOC) is established for later use by the Data Management System (DMS). The VTOC is essentially an index of the files that reside on the volume. Since the VTOC itself is not confined to specific cylinders, there must be a way to locate it. The Standard Volume Label (SVL), which is fixed at Cylinder Ø, Track Ø, Record 3, fulfills this function. Other information relating to the initialized volume is also contained in the SVL.

Another function of the Random Access Volume Initializer is to create a dummy Initial Program Loader (IPL) to thwart any attempt to load the system's control routines from the volume.

## HARDWARE MAINTENANCE ROUTINES

### DISC HARDWARE CHECK ROUTINE

The TSOS Disc Hardware Check routine (DISC HCR) is part of a family of TSOS on-line peripheral test routines. These programs reside with the TSOS software and run, on a demand basis, as Class I user programs.

This multipurposed program routine is intended for use by hardware maintenance personnel. It is to be used to test the 70/564 Disc Storage Unit without resorting to system shutdown. In addition to this, the DISC HCR provides troubleshooting aids and confidence tests for the disc and closely related equipment.

### BASIC PROCESSOR UNIT EXERCISER

The primary function of the BPUEXR is to exercise the internal logics associated with those instructions performing an arithmetic function on the 70/46 Basic Processor, concurrent with other nonconversational programs and conversational programs.

The BPUEXR operates on line under the TSOS Executive as a Class II, Permanent Task, and nonconversational program. This program is not reentrant and is activated to run at intervals indicated by the operator command, BPURUN.

The objective of the BPUEXR is to detect arithmetic instruction malfunctions under multiprogramming, time-shared conditions.

### CARD PUNCH PROGRAM ROUTINE

The TSOS Card Punch Program routine (PCHTST) determines the operability of the Model 70/234 or 70/236 80-column, row-oriented Card Punch and its control electronics are functioning properly.

The PCHTST routine operates under the control of the TSOS Executive System, as a Class I (TOS and TDOS compatible) user's program utilizing TOS physical level FCP.

**CARD READER PROGRAM ROUTINE**

This routine can be used to test all device functions of the Model 70/237 Card Reader. The routine includes special hardware testing functions.

**PRINTER DEVICE SEQMENT ROUTINE**

The Printer Device Segment routine tests the hardware functions within the 70/242, 243-10, and 243-20 line printers. In the event of a malfunction, this program will diagnose the error condition and report (by way of the console typewriter) information necessary to correct this condition.

This routine also provides the capability for on-line troubleshooting and preventive maintenance.

**SEVEN-LEVEL TAPE HARDWARE CHECK ROUTINE**

This routine is used to test seven-level 70/432-442, and -445 Magnetic Tape Units when these units are connected to a Model 70/473 Tape Controller. This routine is used by hardware maintenance personnel; system shutdown is not required.

The routine is run as a Class I user program. It uses physical level FCP and creates I/O situations both normal and abnormal, and interprets the results from the I/O operation.

**NINE-LEVEL TAPE STATION HARDWARE CHECK ROUTINE**

This routine is used to test nine-level Model 70/432, -442, and -445 Magnetic Tape Units. This routine is used by hardware maintenance personnel; system shutdown is not required.

The nine-level tape routine is run as Class I user program. It uses physical level FCP and creates I/O situations both normal and abnormal, and interprets the results from the I/O operation.

## TIME-SHARING SCIENTIFIC APPLICATIONS

The following scientific applications are written completely in FORTRAN IV and currently operate on a Spectra 70/45 using the present FORTRAN Compiler. These programs can be recompiled to produce object decks that operate efficiently on the Model 70/46 as Class I programs.

### STATISTICAL SYSTEM

The Spectra 70 Statistical System consists of a series of statistical programs and a monitor system. The monitor provides unified operating procedures, unified input/output, unified error recovery procedures, and automatic sequential processing of problems. Variables can be expressed as single- or double-precision variables. The following program routines are presently part of the system. Because the system is open-ended, additional programs can be added.

