File No. 1710-36
Form C26-5879-1

IBM Systems Reference Library

IBM 1710 FORTRAN Executive System Reference Manual

The 1710 FORTRAN Executive System provides the user with the ability to direct process control operations with programs written in the FORTRAN language.

This publication supersedes and obsoletes the following IBM publications:

IBM 1710 FORTRAN Executive System Specifications (Form C26-5733)
IBM 1710 FORTRAN Executive Assembly Program Specifications (Form C26-5800)

This edition obsoletes the previous edition (Form C26-5879-0) which was a limited printing for the initial distribution of the programming system.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Comments concerning the contents of this publication may be addressed to:
IBM, Product Publications Department, San Jose, Calif. 95114

CONTENTS

PREFACE

The IBM 1710 FORTRAN Executive System is designed to provide the user with a programming tool that will simplify the creation of user-written process-control programs. The language is primarily FORTRAN with the ability to call Executive control programs such as Analog Input, Analog Output, Contact Sense, etc. In addition, provision for entry to these control programs when using a one-for-one type programming language is supplied with the FORTRAN Executive Assembly Program (FEAP). This program is specially designed to aid the FORTRAN user if he wishes to incorporate his own control programs or to produce FORTRAN sub-programs written in this language.

When the 1710 System is not connected to the process, the FORTRAN Executive System can be operated in a manner similar to the 1620 Monitor I System.

The FORTRAN Executive System can be separated logically into the following five programs:

FORTRAN II-D Compiler
Executive Programs
Supervisor Program
Disk Utility Program
FORTRAN Executive Assembly Program

The material in this publication is organized and presented under these headings. Appendices are included which contain: (1) Listing of FORTRAN Statements; (2) Listing of FEAP Mnemonics; (3) Executive Summary and Loading Sequence; (4) Starting procedures; (5) Error Message Index; (6) Permanent System Symbol Table Listing.

To fully understand the material in this publication, the reader should be familiar with the information contained in the following IBM publications:

1710 Control System Reference Manual (Form A26-5709)
1710 Additional Special Features and Attached Units (Form A26-5660)
1311 Disk Storage Drive, Model 3 (Form A26-5650)
1620 Monitor I System Reference Manual (Form C26-5739)

MACHINE REQUIREMENTS

The FORTRAN Executive requires that the following features be installed on the user's 1710 Control System with the 1711 Data Converter, Model 2.

1. IBM 1311 Disk Storage Drive, Model 3
2. Indirect Addressing
3. Automatic Divide
4. Basic Interrupt
5. Analog Output Setup Interrupt (optional)
6. Serial Input/Output Channel (optional)

INTRODUCTION

The FORTRAN Executive System is a programming system designed to provide users with a simpler means of generating, organizing, and executing process-control programs than those previously available. With this system, user's programs may be written in FORTRAN language or symbolic form; they may use the supplied Executive Control programs; and they can be compiled and/or assembled and stored to disk storage under control of a Monitor.

The FORTRAN Executive System is comprised of five programs:

Supervisor Program
Disk Utility Program (DUP)
FORTRAN II-D Program
FORTRAN Executive Assembly Program (FEAP)
Executive Control Program(s)

The Supervisor, DUP, FORTRAN, and FEAP programs operate as an off-line Monitor and can operate only when the 1710 System is not being used to control the process. The Executive Control Programs, executed during operation of the user's process control programs, are designed to provide the user with an efficient method of using the 1710 features. Calls from FORTRAN programs to the Executive Control Programs are handled by CALL statements.

General Operation

The implementation of the FORTRAN Executive System is accomplished in four phases.

Phase 1. Phase 1 consists of loading the Monitor programs to disk storage and using the FORTRAN Executive Assembly program to assemble the Executive Control programs. The Executive programs must be assembled by the user because only he can define certain parameters that are required by these programs. For example, the Serial Input/Output Control program must know the total number of SIOC units that are attached to the system. This must be defined for the SIOC program when it is assembled. After assembling the Executive programs needed, the Skeleton Executive program must be loaded to disk using the special loader routine provided for this purpose.

Phase 2. Phase 2 consists of using the FORTRAN compiler or the FORTRAN Executive Assembly program to compile or assemble user-written programs. Input may be from cards, paper tape, or typewriter; and output may be in cards, paper tape, or directly to disk storage. Compilation or assembly is done in the noninterruptible mode.

After compiling or assembling, the object program is in relocatable form, i.e., the addresses of the instructions must be modified relative to the program starting address. The modification must be performed before the program can be executed.

Phase 3. Phase 3 consists of using the Disk Utility Program to load the object programs (output from 1710 FORTRAN II or FEAP) to disk storage.

To facilitate rapid loading for execution, DUP combines process control programs into groups called "core loads." A core load consists of the core image form of a main program and the in-core subprograms and interrupt programs that it utilizes. The object programs to make up a core load may be loaded from cards, paper tape, or from disk storage itself. Core loads are always kept in disk storage and the relocatable form of the programs that were used to make up a core load may also reside on disk.

Phase 3 is completed when all programs and subprograms have been loaded to disk storage. The programs are in executable form, so that when they are called from disk storage to be executed, no time will be lost adjusting addresses.

Phase 4. This is the execute phase. The operator first calls in Program Schedule Control. This routine loads the:

Multiply and Add tables
FORTRAN Arithmetic and Input/Output routines
Skeleton Executive program
Core portion of the Input/Output routine

The user specifies the desired program, and the core load is loaded into core storage. After the core load is loaded, a branch occurs to the starting address of the user's main program. From this point the 1710 Control System is capable of controlling the process. The degree of control that the computer maintains over the process is dependent only on the machine configuration and the user's programs.

The following section is included for the benefit of IBM and customer personnel who are familiar with the IBM 1620 Monitor I System and/or the IBM 1710 Executive II System.

The following differences should be noted:

1. The FIND and FETCH operations permit reference to any portion of the disk storage drive that contains the work cylinders.
2. All arrays are written from or read into core storage as one record when RECORD or FETCH statements are used.
3. FIND is no longer a relocatable library function. This routine is stored with the arithmetic and input/output subroutines.
4. The following library subroutines can be specified as "in-core" or used in the arithmetic and input/output block.

LOGF	SINF-COSF
EXPF	ATANF
Subscripting	SQRTF
Disk I/O	

5. The arithmetic and input/output subroutines have been permanently placed on cylinders 0-6.
6. The standard work cylinders area on disk has been changed to 8 through 23.
7. LOCAL subprograms, previously stored in the work cylinders (temporary storage), are stored permanently on disk storage in core image format.

   Names for LOCAL subprograms are restricted to five characters or less. Up to ten copies of the same subprogram in core image format can be stored on disk storage and used by mainline programs that call them. They will be used if they can fit into the core area for LOCAL subprograms in the mainline core load. If no copy meets this requirement, a new copy is created and stored on disk.

   LOCAL subprograms that call other subprograms are unique and are identified as such. They can be used only with the first mainline program that calls them. If another mainline requires the same subprogram, the subprogram must be renamed and loaded again, for a second unique copy.

   LOCAL subprograms used with mainline core loads must be executed in the masked mode if in-core interrupt subprograms that call LOCAL subprograms are included in the mainline core load.
8. Interrupt routines can be loaded into core storage with mainline programs by specifying the interrupt identification in an INTCR control record.
9. Interrupt core loads can include all normal FORTRAN II-D core load data; e.g., subprograms-in-core, FORTRAN library functions, LOCAL subprograms.
10. The reading of control records when loading FORTRAN programs is controlled by a CCEND control record instead of a control card count.

11. Programs being compiled for off-line (nonprocess control) purposes must include an OFFLN control record.
12. The floating-point mantissa length can be varied from 2 through 8. The mantissa length and fixed-word length must be the same for all process-control programs. Fixed word length may be from 4 through 10.
13. The Mainline Core Load Map is stored in the first three sectors of the mainline core load.
14. When called from an interrupt routine, the SIOC, ADC, and AO must have an I suffix, e.g., SIOCI.
15. All programs that use COMMON must reserve two fixed-point variables for system use. These must be the first entries of COMMON. The first such variable should be loaded with zeros initially if it is to be used by the user's program.

COMMUNICATIONS AREAS

There are several areas of disk storage and core storage that will be used as communications areas by the user's programs and the programs that make up the FORTRAN Executive System.

The user must specify various sequences, priorities, address information, and error restart procedures. These items are then automatically placed in "maps" or tables, which permit the FORTRAN Executive System to be self-directing with no mandatory operator interventions. A map is simply a series of records.

A description of each of the communications areas follows.

Disk Identification Map (DIM) Table

The DIM table contains an entry for each program or data area stored in disk storage. It resides on cylinder 24 and can accommodate up to 999 20-digit entries. The Disk Utility Program maintains the DIM table.

The format of the 20-digit DIM entry follows:

$$D\bar{D}DDDD\bar{S}SS\bar{C}CCCC\bar{E}EEEE\ddagger$$

$D\bar{D}DDDD$ is the disk sector address of the program or data.
$\bar{S}SS$ is the sector count.
$\bar{C}CCCC$ is the first core address. If this field is all 9's, the program is in relocatable format. If the units position is flagged, the FORTRAN loader is used to load the program.
$\bar{E}EEEE$ is the entry address. This address is relative to the load address (first core address to be loaded) for programs in relocatable format.
$\ddagger$, $\bar{\ddagger}$, $\ddagger$, or $\bar{\ddagger}$ is the rightmost character of a DIM entry.
These characters indicate the following conditions about a referenced program.

Character	File Protected	Permanently Assigned
$\ddagger$	Yes	No
$\bar{\ddagger}$	Yes	Yes
$\ddagger$	No	No
$\bar{\ddagger}$	No	Yes

NOTE: (1) A file-protected program can be read but not written because it has read-only flags written in all (or any one) of the sector addresses in the disk area which it occupies. If read-only flags are written by a user's program in sector addresses where a program is stored, that disk storage area will not be "file protected," however, data cannot be written in individual sectors which contain read-only flags. (2) A permanently assigned program cannot be repositioned (moved) in disk storage because it has been assigned to a given address by the user.

DIM entries are located in the following manner:

1. Refer to the Equivalence table to find the 4-digit DIM entry number.
2. Double this number and add the sum to 048000.
3. Use the leftmost five digits of the result to locate the disk sector of the DIM table.
4. Use the rightmost digit of the result to find the particular DIM entry.

The DIM table can be expanded (see DEFINE PARAMETER ROUTINE) to contain up to 4995 DIM entries.

Sequential Program Table

The second through eighty-first sectors (80 sectors) of cylinder 99 are reserved on each disk pack for a Sequential Program table which lists the programs, tables, and data areas sequentially by DIM numbers, and available storage space by special coding. The Sequential Program table is used by the Disk Utility program to determine the order of programs and available storage space. When a program is added to or deleted from disk storage, the table is updated to reflect the new sequence. Each 80-sector table will accommodate up to 2000 4-digit entries. Three types of entries are included in a table.

1. DIM entry numbers for every program or data specified in the DIM table.
2. Available sector count to indicate the number of available sectors between programs or data within cylinders.
3. Cylinder entry numbers to identify the beginning of each cylinder.

Available sector numbers always begin with 9 (9xxx); the three rightmost digits denote the number of consecutive available sectors. For example, 9021 indicates that 21 consecutive sectors are unused within a cylinder. A maximum of 200 available sectors can be represented by an entry.

Cylinder entry numbers always begin with 7 (7xx); the two rightmost digits represent the cylinder number. One hundred of these entries are contained in the table, one for each of the 100 cylinders numbers 00-99.

An example of how the three types of entries might appear in a Sequential Program table for cylinders 48-52 follows:

7048 0434 0435 7049 0436 9010 0437 7050 0437 7051 0437 7052 0437

where the programs identified by DIM entry numbers 0434 and 0435 occupy all 200 sectors of cylinder 48, and the programs identified by DIM numbers 0436 and 0437 occupy all sectors of cylinders 49-52, with the exception of 10 unused sectors between the two programs in cylinder 49. Note that programs that overlap cylinders have the associated DIM entry number repeated for each cylinder on which it is stored.

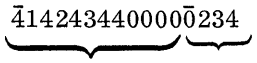
Since the SP List is so important and since the user might unintentionally destroy a portion of the list, an edit program which will restore a correct list to the Monitor pack is provided (see Sequential Program List Edit Routine).

The length of the Sequential Program table is 80 sectors unless shortened by a DEFINE control record (see DISK UTILITY PROGRAM). Fifty sectors should provide sufficient space in the table if 1000 programs are to be written on a disk pack. The user may change the size of the table for his particular needs.

Each pack in the user's system will contain an SP List that specifies the contents of that pack. A new (initial SP) List is automatically placed on a pack when it is labeled (see DUP DLABL).

Equivalence Table

The Equivalence table contains the alphabetic name and the DIM entry number of each program that has been assigned a name by the user. A name is not necessary unless the program is to be referenced using the name. In many cases only a DIM number is needed. The format of the Equivalence table is shown with an example below.



Program Name — Dim Entry Number

This example shows a program named ABCD as being equivalent to DIM entry number 0234. A 5-digit disk pack identification label that can be used by other systems must be written on the 32nd through 36th position on the last sector of cylinder 99. This sector should be given the sector address 00199 regardless of the addressing.

Mainline Core Load Map

The Mainline Core Load Map is used by the System to determine how the user wants each "core load" to be handled. A core load is defined as a mainline program together with the subprograms, subroutines, and data that are to be made available to the mainline program.

The Mainline Core Load Map is stored with the core load on disk, and when a particular core load is brought into core storage for execution, the Core Load Map that pertains to that core load is also loaded.

The Mainline Core Load Map consists of 33 core locations apportioned in the following manner:

1. The 3-digit DIM of the core load related to this record.
2. The 3-digit DIM of the next core load (the one the user wants to have executed at the completion of the current core load).
3. The 3-digit DIM of the exception core load (refer to System Alert Control Program).
4. The 3-digit DIM of the restart procedure core load (refer to System Alert Control Program).
5. A 1-digit Alert Procedure indicator. This indicator is interrogated by the System Alert Control program to determine how the user wants a particular error condition to be handled. The user has three choices:
   - 0 Halt the program in an interruptible mode
   - 1 Record the error, but do not halt
   - I Branch to the exception procedure program
6. The 6-digit disk address of the next mainline program.
7. The 3-digit sector count of the next mainline program.
8. The 5-digit core storage address which is the core loading address of the next sequential mainline program.
9. The 5-digit core storage address which is the entry address of the current core load
10. A record mark.

Interrupt Indicator Table

This table determines the order in which the user-assigned interrupt indicators are to be interrogated. It consists of a 2-digit field for the SIOC, CE, Operator Entry, Process interrupts, and the Timed interrupts.