#### Anaylsis of Variance

This program routine helps analyze repetitive experiments. The numerical results of an experiment serve as the input to the program while the output is an "analysis of variance table." The table helps the user to make probability statements about the existence of unknown effects in an experimental situation. The program can handle six classifications and N observations, where N satisfies

$$N = \frac{\leq S-45,000}{6}$$

#### Factor Anaylsis

The Factor Analysis program routine accepts information describing a set of variables and reduces the number of variables necessary to describe a situation if mathematically possible. The routine also can be used to find the eigenvalues and eigenvectors of a real, symetric matrix. The routine can handle a correlation matrix of order as high as N, where N satisfies

$$S-40,000+4N(N+1) \qquad (S=\text{Memory Size})$$

**Regression and Correlation**

The Regression and Correlation program routine is designed to analyze large amounts of independent observations of sets of variables. The analysis is performed to discover inferred or suspected relationships among the measured variables. The analyst hypothesizes a linear function with unknown coefficients to describe the relationships between the variables. The program, in a stepwise manner, chooses the variables and the coefficients of the variables that best fit the hypothesized linear function.

**Nonlinear Regression**

Some problems cannot be reduced to a linear function and therefore cannot be solved by linear regression. Nonlinear regression would be employed where the hypothesized equation is of a nonlinear nature. Almost any equation that can be expressed in FORTRAN can be used.

FUNCTION MINIMIZER

This program routine accepts as input a function of many independent variables and by iterative procedure finds the values of the independent variables that minimize the function. The program accepts any function that can be expressed as a FORTRAN IV subprogram and can be expressed within available memory.

**Description and Tabulation**

Simple Data Description

This routine computes simple averages and measures of dispersion of the variables, omitting values that the user specified.

Correlation with Transgeneration

This routine computes correlation coefficients, averages, and measures of dispersion on entering variables and/or transgenerated variables from selected cases.

Correlation with Item Deletion

This routine computes a correlation matrix omitting values of variables that the user specifies to be deleted.

Description and Tabulation (Cont'd)

Asymetric Correlation with Missing Data

This routine computes large correlation matrices or subsets of large correlation matrices from data with possibly missing values.

Alphanumeric Frequency Count

This routine computes frequencies of legal characters on one-column data from cards or magnetic tape.

General Plot Including Histogram

This routine provides a method by which graphs and histograms can be produced.

Description of Strata

This routine separates the cases into groups based on specified intervals of the conditioning variable. For these selected groups, computations are performed for specified conditioned variables.

Description of Strata with Histograms

This routine groups the data into a specified number of groups and prints histograms for each variable in the groups. The number of classes or categories of the histograms can be specified, or computed by the program.

Cross Tabulation with Variable Stacking

This routine computes two-way frequency tables of data input. Frequency tables are computed from specified ranges of the original variables, variables after transgeneration, stacked variables, or combinations of these.

Cross Tabulation, Incomplete Data

This routine performs cross-tabulations of input data, excluding specified special values or codes used to designate missing values.

Description and Tabulation (Cont'd)

Data Patterns for Dichotomies

This routine finds frequencies and patterns of any one particular specified code in the input data. The program uses 0's to designate the specified code or missing values, and 1's to designate other values.

Data Patterns for Polychotomies

This routine prints patterns of one-column data and item numbers to identify cases having these data patterns.

Mcan Substitution

This routine replaces missing values of a variable by the mean of its non-missing values. It will handle up to 5,000 variables and any number of cases.

MULTIVARIATE ANALYSIS

Principal Component Analysis

This routine computes the principal components of standardized data and orders each case by the size of each principal component separately.

Regression on Principal Components

This routine computes the principal components of standardized data. Each dependent variable is then regressed on the first, first two, first three, and all principal components.

Identification of Outliers

This routine defines the multivariate data for outliers by computing the Mahalanobis distance of each case from the center of the distribution of the remaining cases.

Factor Analysis

This routine performs a factor analysis of up to 198 input variables. The factoring can be done using either covariance or correlation matrices.

Canonical Analysis

This routine computes the canonical correlations between two sets of variables.

MULTIVARIATE ANALYSIS

Discriminant Analysis for Two Groups

This routine computes a linear function of given variables measured on each individual of two groups. This function serves as an index for discrimination between two groups.

Discriminant Analysis for Several Groups

This routine computes a set of linear functions for classifying an individual into one of several groups.

Stepwise Discriminant Analysis

This routine performs a multiple discriminant analysis in a stepwise manner. At each step one variable is entered into the set of discriminating variables.