The SIOC interrupt indicator number (45) is always inserted as the first entry in the table. The second interrupt indicator number is reserved for the CE Interrupt Indicator (27). The user can assign the priority of the interrogation sequence for the Operator Entry Interrupt indicator, the Timed Interrupt indicators and the Process Interrupt indicators. This information must be supplied when the Master Interrupt Control program is assembled. The order of interrupt indicator interrogation is determined for all mainline core loads at that time.

Because of its importance, the Any Check indicator (19) will always be interrogated first, followed by: Seek Complete (42), Multiplex Complete (40), Analog Output Setup (41), and those in the Interrupt Indicator table.

Interrupt Subprogram Identification Map

The Interrupt Subprogram Identification map (ISIM) contains address information relating to the following Executive and user's interrupt programs.

1. SIOC Execute Interrupt program (Indicator 45)
2. CE Interrupt program (Indicator 27)
3. Operator Entry Interrupt program (Indicator 18)
4. Process Interrupt programs (Indicator 48-59)
5. Timed interrupts (Indicator 43, 44, and 47)

A total of 14 Process and timed interrupts (in any sequence) are allowable. Each ISIM record is 16 positions in length and is composed of the following fields:

1. One-digit status control code
2. Six-digit disk address of the program
3. Three-digit sector count of the program
4. Five-digit core storage address where the program will be located for execution (this is also the entry address)
5. Record mark

A group mark is used in place of the ending record mark for interrupt subprograms that call FORTRAN library subroutines.

The ISIM is assembled and updated automatically by the Disk Utility Program. It is always in core storage when a process control program is being executed.

When an interrupt occurs and the interrupt program associated with it is not to be executed immediately, the interrupt is recorded by the Master Interrupt Control program which places a flag over the record mark or group mark in the corresponding ISIM record. See Program Schedule Control Program for a description of how interrupts are serviced.

The ISIM allows the Executive Control programs to call interrupt programs without knowing specific addresses.

The status control code of each ISIM entry is placed there when a mainline core load is loaded for execution. The data for the status control codes is obtained from the Status Table for Interrupts that accompanies the core load.

Status Table for Interrupts

The Status Table for Interrupts is used to identify which interrupt programs are to be recorded for execution at a later time. This table consists of a series of 1-digit entries. These entries correspond directly to a list of interrupt routines that have been placed in the Interrupt Subprogram Identification Map. Each entry in the Interrupt Subprogram Identification map corresponds to an entry in the Status Table for Interrupts.

When preparing the Status Table for Interrupts, the FORTRAN Executive System inserts a 1, 0, or $\bar{0}$ into each 1-digit location. These digits cause the following actions when interrogated by the Executive System.

Digit	Action
1	The corresponding interrupt subprogram is loaded into core storage with the mainline program and will be executed if it is called. (LOCAL subprograms that service interrupts are considered "in-core" though they actually reside on disk until execution.)
0	The corresponding interrupt core load is left on disk storage and will be brought into core storage only if it is to be executed.
$\bar{0}$	The corresponding interrupt core load is left on disk storage. If the interrupt associated with this program occurs, it is recorded for servicing at a later time.

When a Mainline Core Load map is brought into core storage, the data in the Status Table for Interrupts is transferred to the Interrupt Subprogram Identification Map.

Permanent Core Storage Area (Off-Line)

Core Storage Positions	Description
402-421	A 20-digit DIM entry or a 14-digit disk control field being used by the I/O routine, Disk Utility program, or other programs.
422-425	Starting address of work cylinder. Only the four leftmost positions of the sector address are given. This address will be $\bar{1}000$ (with the flag) unless changed by a DFINE control card.
426	Source of FEAP or FORTRAN source program input, 1 = typewriter. 3 = paper tape. 5 = card.
427	If this position is flagged, loading resumes after a TCD at core storage address 00000. If unflagged, loading resumes at the core storage address specified by positions 435-439 of this Communications Area.
428	Source of object program being loaded, 3 = paper tape. 5 = card. 7 = disk. A flag in this position indicates that a DEND type entry starts execution of the object program. No flag indicates that a TRA-TCD type entry starts execution of the object program.

Core Storage Positions	Description
429	Source of monitor control input, 1 = typewriter. 3 = paper tape. 5 = card. A flag is present in this position if library subroutines are to be called with FORTRAN object programs.
430-434	"High" indicator, i.e., the core storage address of the highest position to be loaded plus one.
435-439	Address where loading is resumed following an FEAP TRA statement. This address will always be one of the following: $\bar{0}0000$, $\bar{0}0075$, $\bar{0}0150$, or $\bar{0}0225$.

Permanent Core Storage Area (On-Line)

402-405	The starting address (four high-order positions) of the disk storage working cylinders.
411-415	"High" indicator, i.e., the core storage address of the highest position to be loaded plus one.

Executive Transfer Vector

The Executive Transfer Vector (ETV) is contained in the Skeleton Executive program. The exact location varies with the FORTRAN Subroutine Set used. The label CTVT will be in core location 07600 with subroutine sets 1 and 3; 09700 with set 2; 09100 with set 4.

Location	No. of Char.	Label	Using Prog	Description
CTVT	1	MASK	MIC, PSC	Flagged if Exec call made from mainline while masked.
CTVT+1	5		MIC	Return address to mainline
CTVT+6	7	EXECML	USER, MIC	Entry point to MIC from a mainline call
CTVT+13	5		MIC	Return address to interrupt
CTVT+18	7	EXECIN	USER, MIC	Entry point to MIC from a interrupt call
CTVT+25	5	EXPSCP	MIC, PSC	Entry address to PSC from a mainline call
CTVT+30	7	EXMICP	USER, AO, AI	Return address from all control programs
CTVT+37	1	PII	MIC, SAC, PSC	Process interrupt in execution

Location	No. of Char.	Label	Using Prog	Description
CTVT+38	1	INTPRO	PSC, MIC	Interrupt in progress
CTVT+39	1			Not used
CTVT+40	33		SAC, PSC	Current Mainline map
CTVT+73	3	ERSV	SAC & PSC	Identification code of restart mainline number
CTVT+76	1	AODOWN	SAC, AOC	Digit set when AO check indicator is turned on. All Analog outputs are terminated when this digit is set. To start AOC, digit must be reset or AOC program reloaded.
CTVT+77	1	AOBSY	AOC, MIC	Digit is set when AOC is in operation
CTVT+78	1	AOSWS	AOC	Control digit for AOC
CTVT+79	1	AOSWA	AOC	Control digit for AOC
CTVT+80	1	IOERR	SAC, IORT	Hard error in IORT
CTVT+81	1	FFIN	SIOC, SIOC FORMAT	FORMAT finished
CTVT+82	1	FORMAT	SIOC, SIOC FORMAT	FORMAT in process
CTVT+83	2	AOCNT	AOC	Analog output item counter
CTVT+85	5	LASTR	AOC	Address of last record in analog output table
CTVT+90	5	ERLOC	SAC, AOC, ADC	Address +12 of instruction that caused error
CTVT+95	5	SAC	SAC, AOC, MIC, IORT	Entry to core portion of SAC
CTVT+100	5	COMO	SAC, SIOC	Entry point to SIOC from SAC after an SIOC printer error
CTVT+105	2	NUMDEV	SAC	Number of SIOC units in 1710 system
CTVT+107	1	TYERR	SIOC, SAC	Control digit for SAC to know if SIOC peripheral error has occurred

Location	No. of Char.	Label	Using Prog	Description
CTVT+108	5	INTAB	SIOC	Address of devices
CTVT+113	3	LOFCAR	SIOC, USER	Length of one line of output printer message
CTVT+116	5	PRTTBL	SIOC, SAC, CB, PSC	High-order address of buffer table which designates the buffer area to be used by a particular output printer
CTVT+121	5	SDIGIT	SIOC, SAC, CE	SIOC unit busy digit table
CTVT+126	5	TABL	SIOC	Low position of first 5-digit field
CTVT+131	5	TAB9	SIOC, SAC, CE	Address of SIOC error count field
CTVT+136	2	SECT	SAC	Restart sector count
CTVT+138	5	SACPSC	SAC, PSC	Entry point to PSC after a unit response failure
CTVT+143	1	SIOCCT	SAC, MIC, PSC	Any SIOC interrupt control digit
CTVT+144	5	INTSIO	MIC, SIOC	Entry point to SIOC program when a SIOC interrupt occurs
CTVT+149	1	AIBSY	MIC, ADC	Digit is set when ADC is in operation
CTVT+150	5	EXBI	MIC, AO	Entry point to MIC from a call sequence to AO
CTVT+155	15	COMMON	MIC, AO, AI	COMMON linkage area
CTVT+170	1	INX	MIC,	Disk exchange indicator
CTVT+171	5	EXMLO	AI, AO, SIOC	Return from control programs to mainline
CTVT+176	5	INOUT	AI, AO, SIOC	Return from control programs to interrupt
CTVT+181	3	CSC	PSC	Sector count of current mainline program
CTVT+184	5	INTPSC	PSC, MIC	Address for entering PSC between core loads
CTVT+189	5		MIC	Address to enter MIC before returning to mainline

Location	No. of Char.	Label	Using Prog	Description
CTVT+194	3	SAVID	PSC	Dim number of special call mainline
CTVT+197	5	SAVRET	PSC	Return address of saved program
CTVT+202	5	DIAGNO	PSC	Address of current quick look
CTVT+207	1	SLNTR	AOC	AOC Slew-Trim control
CTVT+208	2	ALCT	SIOC	Error count chart numbers
CTVT+210	2	ALPRT	SIOC	SIOC Alert message printer address
CTVT+212	15	AOREC	MIC, AOC	AO Disk control field
CTVT+227	5	CALAO	MIC, AOC	Entry to AO on a call sequence
CTVT+232	5	AIINT	MIC, ADC	Entry to AI on an ADC interrupt (40)
CTVT+237	5	CALAI	MIC, AI	Entry to AI on a call sequence
CTVT+242	5	INTBEG	MIC, PSC	Entry address of MIC for interrupt entry
CTVT+247	5	GETDIA	PSC, USER	Entry address to call diagnostic program
CTVT+252	4	TCO	PSC	Previous reading of time clock
CTVT+256	4	NXL	PSC	Time of next logging operation
CTVT+260	5	RET	PSC, DIAG	Diagnostic Return address to PSC
CTVT+265	5	MICKEY	AIC, SIOC	Entry to MIC from SIOC for M.E. and S.S. Interrupt
CTVT+270	5	ADDR	MIC, SIOC	Address of Exec. call variable 0 and return address to user's program
CTVT+275	5	SKSAB	SAC1, SAC2	RETURN address of SAC2 to SAC1
CTVT+280	5	SACWK	SAC1, SAC2	Address of work constants for SAC1, SAC2

Location	No. of Char.	Label	Using Prog	Description
CTVT+285	5	FILTAS	SAC1, SAC2	SAC1 Exit control instruction
CTVT+290	5	SRET	MIC, SIOC	Return from SIOC to MIC on a call statement
CTVT+295	5	EXSIOC	MIC, SIOC	Entry to SIOC from a call statement

Disk Sector Area (Sector 19663)

Disk Sector Positions	Description
00-19	DIM entry used by I/O routine and Supervisor program.
20-21	Not used. Available for use by the user.
22	0 indicates that the program to be loaded into disk storage is in core image format; 1 indicates that the program to be loaded into disk storage is in relocatable format.
23	0 indicates card output; 1 indicates paper tape output.
24-35	Six-character alphabetic name of user's source program to be loaded into disk storage after assembly.
36-39	Four-digit DIM entry number of user's source program to be loaded after assembly.
40-44	Basic core address for process core loads. The FORTRAN will load the Mainline programs starting at the even hundred address formed by placing zeros in the two low-order positions of this field.
45-46*	Two digits (ff) indicating length of mantissa for FORTRAN subprograms. (Standard mantissa length is 08.)
47-48*	Two digits (kk) indicate FORTRAN fixed-point word length (04 standard length).
49	Digit indicates number of disk storage drives available to the Monitor System.
50-72	Supervisor Program indicators.
73	Source of input, other than disk, for FORTRAN subprograms (from DFINE control record; 5 is standard, 3 = paper tape, 5 = card).
74-75	Not used.
76	Object machine core size (from DFINE control record; 1 when system is delivered).

Disk Sector Positions	Description
77-81	Not used.
82	FORTRAN A and I/O subroutine set numbers (from FORX or XEQS control record).
83*	FORTRAN A and I/O standard subroutine set number (from DFINE control record; 1 when system is delivered).
84-88	First core storage address of a relocatable object program.
89-93	Computed relocation address of a relocatable object program.
94-98	Card sequence number.
99	A record mark (≠).

* These items are the systems standards. See Define Parameters under Disk Utility Program.

IBM 1440, 1401, 1410 Systems Header Label Area

To facilitate the processing of common disk packs, a standard alphabetic identification label is created on the 1401, 1410, or 1440 Systems. This label is not used by the 1710 System. The disk storage area (first 19 sectors of the last disk track of the last cylinder) reserved for this label can be released for other storage purposes, using the DELET Disk Utility routine, if a disk pack will be used with the 1710 only. The DIM entries for the four modules that may be connected to the system are 166, 167, 168 and 169.

Mutual Disk Pack Label

A 5-digit disk pack indentification label that can be used by the other systems (1440, 1401, or 1410) must be written on the 32nd through 36th position on the last sector of cylinder 99. This sector should be given the sector address 00199 regardless of the addressing scheme used on the remainder of the disk pack. The sector can be labeled automatically using the Define Disk Pack Label routine of the Disk Utility program.

Disk Pack Label

The first sector of cylinder 99 is a label sector, that is, it contains a label to identify the disk pack. Each disk pack used by the FORTRAN Executive System must include this label. A 5-digit disk pack indentification number in the five leftmost positions of the sector constitutes the label. This number is used to provide protection for user's records as explained in the section entitled Disk Pack Identification Numbers. This file-protected label must be generated using the Define Disk Pack Label routine. The DIM numbers for these labels are: 0158, 0159, 0160, and 0161 for packs placed on modules 0, 1, 2, and 3, respectively.

1710 FORTRAN II-D

The 1710 FORTRAN II-D program is a compiler program; it reads user-written FORTRAN statements in the form of cards, paper tape, or typewriter input, and compiles an object program. It is similar to the 1620 FORTRAN II-D program described in the IBM 1620 Monitor I System Reference Manual (Form C26-5739), and with a few exceptions, the statements that are available for use with the 1620 version apply also to the 1710 version. Only the differences between the 1620 and 1710 versions and the operating procedures are described in this section. A complete list of all 1710 FORTRAN II-D statements appears in Appendix A.

The FORTRAN compiler program can be called for operation using a Monitor FOR or FORX control record, however, if it is used on a 1710 Control System, the computer must be operated off-line.

Using the FORTRAN language, the user may write process mainline programs, process interrupt programs, process interrupt subprograms, nonprocess mainline programs, and nonprocess subprograms. These program types, their control cards, and statement restrictions are defined in the following paragraphs. A complete list of all 1710 FORTRAN II-D statements appears in Appendix A.

Nonprocess Programs

Programs to perform nonprocess functions can be compiled when an OFFLN control card is included with the source statements.

Nonprocess programs can be executed on a 1710 Control System only in the noninterruptible mode. They cannot contain a statement to call one of the Executive programs (CALL ADC, CALL SIOC, etc.). The CALL LINK, CALL EXIT, and STOP statements are valid, and Input/Output statements make use of the Supervisor I/O routines.

Process Programs

User-written programs designed to operate the 1710 Control System are considered process (or process-control) programs. There are two types of process programs: mainline programs and interrupt programs.

Mainline process programs are normally executed when the computer is in the interruptible mode of operation. They normally perform and deal with lower priority process operations than interrupt programs.

Interrupt programs are called into operation by the occurrence of an interrupt, and are usually executed in the noninterruptible mode of operation. Interrupt programs can be recorded for execution at a later time, in which case, they can be executed in the interruptible mode.

Interrupt programs can be further subdivided into two types: interrupt subprograms and interrupt mainlines. Basically, the difference is whether or not the SUBROUTINE subprogram statement is included in the source program. The interrupt mainline is the main program for an interrupt core load. Interrupt subprograms are included as part of a mainline core load; in this way, interrupts can be serviced sooner than with the interrupt core load method. (Interrupt programs cannot be recorded). The subprograms that are called by and accompany an interrupt mainline are not considered interrupt subprograms.

Source statements for interrupt programs must be preceded with an INTER control statement. The INTER control statement serves as notification to the compiler to generate a branch instruction to the Master Interrupt Control program for RETURN statements (used in interrupt subprograms), and for CALL RTMIEC statements (used in interrupt core loads).

STATEMENT MODIFICATIONS

Each statement listed below is available only in the 1710 FORTRAN II-D program or is modified from the description given in the IBM 1620 Monitor I Reference Manual (Form C26-5739). The following descriptions give the changes that have been made to the statements.

The CALL EXIT, CALL LINK, and STOP statements are permitted in a source program only if the source program statements are preceded with an OFFLN control statement which indicates that this program is a nonprocess-control program.

PAUSE Statement

The PAUSE statement is compiled to a WAIT (58) instruction when used in a process program allowing interrupts to be recognized immediately.

IF (SENSE SWITCH) Statement

This statement has been expanded to allow a program check of the setting of the 1710 Branch Indicators as well as the Console Program Switches.

Example:

```
IF (SENSE SWITCH 70) 18, 39
```

which means, "If PBI number 70 is on, transfer to statement 18, otherwise transfer to statement 39."

FIND Statement

The FIND statement is not effective with subroutine sets 1 and 3. It may be compiled in programs to be used with sets 1 and 3, but control is immediately transferred to the next sequential instruction when the FIND routine is entered. The access arm is positioned when the FIND statement is used with subroutine sets 2 and 4.

NOTE: Any area of the drive that contains the work cylinders can be specified in a FIND Statement.

FETCH Statement

Any area of the drive that contains the work cylinders can be specified in a FETCH statement.

RETURN Statement

For interrupt subprograms, this statement is compiled to a branch to the Master Interrupt Control program.

CALL RTMIEC Statement

This statement must be used as the last executable statement in each interrupt mainline program (interrupt core load) to return control to the Master Interrupt Control program. The CALL RTMIEC statement can be used in a subprogram called by an interrupt mainline to return control to the Master Interrupt Control Program.

CALL MASK Statement

This statement causes the compiler to generate an in-line Mask instruction in the object program.

CALL UNMASK Statement

This statement causes the compiler to generate an in-line Unmask instruction in the object program.

CALL ADC Statement

This statement is used to call the Analog-Digital Control (ADC) program. The ADC program and the CALL statement parameters are described in the Executive Control Program section.

CALL PSC Statement

This statement is used to call the Process Schedule Control (PSC) program. The PSC program and CALL statement parameters are described in the Executive Control Program section.

CALL SIOC Statement

This statement is used to call the Serial Input/Output Control (SIOC) program. The SIOC program and the CALL statement are described in the Executive Control Program section.

CALL AOC Statement

This statement is used to call the Analog Output Control (AOC) program. The AOC program and the CALL statement parameters are described in the Executive Control program section.

CALL DIAG Statement

This statement is used to call the Diagnostic Control program. The Diagnostic Control program is described in a separate section (see Diagnostic Control Program).

SUBPROGRAMS

The following descriptions cover subprograms that are supplied with the FORTRAN Executive System to provide the user with a means to perform specific 1710 operations. The subprograms are supplied in symbolic FEAP format to be assembled by the user (see Assembly Procedures).

Contact Sense Subprogram

This subprogram reads the status of 20 HSCS points, converts the reading into binary data, and stores the data into the low-order positions of five fixed-point variables.

CALL Statement

CALL CS (I_1, I_2)

I_1 is a fixed-point literal or variable containing the terminal address for a group of contact points. The terminal addresses are shown in Table 1. Only the first 20 points of each address group are converted. I_2 is a fixed-point array or the first of five consecutive fixed-point variables into which the binary status will be read.

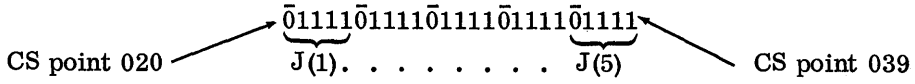
Table 1. Contact Sense Addresses

Terminal Address	Contact Sense Points Scanned
00	000-019
01	020-039
02	040-059
03	060-079
04	080-099
05	100-119
06	120-139
07	140-159
08	160-179
09	180-199
10	200-219
11	220-239
12	240-259
13	260-279
14	280-299
15	300-319
16	320-339
17	340-359
18	360-379
19	380-399

Example: (Assume fixed-point word length (k) of 5)

CALL CS(ITAS, J (1))

If ITAS is 00001, points 020 through 039 will be decoded and placed into J(1), J(2), J(3), J(4), and J(5), respectively. The array would appear as shown if all twenty contacts were closed.



Contact Operate Subprogram

This subprogram is used to operate a specified contact point.

CALL Statement

CALL CO (I_1)

I_1 may be either a literal (the terminal address), or a fixed-point variable that contains the address of the contact to be closed.

Real-Time Clock Subprogram

This subprogram reads the 1711 real-time clock.

CALL Statement

CALL CLOCK (I_1)

I_1 is a fixed-point variable in which the value of the clock will be stored. This routine can be used to read the High Resolution clock (6 digits), but will utilize only the four high-order digit positions.

Manual Entry Subprogram

This subprogram reads in the seven digits of information from the 1711 Manual Entry Switches.

CALL Statement

CALL MEOP (I_1, I_2)

I_1 is a fixed-point variable where the data of the three high-order Manual Entry switch positions is to be stored, and I_2 is a fixed-point variable where the data of the four low-order Manual Entry switches is to be stored. The low-order positions of the variables will be used for the switch data.

LIBRARY FUNCTIONS

There are seven library functions included in the 1710 FORTRAN Executive System. The functions are:

Type of Function	FORTRAN Name
Logarithm (natural)	LOGF
Exponential	EXPF
Cosine of an angle given in radians	COSF
Sine of an angle given in radians	SINF
Arctangent of an angle given in radians	ATANF
Square Root	SQRTF
Absolute Value	ABSF

All of these functions are provided in relocatable form. The relocatable subroutines can be placed with the mainline program when the core load is formed (see *LIBCR control card). Any function not loaded with the mainline program is "flipped" in the Arithmetic and Input/Output subroutine area. They are read in only when called for by the object program. (The flipped library subroutines are permanently placed in disk storage within the first six cylinders for faster access.) All user-written library subroutines are placed with the core load when it is formed.

FORTRAN CONTROL RECORDS

The FORTRAN Compiler utilizes six control records to specify output options, etc. When they are used, these records may be in any order but they must be read in between the FOR (or FORX, when compiling off-line programs) Control record and the source program statements.

The FORTRAN Control records must have an asterisk in column 1 and the Name must be punched beginning in column 2. If a control record is read and is not a legally named record, the message shown below is typed, and the program halts.

ERROR, INVALID CONTROL RECORD

The operator must correct the invalid record in the input unit and depress START.

The prescribed format and specific function of each control record is described below.

OFFLN. The OFFLN control record must precede the source statements for a non-process control program or subprogram. In effect, the OFFLN record serves as notification to the compiler to bypass error checks for STOP, CALL LINK, and CALL EXIT statements and to compile the PAUSE statement as a Halt. The format of the OFFLN control records is as follows:

Columns	1-6	*OFFLN
	7-80	not used

FANDK. The FORTRAN Compiler, as delivered to the user, will process an object program with a floating-point word length of 10 digits (f of 08 + 2=10) and a fixed-point word length of 4 digits. The operator may vary these lengths, at compilation time, by using the FANDK control record. The format of the FANDK Control record is as follows:

Columns	1-6	*FANDK
	7-8	ff
	9-10	kk
	11-80	not used

where ff is the floating-point mantissa length, and kk is the fixed-point word length. If entry is from the console typewriter, the same format must be followed.

The range of f is 2 through 8; the range of k, 4 through 10. If f or k is out of the prescribed range, the following message is typed:

ERROR, F OR K OUTSIDE RANGE

PSTSN. This control record causes the symbol table and addresses of numbered statements to be punched.

The format is as follows:

Columns	1-6	*PSTSN
	7	n
	8-80	not used

where n is 2 if paper tape output is desired, or 4 for card output. See NOTE following POBJP.

POBJP. The POBJP Control record causes the object program to be punched following compilation. The format is as follows:

Columns	1-6	*POBJP
	7	n
	8-80	not used

where n is the same as for the PSTSN Control record.

The format of the processor output (object program) is given under LOADER ROUTINE in the SUPERVISOR section.

NOTE: If n is not 2 or 4 in the PSTSN or POBJP Control records, the following message is typed out on the console typewriter and the program will halt.

ERROR, INVALID OUTPUT UNIT CODE

The operator must correct the record that contains the error and depress the 1710 Start key.

LDISK. The LDISK Control record causes the object program to be moved to a permanent area of disk storage following compilation. The format for the LDISK control statement is:

Columns	1-6	*LDISK
	7-12	Name (optional)
	13-16	Number (optional)
	17-80	not used

where Name is the left-justified program name, and Number is a 4-digit DIM entry number not already in use. If a DIM entry is not supplied, the Disk Utility program will assign one.

After compilation, the Disk Utility Program will load the programs to disk and create a DIM entry for the program. At that time, the Name supplied in the LDISK record will be placed in the Monitor Equivalence table. It is not necessary to supply the name of a FUNCTION or SUBROUTINE subprogram. The name used in the FUNCTION or SUBROUTINE statement will be used.

INTER. The INTER control record must precede each interrupt source program or interrupt subprogram. This record, in effect, causes the compiler to generate a branch to the Master Interrupt Control program when a RETURN statement is processed in a subprogram. For programs and subprograms, the INTER record causes the compiler to ensure that the entry address is the same as the first core address. (The first instruction compiled for an interrupt program or subprogram is always a branch instruction.)

Columns	1-6	*INTER
	7-80	not used

ENTERING THE SOURCE PROGRAM

The source program can be entered in the form of a punched paper tape, a deck of punched cards, or a list of statements to be typed in at the console typewriter. This entry option is specified in the FOR or FORX Monitor Control record.

Operating Procedures

All of the following operations may be performed before the processor is called, except possibly items 1 and 3. If the operation taking place just prior to the compilation of a source program required the Console Program switches to be set differently than the

desired settings for compiling, a Monitor PAUS Control record should have been inserted before the FOR or FORX record. This will allow time for the operator to change the switches.

The operations required to process a source program are as follows:

1. Set the Console Program switches for the desired compilation operations (see Table 2).
2. Set all check switches to PROGRAM.
3. If punching is to take place, ready the paper tape punch with feed code leader or, ready the card punch by loading blank cards and depressing the Punch Start key.
4. Place a JOB Control record in the input unit.
5. Place a FOR or FORX Control record in the input unit (see the Monitor Control Records section for format).
6. Place any desired FORTRAN Control records in the input unit (Load Control Records).
7. Place the source program statements in the input unit (specified in the FOR or FORX record).

Table 2. FORTRAN Compilation Switch Settings

SWITCH	ON	OFF
1	Source statements are typed on the console typewriter as they are processed. Source statement errors are typed in the form ERROR n.* At the end of Phase 1, symbol table and statement numbers are typed out.	Source statements are not listed. Source statement errors are typed in the form SSSS and CCCC ERROR n.* Symbol table and statement numbers are not typed out.
2	Trace instructions for arithmetic statements are compiled but no additional instructions are generated.	Trace instructions for arithmetic statements are not compiled.
3	A trace instruction is compiled to trace the value of the expression generated in an IF statement. An additional instruction is generated in the object program for every IF statement.	Trace instructions for IF statements are not compiled.
4	Errors made while typing source statements can be corrected by a. turning on switch 4, b. pressing the Release and Start keys,	c. turning off switch 4, d. retyping statement.

*See description under Phase 1 Errors

To resume machine operation, if the machine was stopped to allow the operator to perform any of the above operations, depress the Start key.

Typewriter Input. If source statements are entered by way of the console typewriter, each statement must be terminated with a record mark. After a statement is typed, the operator must depress the R-S key to process that statement. As soon as a statement is processed, the carriage returns to await entry of the next statement. A statement of up to 330 characters may be typed with no intervening punctuation, spacing, etc.

Normally card format need not be followed, however, in a comment statement the C must be followed by at least two blanks (spaces) before the comment is typed.

Phase I Errors

During Phase I of compilation, a number of tests are made for source program errors. If an error is found in a source statement and Program Switch 1 is on, a message in the form

ERROR n

is typed, where n is the error code (see Table 3). If switch 1 is off, the error message is in the form

SSSS + CCCC ERROR n

where ssss is the last statement number encountered by the program prior to the error, and cccc is the number of statements following the last numbered statement, ssss + cccc is the statement that contains the error. For example, the message

0509 + 0012 ERROR 1

means that the twelfth statement following the statement numbered 509 is incorrect. If an error occurs before a statement number is encountered, ssss will be 0000. Errors detected after processing the END statement reference the END statement. Comment cards, blank cards, and continuation cards are not included in the statement count.