REGRESSION ANALYSIS

Simple Linear Regression

This routine performs simple linear regression analysis on single or combined treatment groups with unequal sample sizes.

Stepwise Regression

This routine computes a sequence of multiple linear regression equations in a stepwise manner.

Multiple Regression with Case Combination

This routine performs multiple regression and correlation analyses on the data within selected subsamples from the same population.

Polynomial Regression

This routine computes a polynomial regression of the form:

$$Y = \alpha + \beta_1 X + \beta_2 X^2 + \ldots. + \beta_k X^k \quad (K^{th} \text{ degree})$$

where K is some positive integer.

REGRESSION
ANALYSIS
(Cont'd)

Periodic Regression and Harmonic Analysis

This routine performs a periodic or harmonic regression analysis using the regression function of the form:

$$Y_t = a_0 + \sum_{i=1}^{\eta} a_i \cos (2\pi it/k + b_i \sin (2\pi it/k)$$

Amplitude and Phase Analysis

This routine computes the amplitude and phase of wide-band noise contaminated by extraneous noise.

Autocovariance and Power Spectral Analysis

This routine performs the autocovariance and power spectral analysis of a given number of time series.

Multiple Time Series Spectral Analysis

Beginning with a sequence of cross spectral matrices, this routine estimates multiple coherence functions and frequency response functions between a set of input and a set of output series.

Multiple Time Series Spectral Estimation

This routine estimates auto-spectra, cross spectra, and coherences for up to 100 stationary time series.

Time Series Spectrum Estimation

This routine estimates auto-spectra, cross spectra, and coherence for stationary time series. Each series is decomposed into its component frequencies by evaluating its finite Fourier transform.

Mean Frequency Epoch Analysis

This routine partitions a time series into epochs of specified size, computes a mean power and frequency for each epoch and plots them against time.

VARIANCE ANALYSIS

Analysis of Variance for One-Way Design

This routine computes an analysis of variance table for one variable of classification.

Analysis of Variance for Factorial Design

This routine computes an analysis of variance for a factorial design.

Analysis of Covariance for Factorial Design

This routine performs a full analysis of covariance.

Analysis of Covariance with Multiple Covariates

This routine computes the analysis of covariance for one analysis of variance variable with multiple covariates and unequal treatment group sizes.

Multivariate Analysis of Variance and Covariance

This routine handles arbitrary, complete, balanced analysis of variance and analysis of covariance designs with an arbitrary number of covariates.

Analysis of Covariance

This program routine performs a one-way analysis of covariance, using one or more covariates. Group sizes can be unequal and parallel analyses can be performed using several dependent variables.

Multivariate General Linear Hypothesis

Given a multivariate regression model of the form $Y = X\beta + E$, this routine computes estimates for the matrix $\beta$ and the covariance matrix of Y. In addition, U-statistics and, when possible, F-statisitcs are computed for arbitrary hypotheses of the form $A\beta C = D$, which are specified by the user.

General Linear Hypothesis

This is univariate, general, linear hypothesis routine that makes the analysis of missing data, analysis of variance, and analysis of covariance designs very simple. By means of very simple specifications this program automatically creates the dummy variables

VARIANCE ANALYSIS (Cont'd)

required for any complete balanced analysis of variance of covariance design with or without missing data.

General Linear Hypothesis with Contrasts

This routine is similar to the General Linear Hypothesis routine but it also estimates and tests the statistical significance of the parameters that occur in the general linear hypothesis model.

TEXSM

This routine generates the triple exponential smoothing series of a given series.

DISC 1

This routine computes means of variables in each group and a poled dispersion matrix for all the groups.

DISC 2

This routine computes a set of linear functions that serve as indices for classifying individuals into particular groups.

LGRAN

This routine computes $Y_2 + (x)$ for a discrete set of X and Y values.

FORIR

This routine calculates the Fourier coefficients of order M for a given tabulated function f (x) where $0<X<2\pi$ over the interval $2\pi/_{2N+1}$.

RKI

This routine integrates a first order differential equation.

FRES

This routine computes the Fresnel integral.

DTM

The Digital Terrain Model Location System, a highway design program, is a method of recording a terrain data and an associated set of integrated programs that work with that terrain data and a roadway design, specified by the engineer.