If any Type I errors (see Table 3) are found during Phase I, no attempt is made to process the source program through Phase II. At the completion of Phase I, control is returned to the Monitor Supervisor program.

If a Type II error is found (other than Error 60), compilation continues on through Phase II. However, any FORTRAN Control records specifying output that were included with the source program will be disregarded and control will transfer to the Supervisor program at the completion of Phase II. If Error 60 is encountered, normal processing is continued since N_1 and N_2 can be corrected when loading the object program (see Subroutine Error Checks).

Phase II Errors

During processing of the intermediate output, certain checks are performed which were impossible to perform during Phase I. If an error is detected, an error message in one of the following forms is typed:

XXXX SYMBOL TABLE FULL
XXXX IMPROPER DO NESTING
XXXX DO TABLE FULL
XXXX MIXED MODE

where xxxx is the relative number of the statements within the program, not counting storage allocation statements, comments, or blank cards. The number does not correlate with an actual statement number.

Table 3. FORTRAN Source Program Error Codes

TYPE 1: Compilation continues but outputting of intermediate output is stopped. Only one error of this type is detected in any one statement.

Error No.	Condition
1	Undeterminable, misspelled, or incorrectly formed statement.
2	Syntax error in a nonarithmetic statement (exception: DO statements).
3	Dimensioned variable used improperly, i.e., without subscripting, or subscripting appears on a variable not previously dimensioned.
4	Symbol table full (processing may not be continued).
5	Incorrect subscript.
6	Same statement number assigned to more than one statement.
7	Control transferred to FORMAT statement.
8	Variable name greater than 6 alphameric characters.
9	Variable name used both as a nondimensioned variable name and as a Subroutine or Function name.
10	Invalid variable within an EQUIVALENCE statement.
11	Subroutine or Function name or dummy variable used in an EQUIVALENCE statement (subprogram only).
12	k not equal to f + 2 for equivalence of fixed point to floating point variables.
13	Within an Equivalence list, placement of two variables previously in Common, or one variable previously equivalenced and another either equivalenced or placed in Common.
14	Sense Switch number missing in an IF (Sense Switch n) statement.
15	Statement number or numbers missing, not separated by commas, or nonnumerical in a transfer statement.
16	Index of a computed GO TO missing, invalid, or not preceded by a comma.
17	Fixed point number greater than k digits.
18	Invalid floating point number.
19	Incorrect subscripting within a DIMENSION statement.
20	First character of a name not alphabetic.
21	Variable within a DIMENSION statement previously used as a nondimensioned variable, or previously dimensioned or used as a Subroutine or Function name.
22	Dimensioned variable used within an arithmetic statement function.
23	More than four continuation cards.
24	Statement number in a DO statement appeared on a previous statement.
25	Syntax error in a DO statement.
26	FORMAT number missing in an input/output statement.
27	Statement number in an input/output statement appeared previously on a statement other than a FORMAT statement, or a number on a FORMAT statement appeared in other than an input/output statement.
28	Syntax error in input/output list or an invalid list element.
29	Syntax error in CALL statement, or an invalid argument, or invalid arguments in a CALL statement to an interrupt subprogram.
30	SUBROUTINE or FUNCTION statement not the first statement in a subprogram.
31	Syntax error or invalid parameter in a SUBROUTINE or FUNCTION statement.
32	Syntax error or invalid variable in a COMMON statement.
33	Variable in a Common list previously placed in Common or previously equivalenced.
34	Library function name appeared to the left of an equal sign or in a COMMON, EQUIVALENCE, DIMENSION, or input/output statement; or function name not followed by a left parenthesis.
35	Syntax error in FORMAT statement, or invalid FORMAT specifications.
36	Invalid expression to the left of an equal sign in an arithmetic expression.
37	Arithmetic statement function preceded by the first executable statement.
38	Invalid expression in an IF or CALL statement, or invalid expression to the right of an equal sign in an arithmetic statement.
39	Unbalanced parenthesis.
40	Invalid argument used in calling an Arithmetic statement function or Function subprogram.
41	Syntax error in disk I/O statement.
42	Disk I/O list omitted.
43	Disk I/O list contains both simple variables and array names.
44	COMMON exceeds core storage size. (May occur when large array is defined.)
45	STOP, CALL LINK, or CALL EXIT statement appears in a process control program.

TYPE 2: Compilation of intermediate output continues.

Error No.	Condition
51	DO loop ended with a transfer statement.
52	No statement number for next executable statement following a transfer statement.
53	Improperly ended nonarithmetic statement.
54	Unnumbered CONTINUE statement.
55	Number of Common addresses assigned in excess of storage capacity because of Equivalence. See note at end of Table.
56	Statement number or subscript greater than 9999 (only first 4 significant digits are retained).
57	RETURN statement appeared in program other than a subprogram (statement ignored).
58	RETURN statement not contained in a Subroutine or Function subprogram.
59	Statement number not defined. See note at end of Table.
60	Syntax error in DEFINE DISK statement, invalid use of, or DEFINE DISK statement missing.

NOTE: Errors 55 and 59 are not detected if Type 1 errors occur during compilation.

If an IMPROPER DO NESTING or MIXED MODE message occurs, compilation is continued, but only to check for other errors. The FORTRAN Control records, PSTSN, POBJP, and LDISK will be disregarded, the object program will not be executed and control will be returned to the Supervisor program.

Compilation stops immediately after the SYMBOL TABLE FULL or DO TABLE FULL message is typed and control is returned to the Supervisor program. The approximate allowable number of symbols differs with the core storage size of the source machine. For a 1620 with 20,000 positions, approximately 200 symbols are allowed. For a 1620 with 40,000 or 60,000 positions, the number of symbols allowed is approximately 1200 or 2200, respectively.

End of Compilation

When all of the intermediate output is processed, the following messages are printed:

nnnnn CORES USED
aaaaa NEXT COMMON
END OF COMPILATION

where nnnnn is the number of core positions the object program requires (object program and data areas except COMMON), and aaaaa is the next available core storage position of the COMMON area, (aaaaa + 1 is the last used position of COMMON).

If FORTRAN Control records specifying output are included with the source program, the outputting takes place following the END OF COMPILATION message.

Identification Data

When a program (or subprogram) is compiled, it is headed with an identification record that will be used when the program is to be loaded into core storage for execution.

Both main program and subprogram header identification records are shown and described as below:

Mainline or Link

$\bar{0}0100\bar{0}$	2	$\bar{6}7$	$\bar{9}87898$	$\bar{N}1$	$\bar{N}2$	$\bar{W}$ord Length	$\bar{R}$ec Length	Length	$\bar{f}f$	$\bar{k}k$	$\bar{F}$irst Core	$\bar{N}$ext Common	Subroutine Indicators
5	1	2	6	2	5	2	3	5	2	2	5	5	30
Digits													

Subprogram

$\bar{0}0100\bar{0}$	2	$\bar{6}7$	$\bar{9}87898$	$\bar{S}$ubprogram Name	Length	$\bar{f}f$	$\bar{k}k$	$\bar{E}$ntry Address Less Six	$\bar{N}$ext Common	Subroutine Indicators
5	1	2	6	12	5	2	2	5	5	30

$\bar{0}0100\bar{0}$	The address of the origin of the program less 100.
2	An indicator to the relocating loader that a constant to be relocated is forthcoming.
$\bar{6}7$	The number of digits in the forthcoming constant.
$\bar{9}87898$	An arbitrarily chosen constant to identify this as a header record for a FORTRAN program.

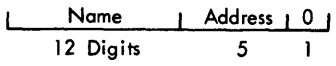
Subprogram Name	The name of the program in double digit representation (left-justified). Used only in subprograms and FORTRAN function headers.
N1	The number of words per disk record. (From the DEFINE DISK FORTRAN statement.) This field is present only for mainline programs and links.
N2	The number of logical records in the disk as used by the FORTRAN program. (From the DEFINE DISK FORTRAN statement.) This field is only present in mainline program and link header records.
Word Length	The number of digits in the words used to determine a logical record. This value is the larger of the floating word and the fixed word length.
Rec. Length	The number of sectors to be used when reading or writing logical records. This value is limited to the numbers 1 and 2.
Length	The length of the program (This must be an even number).
ff and kk	The length of the mantissa and the fixed point words in this program.
Entry Address Less Six	The first location in the subprogram, less six, to enter the subprogram.
First Core	The first location in the program to begin execution. Present only for mainline or link programs.
Next COMMON	The next location available in COMMON. This must be an even address (e.g., 19998) so that COMMON can correctly be written on disk during operation of FORTRAN loader. Subprograms do not use this value.
Sub. Indicators	A digit position for each library subroutine in the FORTRAN system.

The identification record occupies one whole sector when it is on the disk. The format for the balance of this sector, if the program is in relocatable format, is shown below:

$\bar{0}0000\bar{2}\bar{1}701234567891234567$

Subprograms Called by FORTRAN

The names of the subprograms called by a program are stored at the end of the program. The address within the calling program where the address of the subprogram will be placed is also stored along with the name of the subprogram. These 18-digit name and address records are created for the subprograms called and the last record includes a record mark as the last character. If no functions or subprograms are called, a record mark is placed in the first even core location following the program. Up to 100 subprograms may be used with any one FORTRAN main program or link (50 can be loaded with the program; 50 can be called on an as-needed basis, i.e., LOCAL).



The names and addresses of the subprograms called are moved to the FORTRAN loader work area when the subprogram is loaded. This FORTRAN loader will determine which of the subroutines called by the subprogram have not already been loaded, and will load those routines (exception: LOCAL subprograms cannot call a new subroutine). The proper addresses are placed within the calling programs to link them with the subroutines that they call.

Trace Feature

Under program switch control, instructions can be compiled into the object program to enable the operator to trace the flow of the program when it is executed. During execution of the object program, the trace output is under control of Program Switch 4 as described under Operating Procedures.

The trace output contains the value of the left-hand side of each executed arithmetic statement and/or, the value of the expression calculated in an IF statement.

SUBROUTINES

The subroutines for 1710 FORTRAN II-D are divided into two groups: (1) Library subroutines, and (2) Arithmetic and Input/Output subroutines.

Library Subroutines

Fifteen relocatable subroutines are included in the FORTRAN library (see Table 4).

Table 4. FORTRAN II-D Library Subroutines

TYPE OF FUNCTION	SYMBOLIC NAME	SUB-ROUTINE NUMBER	LIBCR REQUEST NAME	DIM ENTRY NUMBER	TRANSFER ADDRESS	STORAGE REQUIREMENTS	
						WITH FLOATING POINT FEATURE	WITHOUT FLOATING POINT FEATURE
Logarithm (natural)	LOGF	1	LOGF	10	02936	790	850
Exponential	EXPF	2	EXPF	11	02941	1066	1158
Subscripting (1 dimension)	ENTSC1	3	SUBSCR	12	02946	552	552
Subscripting (2 dimensions)	ENTSC2	4		13	02951		
Subscripting (3 dimensions)	ENTSC3	5		14	02956		
RECORD	RECORD	6	DKIO	15	02961	1564	1564
FETCH	FETCH	7		16	02966		
Routine to load or unload disk buffer	DKLIST	8		17	02971		
Routine to write or read arrays	DKARAY	9		18	02976		
Routine to complete FETCH or RECORD	DKEND	10		19	02981		
Cosine	COSF	11	SINCOS	20	02986	880	940
Sine	SINF	12		21	02991		
Arctangent	ATANF	13	ATANF	22	02996	1234	1258
Square Root	SQRTF	14	SQRTF	23	03001	526	540
Absolute Value	ABSF	15	*	24	03006	26	26

* The Absolute Value subroutine is always loaded with the arithmetic subroutines

Two forms of the supplied library subroutines are included with the FORTRAN II System. One form is for users that have the floating-point feature installed on their machine; the other form operates without the floating-point feature. Only one form is loaded when the FORTRAN Executive System is initially loaded by the user.

The library subroutines can be stored with the core load that uses them (this is accomplished with a LIBCR control statement when the core load is formed), or, the subroutines can be loaded when required during execution of the object program. The latter method is accomplished by an indirect branch to a read-in routine. The subroutine is read into an area commonly reserved for the Arithmetic and Input/Output subroutines. This method is referred to as "flipping." The Absolute Value routine does not need to be specified in an LIBCR control statement since it is always loaded with the arithmetic subroutines.

Up to fifteen user-written subroutines may be added to the FORTRAN library. The subroutines can be written in symbolic language and assembled with the FORTRAN Executive Assembly program. Procedures for adding subroutines and information for writing subroutines is given in the FEAP section under Adding Subroutines to the FORTRAN Library.

The FORTRAN statements RECORD and FETCH are processed by relocatable subroutines numbered 6 through 10. These routines are loaded into core storage only if the disk FORTRAN statements are utilized in the object program and an LIBCR control record has specified DKIO to be loaded into core. Different routines may be used to RECORD (or FETCH) an array. The routines that will write out or read in an entire array with one disk instruction will be used when the whole array is to be handled as one entity. The maximum speed in reading and writing of data from and to the disk is attained with even values for f and k, but whole arrays may be read or written as an entity if f and k are not both even. Either of these methods is faster than specifying the array as a list through the use of subscripting.

The FIND statement is used to position the access arm in the disk storage drive in advance of a FETCH or RECORD. This is not done if either subroutine set 1 or 3 is selected. It may be necessary for the FORTRAN system to change the position of the arm after a FIND operation has been initiated. No automatic repositioning to the FIND cylinder will occur after an arm disturbance until the next FETCH or RECORD statement is executed.

The FORTRAN statements that use subscripting notation will determine which of three different subscripting subroutines are entered by the program. Subroutine numbers 3, 4, and 5 are used to identify subscripting routines that handle one, two, and three dimensions, respectively (this is actually one subroutine with three entry points). Specifying that the subscripting routine be in core will slightly speed up the handling of non-disk I/O. All single subscripting is done at compilation time unless calculation of the subscript is needed (i.e., 5**I).

Arithmetic and Input/Output Routines

The arithmetic and input/output subroutines, including constants and working areas, are basic routines needed for proper execution of the object program. They are loaded without being specifically called for by the object program. Besides performing the fundamental tasks of adding, subtracting, etc., these routines also perform some diagnostic testing on the data being manipulated.

There are four arithmetic and input/output subroutine sets provided with the FORTRAN Executive System:

Set 1. The routines in this set are divided into groups, with each group containing several different routines. When a specific routine is needed, its group is loaded to core storage (from disk storage) overlaying the previous group. In this way, less core storage is required since the groups alternately occupy the same area.

When set 1 is used, the user's FORTRAN mainline programs (off-line) or the Executive routines (on-line) can be loaded into core storage starting at position 07600. The Automatic Floating Point Operations feature is not required for the use of this set.

Set 2. The routines in this set all all loaded to core storage and thus occupy more area than set 1; however, during execution of the object program no time is lost by having to read in the required group.

With set 2, the mainline program (off-line) or the Executive routines (on-line) can be loaded into core storage starting at position 09700. The Automatic Floating Point Operations feature is not required for the use of this set.

Set 3. This set is similar to set 1 except that the Automatic Floating Point Operation feature is required for use of set 3. The routines are grouped as in set 1, and the user's mainline programs (off-line) or Executive routines (on-line) can start at core location 07600.

Set 4. This set is similar to set 2 except that the Automatic Floating Point Operation feature is required for the use of set 4. The routines are all loaded to core storage prior to program, execution and the user's mainline program (off-line) or the Executive routines (on-line) can start in core location 09100.

Only two sets (one long form and one short form) are loaded to disk storage with the system. If the user's machine features do not include the Automatic Floating Point Operation feature, sets 1 and 2 must be loaded. The user further selects the long or the short form as being the "system standard" set. When the FORTRAN Executive System is delivered, the short form (set 1 or 3) is defined as being standard. This can be changed with the Disk Utility Program DFINE Control record.

The long or short form can be selected at execution time by placing the set number in column 28 of the XEQS Control record. If no digit is placed in this column, the system standard will be used. The user must remember, however, that if the short form was chosen when the core load was formed, the program cannot use the long form during execution because the program would overlap part of the Executive routines.

The digits used to select the set correspond directly to the set numbers as described above.

1 - Short form, no floating-point feature. Read most routines in when required.
2 - Long form, no floating-point feature. All routines in core.
3 - Short form, Automatic Floating Point feature required. Read most routines in when required.
4 - Long form, Automatic Floating Point feature required. All routines in core.

Any digit higher than 4 will result in the error message:

ER L8

After this message, the loading routine will set the digit to the System standard and continue. If a 3 or 4 is used for a set number and only set 1 and 2 are on the disk, 1 or 2 will be used, respectively. Similarly, if 1 or 2 is used and only 3 and 4 are on the disk, 3 will be used in place of 1, and 4 will be used in place of 2. No message will be typed in this event. If set 4 is specified, and set 2 is loaded on disk, erroneous operation may result because set 2 requires more core area.

The arithmetic and input/output subroutines supplied with 1710 FORTRAN II-D are shown in Table 5.

Table 5. FORTRAN Arithmetic and Input/Output Subroutines

Subroutine	Symbolic Name	Operation	Entry Address
Floating Point Arithmetic			
Add	.FAD	FAC + A → FAC	03424
Subtract	.FSB	FAC - A → FAC	03412
Reverse Subtract	.FSBR	A - FAC → FAC	03436
Multiply	.FMP	FAC x A → FAC	03448
Divide	.FDV	FAC / A → FAC	03460
Reverse Divide	.FDVR	A / FAC → FAC	03472
Set FAC to zero	.ZRFAC	0 → FAC	02638
Fixed Point Arithmetic			
Add	.FXA	FAC + I → FAC	03400
Subtract	.FXS	FAC - I → FAC	03340
Reverse Subtract	.FXSR	I - FAC → FAC	03352
Multiply	.FXM	FAC x I → FAC	03364
Divide	.FXD	FAC / I → FAC	03376
Reverse Divide	.FXDR	I / FAC → FAC	03388
Common Subroutines			
Store FAC and Print	.TRACE	FAC → A or I w/trace	03222
Load FAC	.TOFAC	A → FAC or I → FAC	03198
Store FAC	.FMFAC	FAC → A or FAC → I	03210
Reverse Sign of FAC	.RSGN	- FAC → FAC	03496
Fix a Floating Point Number	.FIX	FIX (FAC) → FAC	03520
Float a Fixed Point Number	.FLOAT	FLOAT (FAC) → FAC	03532
Absolute value in FAC	.ABSF	± A or ± I → FAC	03484
Exponentiation			
Fixed Point J ** I	.FIXI	FAC ** I → FAC	04124
Floating Point A ** (±I)	.FAXI	FAC ** (± I) → FAC	04128
Floating Point A ** (±B)	.FAXB	FAC ** (± B) → FAC	04244
Input/Output			
Seek a cylinder	.FINDI	Seek Record I	02898
Read Card	.RACD	Initiate Read Card	04484
Read Tape	.RAPT	Initiate Read Paper Tape	04460
Read Typewriter	.RATY	Initiate Read Typewriter	04436
Write Card	.WACD	Initiate Write Card	04412
Write Tape	.WAPT	Initiate Write Paper Tape	04388
Write Typewriter	.WATY	Initiate Write Typewriter	04364
Format Control			
	.ITYPE	Handle I, H, F, E, A, and X specifications	04508
	.HTYPE		04676
	.FTYPE		04628
	.ETYPE		04652
	.ATYPE		04604
	.XTYPE		04224
	.SLASH	Begin a new line	06588
	.REP	Repeat a single format spec.	04076
	.REP 3	Repeat a group of specs.	06928
	.REDO	Repeat the entire format spec.	06828

NOTE: The symbols above are in the System Symbol Table. The period is a part of each symbol.

FAC - simulated accumulator

A & B - floating point variables

I & J - fixed point variables

→ - store in

NOTE: The label FAC can be used to reference the simulated accumulator

Subroutine Error Checks

A number of error checks have been built into the library subroutines. The basic philosophy in the disposition of an error is to type an error message, set the result of the operation to the most reasonable value under the circumstances, and continue the program (note error D1 exception, described below). Subroutine error codes, the nature of the error, and the value of the result in FAC (symbolic name of the accumulator in which arithmetic operations are performed) are listed in Table 6.

Table 6. FORTRAN Subroutine Error Codes

ERROR CODE	ERROR	RESULT IN FAC
D1	Disk I/O used without a DEFINE DISK statement.	
D2	Logical record specified by RECORD statement exceeds N2.	
D3	No group mark found at end of an array that was read from disk.	
E1	Overflow in FAD or FSB	$\bar{9}9.....9\bar{9}\bar{9}$
E2	Underflow in FAD or FSB	$\bar{0}0.....0\bar{9}\bar{9}$
E3	Overflow in FMP	$\bar{9}9.....9\bar{9}\bar{9}$
E4	Underflow in FMP	$\bar{0}0.....0\bar{9}\bar{9}$
E5	Overflow in FDV or FDVR	$\bar{9}9.....9\bar{9}\bar{9}$
E6	Underflow in FDV or FDVR	$\bar{0}0.....0\bar{9}\bar{9}$
E7	Zero division in FDV or FDVR	$\bar{9}9.....9\bar{9}\bar{9}$
E8	Zero division in FXD or FXDR	$\bar{9}99.....$
E9	Overflow in FIX	$\bar{9}9.....$
F1	Loss of all significance in FSIN or FCOS	$\bar{0}00.....0\bar{9}\bar{9}$
F2	Zero argument in FLN	$\bar{9}9.....9\bar{9}\bar{9}$
F3	Negative argument in FLN	$\ln/x/$

ERROR CODE	ERROR	RESULT IN FAC
F4	Overflow in FEXP	$\bar{9}9.....9\bar{9}\bar{9}$
F5	Underflow in FEXP	$\bar{0}0.....0\bar{9}\bar{9}$
F6	Negative argument in FAXB Negative argument in FSQR	$/A/^{B}$ SQR/x/
F7	Input data in incorrect form or outside the allowable range	
F8	Output data outside the allowable range	
F9	Input or output record longer than 80 or 87 characters (whichever is applicable to the I/O medium being used)	
G1	Zero to minus power in FIXI	$\bar{9}99.....$
G2	Fixed-point number to negative power in FIXI	$\bar{0}00.....$
G3	Overflow in FIXI	$\bar{9}99.....$
G4	Floating-point zero to negative power in FAXI	$\bar{9}9.....9\bar{9}\bar{9}$
G5	Overflow in FAXI	$\bar{9}9.....9\bar{9}\bar{9}$
G6	Underflow in FAXI	$\bar{0}0.....0\bar{9}\bar{9}$
G7	Zero to negative power in FAXB	$\bar{9}9.....9\bar{9}\bar{9}$

The error printout is in the form

ER XX

where xx is the error code in the the table.

If error D1 occurs, the machine halts, the typewriter carriage returns, and the operator must enter the DEFINE DISK statement parameters by means of the typewriter in the form of

NNXXXXX

where NN corresponds to N_1, and XXXXX corresponds to N_2 as described for the DEFINE DISK statement. Error D1 will be indicated until the values of N_1 and N_2 are within the correct range.

The FORTRAN loader further checks the value of N_2 (number of data records as specified in the DEFINE DISK statement) to see if the N_2 disk work area would be overlaid by operation of the FORTRAN loader. The FORTRAN loader uses the disk working area (starting from the high-order positions) for tables, COMMON save area, and LOCAL subprograms. If N_2 times Record length plus 1 is greater than the lowest disk address used by the FORTRAN loader, N_2 will be redefined as

$$\frac{X - 1}{\text{Record Length}}$$

where X is the lowest disk address used by the FORTRAN loader. The user is notified of this action by the following message:

MAX N2 ALLOWABLE X̄XXXX

where X̄XXXX is the maximum allowable value for N_2 Loading and execution of the programs continues.

If Error D2 occurs, the specified record will not be written, and the index value (I) may be incorrect.

If Error F7 occurs, the field which is incorrect is replaced by zeros, and processing continues.

The exponent portion of an E-type input data field must be right-justified in that field and may contain only one sign. Deviations from this rule are not checked. For exponents greater than 99 (absolute value), the value is reduced modulo 100.

If Error F8 occurs, the incorrect field is set to blanks in the output record, and an additional record is typed. This record contains the incorrect field in the form

E (f + 6). f for floating-point numbers, and
I (k + 1) for fixed-point numbers.

This additional record is also produced on the output unit (card punch, tape punch, or typewriter) called for by the source statement.

If Error F9 occurs, the incorrect field is ignored and processing continues. However, a remote possibility exists that part of the subroutines and the object program may have been destroyed by the abnormal record. In this case, the program may inexplicably halt at some later point in its execution.

Symbol Table Listing

If Program Switch 1 is in the ON position during the FORTRAN compilation, the storage addresses of the symbol table will be listed in the following order and form.

1. Floating point constants Fixed point constants

   X̄XXXX M̄MMMMMMMMC̄C X̄XXXX F̄FFFF

   where X̄XXXX is the low-order address of the constant.
   M̄MMMMMMMMC̄C is a floating-point constant.
   F̄FFFF is a fixed-point constant.

2. Simple variable Dimensioned variables

   X̄XXXX NAME X̄XXXX NAME Ȳ YYYY

   where X̄XXXX for simple variables is the address at object time where the value for NAME will be stored.
   X̄XXXX for dimensioned variable is the address at object time of the first element in the array, NAME.
   Ȳ YYYY is the address of the last element in the array, NAME.
   If NAME* is typed, this indicates a dummy parameter within an arithmetic statement function.

3. Called subprograms

   X̄XXXX NAME

   where X̄XXXX is the location at which the starting address of the subprogram will be stored.

4. Statement numbers

$\bar{S}SSS\ \bar{X}XXXX$

where $\bar{X}XXXX$ is the address of the first instruction generated for the statement numbered $\bar{S}SSS$.

If the statement number pertains to a FORMAT statement, the location $\bar{X}XXXX$ will be the actual address of the FORMAT specification.

Symbol Table Listing for Subprograms

When compiling a subprogram, the dummy arguments are listed after statement numbers, as follows:

$\bar{X}XXXX$ NAME

where $\bar{X}XXXX$ is the location at which the actual address of the variable in the mainline program, corresponding to the argument, NAME, will be stored upon entering the subprogram. This same form is also used for simple and dimensioned variables.

The addresses listed are not the actual addresses at object time. Since programs are relocated upon loading, the listed addresses have to be adjusted relative to the starting location of the program or subprogram.

CORE LOADS

User's process-control programs are formed into groups called "core loads." A core load consists of a mainline program, subprograms, and subroutines called in the core load, and various communications areas that are used when the core load is loaded to core storage for execution and during core load execution.

Core load programs are stored on disk in core-image format to facilitate rapid loading when the core load is called for execution. The Disk Utility Program DLOAD operation is used when forming a core load. The DLOAD routine, in conjunction with a FORTRAN loading routine performs the job of converting the programs to core-image format, making up the map entries and program linkage areas.

A core load is classed as either a mainline core load or an interrupt core load. The distinction is simple: interrupt core loads are called by the Master Interrupt Control program as a result of an interrupt (or by the Program Schedule Control program if the interrupt was recorded) while the mainline core load is called by the Program Schedule Control program as a result of a CALL PSC statement in a user's program. All portions of an interrupt core load (mainline, subprograms, LOCAL subprograms, and subroutines) are usually executed in the noninterruptible mode; a mainline core load is executed, for the most part, in the interruptible mode.

The first three sectors on disk of a mainline core load contain the Subroutine Transfer Vector, some program parameters (f, k, etc.), and the Interrupt Transfer Vector. Sector 4 contains the Mainline Core Load Map for the core load, and the Status Table for Interrupts. (These communications areas are described in the Executive Control Programs section.) The mainline program for the core load begins at the beginning of the fifth sector.

During the forming of a core load, the loader errors listed in Table 7 may appear, as well as the overlap errors described later.

● Table 7. FORTRAN Loader Error Codes

ERROR CODE	MEANING, REASON	RESULT
L1	Invalid load control record. Control word misspelled, misplaced, or no asterisk.	Typeout JOB ABANDONED: branch to MONCAL*
L2	Invalid name in LOCAL record Not formed according to FORTRAN rules Invalid indicator number in *INTCR Record	Typeout JOB ABANDONED: branch to MONCAL*
L3	Multiple name in LOCAL record Same subprogram name appears more than once for same program or link, or program or link name appears more than once	Typeout JOB ABANDONED: branch to MONCAL*
L4	LOCAL subprogram table full Greater than 50 LOCAL subprograms per link not allowed Mainline table (link names) full More than 50 links calling LOCALS not allowed	Typeout JOB ABANDONED: branch to MONCAL*
L5	Invalid header record Does not conform to standard FORTRAN header record	Branch to MONCAL*
L6	Unequal F or K Subprogram F and/or K does not compare with main program F or K	Subprogram loaded, but should be corrected before execution

*MONCAL is the symbolic name for Monitor Control Record Analyzer routine.