**COGO**

COGO is a Civil Engineering problem-oriented language that can be used in Control Surveys, Land and Right-of-Way Surveys, Highway and Interchange Design, Construction Layout, and Bridge Geometry.

**TRAFFIC SIGNAL PROGRESSION**

This program routine performs all the necessary computations to determine the phasing of the signals and the green bandwidths.

**SALES FORE-CASTING AND CONTROL SYSTEM**

The Sales Forecasting program routine employs a technique of exponential smoothing that requires that only previous sales history data be given. This method estimates the basic components of the sales data and combines these to forecast sales for one or more periods. The unique feature of this package is that, in addition to using a complete forecasting model (including seasonal and trend effects), it utilizes a powerful searching technique to select optimal model parameter values for each product. Going one step further, the program employs a control phase that automatically re-evaluates and adapts this optimal model.

**Special Complex and Real Matrix Subroutines**

REINSK

This subroutine inverts a real matrix, skipping any rows and columns.

RELINV

This subroutine inverts a real matrix.

RESOSK

This subroutine solves an equation AY = X for Y when A and X are real, with the capability of skipping rows and columns of A and X.

RELSOL

This subroutine solves an equation AY = X for Y when A and X are real.

RESHSO

This subroutine shrinks a real vector, leaving out any elements.

Special Complex and Real Matrix Subroutines (Cont'd)

REEXSO

This subroutine expands a real vector inserting a zero element in any location.

RELEXP

This subroutine expands a real matrix, inserting the value 1 x $10^{35}$ in any rows and columns.

COINSK

This subroutine inverts a complex matrix, skipping any rows and columns.

COMINV

This subroutine inverts a complex matrix.

COSOSK

This subroutine solves an Equation AY = X for A and X are complex, with the capability of skipping rows and columns of A and X.

COMSOL

This subroutine solves an equation AY = X for Y when A and X are complex.

COSHSO

This subroutine shrinks a complex vector, leaving out any elements.

COEXSO

This subroutine expands a complex vector, inserting a zero element in any location.

COMSHK

This subroutine shrinks a complex matrix, leaving out any rows and columns.

COMEXP

This subroutine expands a complex matrix, inserting the value 1 x $10^{35}$ in any rows and columns.

## General Matrix Subroutines

### CSUM

This subroutine sums the elements of each column to form a row vector.

### RSUM

This subroutine sums the elements of each row to form a column vector.

### SDIV

This subroutine divides each element of a matrix by a scalar.

### SMULT

This subroutine multiples each element of a matrix by a scalar.

### SSUB

This subroutine subtracts a scalar from each element of a matrix.

### SSAD

This subroutine adds a scalar to each element of a matrix.

### MADD

This subroutine adds two general matrices.

### MSUB

This subroutine subtracts a general matrix ß from general matrix A forming a general matrix C.

### MMULT

This subroutine multiples two general matrices forming a resultant general matrix.

### MTRAN

This subroutine transposes a general matrix.

### MTINV

This subroutine inverts a matrix.

## General Matrix Subroutines (Cont'd)

### COMEQS

This subroutine solves complex simultaneous equations.

### BCT

This subroutine compiles a table of binomial coefficients for every nonegative integer power up to the power specified through 130 and stores the resulting two-dimensional triangular array as a one-dimensional array.

### MATINV

This subroutine inverts a matrix.

## Complex Arithmetic Subroutines

### ZMPY

This subroutine multiplies two complex numbers.

### ZDIV

This subroutine divides two complex numbers.

### ZSQRT

This subroutine finds the square root of a complex number.

### ZSIN

This subroutine finds the sine of a complex number.

### ZEXP

This subroutine finds the exponential of a complex number.

### POLAR

This subroutine finds the polar coordinates of a complex number.

### RKI

This subroutine integrates a first order differential equation.

**Complex Arithmetic Subroutines** (Cont'd)

**FRES**

This subroutine computes the Fresnal integral.

**TEXSM**

This subroutine generates the triple exponential smoothing series of a given series.

**FORIR**

This subroutine calculates the Fourier coefficients or order M for a given tabulated function f (X) where $0 \leq X \leq 2$ over the interval $2\pi/2N+1$.

**RANDOM**

This subroutine produces uniformly distributed random numbers.

**NORRAN**

This subroutine computes normally distributed real random number.

RCA
Information
Systems

CAMDEN, N.J. 08101