ERROR CODE	MEANING, REASON	RESULT
L8	Invalid arithmetic and input/output subroutine set Not defined as 1, 2, 3, or 4	Set defined as system standard is loaded, depending on group loaded
L9	In-core subprogram table full Greater than 50 subprograms not allowed	Ignore above 50th subprogram
L10	New subprogram called from LOCAL subprogram LOCAL subprogram cannot call new subprogram	LOCAL subprogram loaded; new subprogram not loaded
L11	An interrupt subprogram to be loaded in core has been specified as a recorded interrupt	Typeout JOB ABANDONED and Branch to MONCAL*
L12	A core image copy of a LOCAL subprogram has been found to contain a DUP assigned asterisk	Subprogram is not loaded
L13	All available core image LOCAL suffixes have been used	Subprogram is not loaded
L14	Invalid library subroutine name	Type out JOB ABANDONED and branch to MONCAL*
L15	An interrupt subprogram has been called by a mainline program or one of its subprograms	Subprogram not loaded
L16	A SIOC Execute interrupt (45) has been detected as an entry in the INTPR Control Record. This interrupt cannot be recorded.	Subprogram not loaded

Load Control Records

When forming core loads, the following control records may be required to specify certain necessary information. In stacked input, the load control records follow directly behind the DUP DLOAD record. The load control records themselves may be in any sequence except for the FORLD record, which must be first, and the CCEND record which must be last.

FORLD Control Record

The FORLD control record is used to indicate the FORTRAN mainline program to be used in the formation of a core load. The format is as follows:

Columns	1-6	*FORLD
	7-12	Mainline program name
	13-16	DIM number

If the mainline program is in cards or paper tape (System Output Format) no name or DIM number is required. If the mainline program is on disk, either name or DIM number must be specified. If a FORTRAN subroutine set other than the system standard is desired, the set number must be punched in column 17. The standard set is with the Skeleton Executive for all core loads, and the user is responsible for the use of another subroutine set.

INTCR Control Record

The INTCR control record is used to assign interrupt subprograms (in-core or LOCAL) to a mainline core load.

The format of the INTCR control record is as follows:

Columns	1-6	*INTCR
	7-80	Name (of interrupt subprogram), slash(/), indicator number, comma, name, slash, indicator number, comma, name, etc.

Blanks are not permitted between characters. The last indicator number of a card is followed by blanks through column 80. For example:

*INTCRNAMEA/55,NAMEB/57

When an interrupt subprogram is to be loaded-on-call (see LOCAL Control Record), the subprogram must be specified in both the LOCAL and INTCR control records. When an interrupt subprogram is specified only in the INTCR record, it is treated as an interrupt in-core subprogram and will be loaded to core with the mainline program.

Interrupt-in-core subprograms cannot call subprograms or library subroutines that are loaded with (and used by) the mainline program unless the subprogram or library subroutine that is called is executed in the noninterruptible mode by the mainline. (This restriction can be circumvented by using two different names and loading the subprogram twice.) If a core load to service an interrupt is not loaded, that interrupt will be serviced by an interrupt core load which will print the message E20. This routine is provided with the system. If any interrupts are recorded, all interrupts must have associated ISIM entries. If an interrupt is to be recorded, the interrupt routine cannot be placed in core with the mainline. Any call to the ADC, AO, or SIOC programs from an interrupt that is to be recorded must be executed while the calling program is masked.

INTPR Control Record

Interrupts that are to be recorded when they are recognized and executed later must be specified in the INTPR control record. The format of this record is as follows:

Columns	1-6	*INTPR
	7-80	XX,XX,XX, etc.

XX is the indicator number of each interrupt that is to be recorded. Blanks between characters are not permitted. For example,

*INTPR51,54,52

would cause interrupts numbered 51, 52, and 54 to be recorded if they occurred during the mainline core load that is being handled. Interrupt-in-core subprograms and SIOC Enter interrupts cannot be recorded. This control record is used only when loading mainline core loads to disk.

DATA Control Record

The purpose of the DATA control record is to indicate to the FORTRAN loader that all segments of the program have been loaded prior to beginning execution.

The rules for inclusion of the DATA control record are:

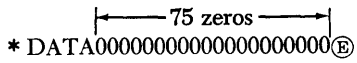
1. If the mainline or link program, or any of its associated subprograms are loaded from the paper tape reader or card reader, a DATA control record must be included in the stacked input whether or not any data is to be read by the program.
2. If the mainline or link program and its associated subprograms are all loaded from disk, a DATA control record must not be included in the stacked input.

The format for the DATA Control record is as follows:

Card

Columns	1-15	*DATA
	6-80	must be blank

Paper Tape



When a DATA record is recognized by the loading routine, a check is made to determine which subprograms have not yet been loaded. If there are any such subprograms they are listed on the console typewriter, the machine halts and the operator must then see that these subprograms are made available for the loading routine to load before depressing the Start key. If all subprograms have been loaded, any remaining data in the input unit will be skipped until the DATA record is read.

LOCAL Control Record

Subprograms that are to be loaded only when called for execution are specified at the time the core load is being formed using a LOCAL control record. Nonprocess load-on-call subprograms are specified when the mainline program is called for execution. The format of the LOCAL control record is as follows:

Columns	1-6	*LOCAL
	7-80	Mainline program name, comma, subprogram name, comma, subprogram name, comma, etc.

LOCAL subprogram names must be five characters or less in length. If a LOCAL subprogram is used to service any interrupt, the user must not permit that interrupt to be serviced while the mainline is executing a LOCAL subprogram. This may be done by masking the mainline during the execution of any LOCAL subprogram it uses. Interrupt core loads may use LOCAL subprograms without masking.

LIBCR Control Record

Any of the supplied library subroutines may be placed in core storage with the mainline program. If the subroutine is not loaded with the mainline, it will be loaded to core storage in the Arithmetic and Input/Output subroutine area when it is required for execution.

To specify subroutines that are to be loaded with the mainline, the name that is associated with the subroutine as shown in Table 4 is placed in an LIBCR control record. The format of this record is illustrated in the following example which specifies that the square root function and the disk input/output routine are to reside with the mainline program.

Columns	1-6	*LIBCR
	7-80	SQRTF, DKIO

The subroutine names must be separated by commas (blanks are not permitted). Only the subroutines listed in Table 4 need be specified as in-core library subroutines. All user-written subroutines are automatically loaded into core storage with the mainlines that call them. The Absolute Value subroutine is always loaded with the arithmetic subroutines and need not be specified in the LIBCR control record.

For nonprocess programs, the LIBCR control record is effective only for the mainline program. All user-written subroutines that are called by a Link program are loaded to core with the Link program, and the IBM-supplied subroutines are "flipped."

CCEND Control Record

The CCEND control record specifies the end of the control cards to be read before the FORTRAN programs are loaded. It must be included even if no LOCAL, INTCR, or INTPR control records are included. The format of the CCEND control record is as follows:

Columns	1-6	*CCEND
	7-80	Not Used

OPERATING PROCEDURES

To execute a previously compiled off-line FORTRAN program, the following items must be placed in the input unit after the Monitor Supervisor program has been read into core storage.

1. JOB Control record
2. XEQS Control record
3. LOCAL Control records (if required)
4. LIBCR Control record (if desired)
5. CCEND Control record
6. Main program (if not previously loaded to disk storage)
7. Subprograms (if required and not previously loaded to disk storage)
8. DATA Control record (Note: This must be supplied unless all programs called have been loaded to disk storage)
9. Input data (if not previously loaded to disk storage)
10. Job End Control record

When called for execution, the main program is converted from relocatable format and loaded into core storage. Following the loading of the main program, the "in-core" subprograms are loaded. If any subprograms are not available, the message

LOAD SUBNAM

is typed, where SUBNAM is the name of the subprogram that must be loaded in the input unit.

When all "in-core" subprograms are loaded, the library subroutines needed by the main program and "in-core" subprograms are loaded into core storage. Following the loading of the subroutines, if any subprograms have been defined as LOCAL subprograms an "object-time read-in routine" is loaded and following it a linkage area is reserved for each LOCAL subprogram.

The linkage data for each LOCAL subprogram includes the disk address, sector count, first core address, entry address, and a record mark or group mark. This linkage can come from either of two sources. If the name of the subprogram is found to be in the Equivalence table and if that subprogram is in core image with core addresses that are within the LOCAL area for the mainline, the core image copy is used as the LOCAL and the DIM entry is used for linkage data. The actual lookup in the Equivalence table is performed on the first five characters as given in the name, plus a sixth character which is supplied by the FORTRAN loader. This character is a 0-9 or an asterisk. If a name – excluding the last character of 0 through 9 – is found in the table, a check is made to determine if the core locations for that core image copy of the subprogram are within the area for LOCAL subprograms that is available with the mainline being handled.

If such is the case, no new copy is needed and the copy that was found is utilized. If one such copy was not loaded into core where the new mainline could utilize it, the search is continued with the next higher number as the sixth character. If any such copy is found to fit as a LOCAL with the new mainline, it is utilized by placing the DIM for that LOCAL in the linkage area. If the search yields a duplicate five character name with an asterisk in the sixth position, the search is abandoned because the subprogram is thus determined to be unique and cannot be used with any other mainline.

If the LOCAL cannot be found in a usable core image form on the disk, it must be loaded from the system input unit. Then the LOCAL subprogram is called for from card or paper tape and loaded into core storage. The address to which this subprogram is loaded will be the first core address for that LOCAL subprogram. The first LOCAL subprogram is then loaded permanently to disk storage by the Disk Utility Program.

DUP will assign an asterisk to the sixth character position of the subprogram name if the subprogram calls subprograms of its own. This establishes the core image LOCAL as unique. The LOCAL may call library subroutines that are flipped or are in core for use by the mainline. Interrupt core loads require pseudo-calls in the mainline program for library subroutines used by LOCALs. LOCALs cannot call library subroutines that are not also available for use with the mainline.

DUP will assign the lowest available number between 0 and 9 to the LOCAL if it might be used by some other mainline. In either case, the amended subprogram name is placed in the Equivalence table. If all of the ten copies have been loaded, the DUP handles the LOCAL name as an illegal duplicate name. The user must choose a different name and load the subprogram again.

LOCAL subprograms loaded by way of the System input unit must be stacked following any "in-core" subprograms to be loaded.

OVERLAP Errors

During the loading of the main program, subprograms, subroutines, or the read-in routine or the program linkage areas, the available core storage area may be exceeded.

If a main program or link program would exceed the available area, the following message is typed and control is transferred to the Supervisor program

NAME $\overline{X}$XXXX OVERLAP
JOB ABANDONED

NAME is the name of the program or link program, $\overline{X}$XXXX is the number of core storage positions required by that program. If the program has no assigned name, MAIN is printed for NAME.

If a subprogram would exceed the available area, the NAME $\overline{X}$XXXX OVERLAP message is typed and the named program is not loaded. Subprograms following the "overlap subprogram" are loaded if possible.

If a subroutine would overlap the available core storage area, the message

NN $\overline{X}$XXXX OVERLAP

is typed, where NN is the library subroutine number and $\overline{X}$XXXX is the length of the subroutine. The subroutine is not loaded.

If the LOCAL subprogram read-in routine or program linkage areas exceed the available core storage areas, the message

FLIPER $\overline{X}$XXXX OVERLAP

is typed. FLIPER is the name assigned to the read-in routine and $\overline{X}$XXXX is the length of the routine and linkage area required. The read-in routine and the linkage area are not loaded.

After all possible programs are loaded, and there is any error – overlap or others – the message

EXECUTION INHIBITED

is typed and a branch to MONCAL is executed. (MONCAL is the symbolic name for the entry point to the Monitor Control Record Analyzer routine.)

During loading of a FORTRAN program, the errors listed in Table 7 may appear.

Console Program Switch Settings

Switch 1. When Switch 1 is on, a list of the programs being loaded is typed on the console typewriter. The format of the list is:

XXXXXX NNNNN LLLLL LOADED

where XXXXXX is the name of the program or subprogram or the number of the subroutine, NNNNN is the beginning core storage address, and LLLLL is the length of the program.

Switch 4. When Switch 4 is on, and trace instructions have been compiled into the object program, the trace output is listed on the console typewriter. The trace output contains the value of the left-hand side of each executed arithmetic statement and/or, the value of the expression in an IF statement.

If the typewriter input is called for by the object program the operator must:

1. Type in the required data
2. Turn Console Program Switch 4 to the OFF position
3. Depress the Release key
4. Depress the Single Instruction key seven times
5. Turn Console Program Switch 4 to the ON position
6. Depress the Start key

If the operator makes a mistake when typing the input data, it is necessary only to depress the R-S key and retype the required data.

EXECUTIVE CONTROL PROGRAMS

The Executive Control programs available for use in setting up a control system application are listed below. A detailed description of each program is given later in this section.

1. Master Interrupt Control (MIC). Supervises recognition of interrupts and execution of their respective subroutines.
2. Program Schedule Control (PSC). Supervises program loading and maintains the status of the control and identification maps.
3. Analog-Digital Control (ADC). Performs all analog input operations.
4. Analog Output Control (AOC). Performs all analog output operations.
5. Serial Input/Output Control (SIOC). Handles all operations on the Serial Input/Output Channel.
6. System Alert Control (SAC). Handles all 1710 error conditions.

The Executive Control programs are available in either card or paper tape form. They are separated into individual decks or tapes so that the user may select only those programs which fit his needs.

The programs are in 1710 SPS source language when received by the user. Before they can be used, they must be assembled by the FORTRAN Executive Assembly program. The programs are unassembled so the user can assign program addresses, stipulate error procedures, and in general provide the parameters of the process. These items are supplied through the use of System Symbol Table statements. The assembly procedures are described later in the manual.

SKELETON EXECUTIVE

During process-control operation, the most often needed Executive control programs and data are kept in core storage at all times. These programs and data are termed the Skeleton Executive.

The Skeleton Executive consists of the following programs and data.

1. The Master Interrupt Control program.
2. The Interrupt Subroutine Identification Map.
3. The Executive Transfer Vector (ETV) which contains branch instructions and indirect addresses for communication; as well as common data and indicators.
4. Portions of the Program Schedule Control, System Alert Control, and Serial Input/Output Control programs.
5. The multiply and add tables.
6. The on-line IORT, used by both FORTRAN subroutines and Executive Control programs.
7. FORTRAN subroutines and data.

Skeleton Executive Loader

The Skeleton Executive loader is a small loading program whose primary function is to load the Skeleton Executive when the Executive System is initially started. Its operation is described under Assembly and Loading Procedures.

MASTER INTERRUPT CONTROL PROGRAM

The Master Interrupt Control (MIC) program performs two services: (1) it functions as an interrupt identification routine, and (2) it coordinates all communications between Executive Control programs and user's programs.

These services can be more specifically defined as follows. The MIC program:

1. Determine which interrupt(s) occurred.
2. Examines interrupts in a predetermined sequence, and enters the interrupt routine that services the interrupt, or:
3. Determines whether desired interrupt subroutines are in core storage and, if not, calls them in.
4. Handles the call sequences between user's programs and Executive programs.
5. Returns control to the mainline program when no interrupts remain to be serviced.

The return path from all interrupt programs to the mainline program is through MIC. The means of exit from an interrupt subprogram must always be a RETURN statement. Interrupt Core Loads must use a CALL RTMIEC to cause a branch to MIC.

Whenever an Interrupt indicator comes on while the computer is in the interruptible mode, an automatic branch to the address stored in Instruction Register 3 (IR-3) occurs. For proper operation of process-control programs, the address in IR-3 must be that of the MIC program.

INTERRUPT SERVICING

Upon assuming control, the MIC program tests the interrupt indicators to determine which interrupt occurred. Indicators are tested in a sequence that is prescribed by the user in the Interrupt Indicator table. This table is assembled with the MIC program and consists of 17 two-digit indicator numbers in the order that the interrupt indicators are to be interrogated.

Upon finding the interrupt indicator that is on, the MIC program checks the corresponding entry in the Interrupt Subprogram Identification Map (ISIM). The Status Control Digit in the ISIM indicates two things:

1. Whether or not the routine to service the interrupt is in core. If the routine is in core, MIC saves various areas of the Skeleton Executive, and then branches to the appropriate address in the Interrupt Transfer Vector (ITV). (This is an indirect branch, thus control passes to the interrupt routine.)
2. Whether the interrupt is to be recorded, or serviced immediately. If the interrupt is to be recorded, MIC places a flag over the record mark or group mark in the corresponding ISIM record, and control is returned to the mainline program. (See PSC for a description of how recorded interrupts are serviced.)

If the interrupt is to be serviced immediately, the areas of the Skeleton Executive that might be destroyed by the interrupt core load and the portion of core that will contain the interrupt core load are saved on disk, the interrupt core load is called in from disk and executed, and the saved portions of core storage are restored. The interrupt core load also includes the area required for LOCAL subprograms. The use of any variables in COMMON by the Interrupt Core Load is the responsibility of the user.

NOTE: Interrupt routines-in-core cannot be recorded.

Interrupt Interrogation

Interrupts are classed as internal or external depending upon their origin. Those that come from within the control system (computer) are internal; those that originate within the process are external.

Internal

The internal interrupts are contained in the following list with the name of the program that MIC branches to when the interrupt occurs.

Interrupt	MIC branches to
Any Check	System Alert Control
Seek Complete	Next Sequential Instruction
Any SIOC	Serial Input/Output Control
Multiplex Complete	Analog/Digital Control
CE Interrupt	CE Interrupt Subroutine
Analog Output Setup	Analog Output Control
One Minute	Program specified by user
One Hour	Program specified by user

The timed interrupts may be recorded.

External

There are twelve process interrupt indicators available for assignment by the user. When any one of them is on, MIC checks the status control digit in the ISIM and either records the interrupt or transfers control to the interrupt subprogram designated to service that particular interrupt.

PROGRAM SCHEDULE CONTROL PROGRAM

The Program Schedule Control (PSC) program ensures that all the user's specifications regarding core load scheduling are carried out. In addition it handles recorded interrupts, restarts programs because of error conditions, initiates logging operations, performs quick look diagnostics, and in general, keeps track of the status of core storage at all times.

Core Load Scheduling and Loading

A core load consists of a mainline program supplemented by interrupt subprograms, tables, etc. Core loads are scheduled by the user in the Core Load map when each Mainline Core Load is placed on the disk. Using the Core Load map, the PSC program sees to it that mainline programs are executed in proper sequence.

Core loads are normally executed by a call to the PSC program. This call specifies that the next scheduled core load (per Core Load map) should be loaded. The PSC program takes the disk address of the next core load in the current core load table (located in the ETV area of core storage) and reads in the first four sectors of the next core load. The fourth sector contains the new Core Load map. The record is then picked out of the Core Load map to serve as the control and address information for the new core load. PSC uses this map to determine the entry address when all other PSC operations are finished. All of the next mainline will be read into core with one read starting with the fifth sector of the mainline core load as it resides on disk.

Servicing Recorded Interrupts

According to the digits in the Status Table for Interrupts for each core load, interrupts are either serviced immediately upon detection or they are recorded for later servicing. An interrupt is said to be serviced when the program which is designed to cope with the particular interrupt is executed.

The PSC program handles the servicing of recorded interrupts at the user's request, either during the execution of a core load (serviced in the masked mode) or at the conclusion of a core load (serviced in the interruptible mode).

Servicing recorded interrupts at the end of a core load does not require a specific call sequence. The PSC program, when it is called to load a new core load, handles recorded interrupts according to the following rules:

1. If the units digit of the 3-digit identification code of the next core load is flagged, then all recorded interrupts are serviced before that core load is entered.
2. If this digit (see Item 1 above) is not flagged, the PSC program checks the Status Table for Interrupts in the next Core Load map record. Any current interrupts not scheduled to be recorded in the next core load are serviced before that core load is entered.

If a particular recorded interrupt does not meet either condition above, it continues to be recorded.

An interrupt subprogram that is in core cannot be recorded: it is executed immediately when the interrupt associated with it is recognized.

Programming Considerations for Servicing Recorded Interrupts

The servicing of recorded interrupts at the conclusion of a core load is done in the interruptible mode. Any subsequent return to the noninterruptible mode within the interrupt routine must be programmed by the user with a Mask instruction.

The following rules must be observed whenever an interrupt routine is being executed in the interruptible mode:

1. A Mask instruction must be executed before executing a CALL statement to an Executive Control program. The user must unmask after the Executive call has been completed if the processing of the recorded interrupt is to be continued in the interruptible mode.
2. A Mask instruction must be executed upon completion of the routine, prior to the CALL RTMIEC statement.

Logging Operations

The PSC program has the ability to initiate logging operations between core loads. The routine for logging (when used) must be identified by DIM 67. DIM 67 must be established before assembling the disk portion of PSC. When the system is delivered, this DIM is in use to avoid automatic assignment by DUP for some other type of user program. The user must delete the "program" supplied before loading his logging routine.

Special Operations for Log Routine

The user's logging routine must be written, compiled and loaded as if it were an Interrupt Core Load. The interrupt indicator number to be used when loading must be 99 and the DIM number that identifies the logging routine must be 0067. The exit from the logging routine is with a CALL RTMIEC statement. All of the library routines, Executive programs, subprograms, LOCAL subprograms that are available with an interrupt core load are available with the logging routine. The logging routine will run in the interruptible mode if the user unmasks during execution. The user must mask before calling an Executive Control Program and before CALL RTMIEC is executed.

CALL Statement

At any time during the execution of the mainline program, the user may call PSC to perform any of its aforementioned tasks. The CALL statement and parameter are shown below:

CALL PSC (I_1, I_2)

I_1 — PSC Option A. Specified with 0.
PSC Option B. Specified with 1.
PSC Option C. Specified with 2.
PSC Option D. Specified with 3.
PSC Option E. Specified with 4.

I_2 — A fixed-point available or literal containing XXX, where XXX is the program identification code for the next mainline program to be executed. I_2 is required only for options B and C.

Description of PSC Options

Option A — The next core load will be the one specified in the Core Load Map.
Option B — The next core load will be the one identified by XXX. If XXX is negative, the next core load will be entered in the masked mode.
Option C — The next core load will be the one identified by XXX, which is a special purpose program. (This option differs from Option B in that the user can return to the "calling program" by utilizing Option D.) If XXX is negative, the next core load will be entered in the masked mode.
Option D — Return to the mainline program that called the special purpose program.
Option E — Service all recorded interrupts and return to the next statement in the main program.

ANALOG-DIGITAL CONTROL PROGRAM

The primary function of the Analog-Digital Control (ADC) program is to read and analyze analog input points and to inform the user of overloads (signals exceeding acceptable voltage range) in the analog input area.

CALL STATEMENT

Calls to the ADC program (always in core storage) may be given from a mainline program or from an interrupt routine.

Mainline Call

CALL ADC (I_1, I_2, I_3)

Interrupt Call

CALL ADCI (I_1, I_2, I_3)

I_1 is a fixed-point variable containing XXX, where XXX is the number of ADC points to be read. I_2 is a fixed-point variable containing the TAS address of the first TAS point to be converted. I_3 is the first fixed-point variable in an array where the values of the ADC points are to be placed.

When an overload condition is detected, the ADC program will execute one of the following three options. The user must select this option when the ADC program is assembled.

1. The value of the overloaded point is replaced by 9999, and the number of the overloaded point is placed in locations 02469-02472, with a flag set in 02473.
2. The number of the overloaded point is placed in locations 02469-02472, with a flag set in 02473. The value of the overloaded point is not placed in I_3.
3. Same as option 2, except the message

   ADC OVLD $\overline{0}$XXX

   is typed on the console typewriter. The $\overline{0}$XXX represents the number of the overload point.

The user must clear the flag in location 02473 if he uses it to test for an overload condition. If multiple points are read and more than one is overloaded, only the last overloaded point number is left in core storage positions 02469-02472.

If second call to read ADC points is made before an earlier call has been completely processed, the first call will be completed before the second call will be serviced.

NOTE: Terminal address 299 is used by the ADC program to terminate a series of readings. This point must not be used by the customer. If 299 is required in the user's program, another number of an unused point must be placed in the ADC program before assembly (see ADC listing) or after loading the Skeleton Executive.

ANALOG OUTPUT CONTROL PROGRAM

The function of the Analog Output Control (AOC) program is to select and adjust the various Set-Point Positioners (SPPs) within the user's process. In doing this, the program allows the user to specify different rates of adjustment so that set-point movements can be synchronized.

Maximum accuracy can be ensured by reading feedback signals from the SPP just before adjustment. The feedback signal reflects the most current setting of the SPP, and if read just prior to adjustment ensures that the new setting will be as near as possible to the desired setting. The user must initiate the reading of feedback input points to accomplish this check.

OPERATION

The AOC program may be logically divided into two operating phases. The first is the initializing phase which sets up the conditions for selecting and adjusting the SPPs; the second is the service phase which selects the SPPs and starts the slew and trim operation.

Initializing Phase

When a call to the AOC program is executed, the initializing phase is entered and the following events occur.

1. The call parameters are stored in the applicable record in the Analog Output table, and the activity indicator in that record is set.
2. If a slew is required, a Slew/Trim Program indicator (located in the ETV area) is set to slew. This is done regardless of the previous setting of the indicator since slews are always given priority over trims. The slew/trim indicator is used by the AOC program to determine which type of operation (slew or trim) is to be performed next. If only a trim is required, the AOC program checks an "AOC busy" indicator in the ETV area; if it is not on, indicating that an AOC operation is not in progress, the slew/trim program indicator is set to trim. Since the interval between setup times is 3.6 sec, many SPPs can be set up with CALL statements before the service phase is entered.

   NOTE: The service phase is not entered until the 1710 is in the interruptible mode.

Service Phase

This phase begins when setup time is recognized by the Analog Output Setup Interrupt. The operations performed in this phase are:

1. The Slew/Trim Program indicator is interrogated to determine if a slew operation has been requested for any SPP; if it has, the addresses are selected for each SPP record that has:
   a. an activity code of 1, and
   b. a frequency code of $\bar{0}1$.
2. After all SPPs which require service have been selected, the Analog Output Setup indicator is tested and, if it is still on, the slew operation is readied.

NOTE: This Analog Output Setup indicator (28) is not the same as the Analog Output Setup Interrupt indicator (41). Indicator 41 is turned off when the interrupt is recognized, but Indicator 28 remains on until the 0.7 sec setup time has elapsed.

3. When the slew portion of the output cycle is reached, the slew is performed and all selected SPPs are adjusted. No further adjustments are made until the next 3.6 sec cycle. When no more slews are needed, the Slew/Trim Program indicator is set to trim; trims are then performed in the same manner as were slews. As the adjustment of each SPP is completed, its activity code is changed from 1 to 0.
4. When all SPPs have been trimmed, the AOC program is terminated, the AOC busy indicator is turned off, and control is returned to the calling program.

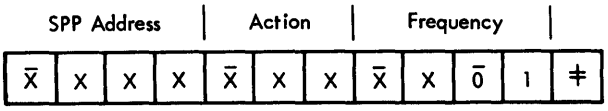
Feedback Check

At the completion of one or more analog output adjustments, the user may perform a diagnostic analysis of the SPPs that were adjusted. A suggested procedure follows:

1. Interrogate the AOC Busy indicator to determine when all desired SPPs have been adjusted.
2. Call the ADC program to read the feedback addresses of the SPPs to be analyzed. If any readings are out of the desired range, the AOC program can be called to make another adjustment to the SPP(s).

Analog Output Table

The Analog Output table is created by the program from data supplied in the CALL statement. The table will vary in size, depending upon the number of SPPs that are being adjusted at any one time. A description of the table entries follows:



SPP Address. The terminal address for upscale movement of the SPP. This address plus one is the terminal address for downscale movement.

Action. The two high-order positions of this field contain the slew count from the CALL statement. The units position is two times the Trim count (one trim equals 0.2 slew).

Frequency. Specifies how often the SPP will be serviced. This entry ranges from 01 to 99, where 01 calls for service every 3.6-sec cycle, 4 calls for service every fourth 3.6-sec cycle, etc. Thus, the higher the number the lower the frequency of adjustment. The 4-digit field is of the form $\bar{0}0\bar{X}\bar{X}$ where $\bar{X}\bar{X}$ is the frequency count. This count is placed in the two leftmost digits which are then "counted down" and permit service when they reach $\bar{0}0$. The two rightmost digits are used to restore the two leftmost digits to $\bar{X}\bar{X}$ after the SPP has been serviced.

LOCATION OPTIONS

There are two AOC programs provided with the FORTRAN Executive System. At assembly time the user must select, assemble, and load the program which will fulfill his requirements.

In-Core Option

AOC Program 1 resides on the disk and the user has the option of placing the AO table on the disk or leaving the table in core.

This option should be selected if time considerations are more significant than core area considerations. The complete table and the AO program is placed with the Skeleton Executive. The AO program requires approximately 1400 positions, and the table requires 12 positions for each SPP to be serviced.

Disk Option

AOC Program 2 resides in core and the AO table must reside in core. With this option, the AO program and, if desired, the AO table are stored on disk cylinder zero. This option should be selected if core considerations outweigh time considerations.

The program and table are called into core either when a CALL statement is executed or when the AO interrupt occurs and the in-core portion of the AO program determines that adjustment of a SPP is required. Between 110 and 450 milliseconds (ms) is required to read in the program (and table) and an additional 110 to 250 ms is required after execution to restore core storage. If the table is stored in core, the read-in time can be reduced to the range of 72-415 ms.

CALL STATEMENT

The AOC program can be called from a mainline program or from an interrupt routine.

Mainline Call

CALL AO (I_1, I_2, I_3)

Interrupt Call

CALL AOI (I_1, I_2, I_3)

where I_1 is the address of FF, FF is the frequency (see Analog Output Table)
where I_2 is the address of PPP. PPP is the SPP terminal address for the upscale movement.
where I_3 is the address of SST. SS is the number of slews to be executed, and T is twice the number of trims to be executed. If T is odd, it will be changed in the AO table to the next lower (absolute value) even amount. A negative I_3 will cause the SPP to be moved in the opposite direction from a positive I_3.

If the parameters of the CALL are to be changed they should be fixed-point variables. If no change is required, literals may be used.

SERIAL INPUT/OUTPUT CONTROL PROGRAM

The Serial Input/Output Control (SIOC) program directs all input and output operations relative to the Serial Input/Output Channel. By use of a CALL statement or an SIOC Interrupt, data can be read from a manual entry unit or a sense switch unit, or can be written on a digital display unit or an output printer. An operation function of the SIOC program is to format all printer messages through use of the FORMAT statement.

CALL STATEMENT

The SIOC program may be called from any process-control user-program. The CALL statements and parameters are shown below.

Type 1 — CALL SIOC (I_1)
or
Type 2 — CALL SIOC (I_1, I_2)
or
Type 3 — CALL SIOC (I_1, I_2, I_3, I_4)

NOTE: When SIOC is called from an interrupt program, the name SIOCI must be used in the CALL statement.

I_1 — is a fixed-point variable or literal containing EMUU, where E designates the unit type as follows:

0 — output printer
1 — digital display unit
2 — sense switch unit
3 — manual entry unit

M is a modifier used to designate the type of operation and must be one of the following:

0 — Print a message
1 — If a manual entry unit is selected, execute a write operation to turn on the Enter light; if a digital display unit is selected, display the data on the unit.
2 — If a sense switch or manual entry unit is selected, read the designated unit.

UU is the unit response indicator; a two-digit constant which indicates the specific unit to be operated. The constant is identical to the last two digits of the unit response indicator associated with the unit being addressed. For example, if 6070 is the unit response indicator for the unit, UU would be defined as 70.

Type 1 — This type is only used to turn on the Enter light. It can be written as a literal: 31 UU, where UU is the unit response indicator, or as a fixed-point variable which contains 31 UU.

Type 2 — I_2 is a fixed-point variable containing the digital display unit data or the fixed-point variable that will receive the sense switch unit reading.

Type 3 — For the output printer:
I_2 is the statement number of the FORMAT statement.
I_3 is a literal or fixed-point variable describing the number of data addresses in the list which follows. This parameter must be included with all printer calls, even if the value is zero. I_4 is the list of addresses of data to be transmitted to the output printer. This parameter is not necessary in all cases.

For the manual entry unit:
I_2, I_3, I_4 are fixed-point variables that will receive the data from the manual entry switches. Four switch settings are read into each variable.

INPUT OPERATIONS

The SIOC input routine takes control when one of the following two situations occurs:

1. A CALL statement is executed which specifies either a sense switch unit or a manual entry unit.
2. An interrupt is initiated by the depression of the Execute button on any of the input units.

CALL Statement Procedure

Input units are read in a masked mode, starting with the lowest numerical address associated with the selected unit. After each reading, the address just read is incremented by one until the twelve addresses of a manual entry unit or the four addresses of a sense switch unit have been read.

The data is read into core storage, starting with the variable specified by I_2 in the CALL statement. The input data is read into the four low-order positions of the variable. If several variables are required to contain the data, as with the manual entry switches, four digits are read into each variable.

Interrupt Procedure

The operator can initiate an interrupt by pressing the Execute button on the unit. This causes the user's program to be interrupted, thereby bringing the SIOC program into use.

All units with Execute buttons must have a common interrupt routine (corresponding to indicator 45). When an Execute button is pressed, the SIOC program places a value (described under COMMON variable) in COMMON and branches to the user's interrupt routine.

COMMON Variable

The second entry in the COMMON area must be reserved for a fixed-point variable when the SIOC program can be called into operation by depression of the Execute button on an input unit. The number placed in the variable by the SIOC program corresponds sequentially with the sequence that the sense switch and manual entry unit indicators are checked. If the indicator is associated with the sense switch or manual entry unit assigned the lowest unit number of any sense switch or manual entry unit, the number placed in the variable is one, the next higher assigned sense switch or manual entry unit is two, etc.

The number placed in the variable should be interrogated in the user's interrupt routine by a computed GO TO statement.

This variable, of course, must be reserved by every program that uses COMMON.

OUTPUT OPERATIONS

Three of the four available types of units can be involved in an output operation. They are:

1. Manual Entry Units
2. Digital Display Units
3. Output Printers

Manual Entry Unit

Although a manual entry unit is essentially an input unit, it can sometimes be considered an output unit. For example, a user's program may require some input data from a manual entry unit. To signal the operator that the data is needed, a CALL statement with a control code of 3 and a modifier of 1 is executed. This causes the SIOC program

to turn on the Enter light on the selected unit and branch back to the calling program. When the operator has entered the data, he can initiate an interrupt by depressing the Execute button.

Digital Display Unit

When a CALL statement specifies a digital display unit, the SIOC program immediately writes four digits on the unit, starting at the address associated with the "thousands" position of the unit.

The data to be written must be stored in the four low-order positions of the variable specified by I_2. The sign of the variable is transmitted to the sign position in the unit.

When a digital display unit error occurs, the SIOC program transfers control to the System Alert Control program which prints an error message on the console typewriter. The message contains the 4-digit value that was to be displayed.

Output Printer

The SIOC program section that pertains to the output printers must reside in core with the Skeleton Executive program. This allows the SIOC program to return to the user's programs between characters of the message being sent to an output printer.

Message Formatting

The SIOC format specification is placed in the body of an H-type FORTRAN FORMAT statement. For example, the FORTRAN statement

FORMAT (7HF8.2, I4)

would cause the conversion specification (F8.2, I4) to become a part of the user's object program, and thus be available to the SIOC program which performs the formatting. The maximum length of the format statement is 87 characters in the body of the H-type FORTRAN Format statement.

Errors detected during formatting are indicated by a message in the form

Snn

on the console typewriter. The nn corresponds to the message numbers as shown in Table 8.

Numerical Conversion Specifications. The SIOC program allows the option of converting stored numerical data into one of four formats before it is printed. The four types of numerical conversion are coded as E, F, I, L. The E-, F-, and I-type conversions are specified in a manner similar to that used for 1620 FORTRAN II-D FORMAT statements except that no automatic fixing or floating takes place; that is, E and F specifications must be used with floating point variables only, and I and L specifications with fixed point variables only. The format of each type is shown in Table 9.

I-Conversion (Iw or Iw.s). With the I specification an integer is printed as an integer, right-justified in the field. The field size is represented by w; w must be greater than or equal to the integer size plus 1. The additional space is for the sign. If w is not sufficiently larger, an error typeout will occur (see Table 8). The Iw.s specification is used to print fixed-point fields of a size other than the system standard of k. The s designates the field size of the variable to be printed ($S \geq 2$). The variable must be fixed-point.

Table 8. SIOC Format Error Codes

ERROR NUMBER	REASON FOR TYPEOUT	ACTION BY PROGRAM
S01	Format statement specified was not of Hollerith enclosed type	Abandon the call
S02	Number of characters not specified in an H-type specification	Go to next specification
S03	n in H specification is too large	Abandon statement
S04	n followed by an alpha character other than A, H, E, F, L, I, or X.	Search to the next comma and continue processing.
S05	An E, F, I, L, or specification is followed by something other than a number	Same as S04
S06	No period in an E, F, or L specification	Same as S04
S07	Period in E, F, L, or I specification was followed by something other than number	Set d equal to zero, then search for next comma and continue processing
S08	After apparently ending a specification other than H or X, there was no comma or /	Assume start of next specification and continue
S09	Apparently a specification starts with other than a number, a letter, (, or /	Assume next character is start of specification
S10	Attempt to nest more than one group repeat	Ignore the new repeat and use first one

ERROR NUMBER	REASON FOR TYPEOUT	ACTION BY PROGRAM
S11	All parentheses were not closed	Abandon statement
S12	Control code specified other than: P, C, B, A, T, S, R, F, M, E	Ignore character
S13	Mode shift specified in format statement	Ignore character
S14	In E specification $w < d + 6$	Set specification equal E14.8
S15	In F specification $w < d + 2$	Set specification equal E14.8
S16	In L specification $w < d + 2$	Set specification equal Lk.0
S17	In I specification w<(No. of Digits) + 1	Set w equal (No. of Digits) + 1
S18	Variable specified to be printed in E specification was not floating	Print actual variable on console typewriter. Print XX.XXXXE±XX per specification
S19	Variable specified to be printed in F specification was not floating	Print actual variable on console typewriter. Print XX.XXXX per specification.
S20	In F specified variable, exponent is greater than allotted area	Same as S19
S21	A variable was specified to be printed with no numerical specification in Format statement	Continue without printing variable

Table 9. SIOC Format Numerical Conversion Types

Conversion Specification	In Core Storage As	Printed As
nEw.d	floating point number	floating point number with an exponent
nFw.d	floating point number	floating point number without an exponent
nIw.s	integer	integer
nLw.d	integer	integer with a fixed decimal point

Legend

n The number of consecutive fields of data that will be printed according to the specification that follows n. If n is not specified, only one field is printed.

w The total number of places that the user desires to have reserved for the converted data.

d The number of places that the user desires to have reserved for data to the right of the decimal point.

s Size of the fixed field in core storage if different from k (system standard).

L-Conversion (Lw. d). With the L specification, an integer is printed with a decimal point. W is the size of the space reserved and d is the number of places printed to the right of the decimal point. W must be greater than or equal to d+2. In an L specification, the integer will be left-justified and the number of spaces designated by w will be printed. The variable must be fixed-point.

E-Conversion (Ew. d). With the E specification, floating-point numbers are printed with an exponent. W is the total space provided on the output printer and d is the number of places to the right of the decimal point. In order to provide space for the decimal point, w must be greater than or equal to d+6. The variable will be left-justified in the output space, while the exponent will be right-justified. The variable must be floating-point.

F-Conversion (Fw. d). With the F specification, floating-point numbers are printed with a decimal point and no exponent. w is the total space reserved and d is the number of places to the right of the decimal point. w must be greater than or equal to d+2. The variable will be right-justified. The variable must be floating-point.

All types of numeric specifications must be separated by commas and may be immediately preceded by a number (e.g., nIw) which designates the number of variables to be specified. In I and F specifications, spaces between fields may be provided by reserving sufficient space; but in E and L specifications, spaces must be specified as blanks (see below).

Alphameric Specifications. There are two specifications that can be used for format alphameric data. They are designated A and H.

A-Specification (Aw). The A specification causes w alphameric characters to be printed from a variable or array name. Since each alphameric character is represented in core storage by two decimal digits, the number of characters actually printed will be the largest whole number resulting from k/2 or f/2. If w is greater than f/2 or k/2, enough spaces will be provided to complete the specification; if w is less than f/2 or k/2, only w characters will be provided. Output will be right-justified. (Note: if k is odd, fixed-point integers to be printed under A specification must be left-justified in machine designation, for example, if k = 5 then AB should be defined as 41420.)

H-Specification (nH). This specification provides a method by which alphameric information may be written. The specification is immediately followed by the alphameric information to be printed. Blanks are considered alphameric data and must be included in the count n.

Space Specifications (nX). To allow spaces in the printed output, the user can specify nX. For example, 25X will cause the SIOC program to place 25 spaces in the printout. The X specification should be used to space fields in E or L type formats.

Repetition of Field Format

It may be desired to print n successive fields within one record, in the same fashion. This may be specified by giving n (where n is an unsigned fixed-point constant before E, F, I, A or L. Thus the statement

FORMAT (9HI2, 3E12.4)

is equivalent to:

FORMAT (20HI2, E12.4,E12.4, E12.4)

Repetition of Groups

A limited parenthetical expression is permitted in order to enable repetition of date fields according to certain format specifications within a longer format specification, thus,

FORMAT (17H2(F10.6, E10.2), I4)

is equivalent to:

FORMAT (26H F10.6, E10.2,F10.6, E10.2, I4)

Only one level of repetition is permitted. Thus,

FORMAT (20H3(I4, 3(F10.2, E10.4))) is invalid.

Scale Factors

The E-type specification implies a scale factor. Therefore, E16.8 for an output field will result in the printing or punching of a maximum of ten significant digits in the form (-)XX.XXXXXXXXE(-)XX. A maximum of f digits can be placed to the right of the decimal point if the d specification is greater than f. In this case, d-f low-order zeros will be inserted to satisfy the d specification. The following guide may be used when working with E-type specifications.

1. If f (floating-point precision)≤w-6, then f significant digits will be printed or punched.
2. If f >w-6, then w-6 significant digits will be printed or punched. For example, if f = 10 and the floating-point number is stored as 123456789135, it will be printed as -12.34567891E-37, according to specification E16.8.

The F-type specification also implies a scale factor. Therefore, F16.8 for an output field will result in the printing or punching of a maximum of fourteen significant digits in the form (-) XXXXXX.XXXXXXXX. However, a maximum of f digits will be placed to the right of the decimal point and the result will be right-justified in the output field. If f is larger than w-2, only w-2 digits will appear in the output.

The X specification should be used to space fields in the E-type format. In the statement

E16.8, 1X, E16.8, 1X, E16.8

a space will be provided between adjacent fields.

The SIOC formatting routine cannot handle P scale factors.

Printer Control Codes. The printer control codes are listed in Table 10. The code is enclosed in parentheses within the FORMAT statement. For example, the statement

FORMAT (39H10HSTART DATA, (T), I4, (R), 8HEND DATA)

would be printed as:

START DATA b - b XXX
END DATA
(1) (2)

(1) Spaces created by tabulate operation.
(2) Variable data.

Table 10. Executive SIOC Printer Control Codes

Format Code	Operation
(P)	Type Numerical Period
(C)	Type Numerical Comma
(B)	Print Black
(A)	Print Red (alert)
(T)	Tabulate Printer Carriage
(S)	Space Printer Carriage
(R) or /	Return Printer Carriage
(F)	Form Feed

Repetition of Control Codes. It may be desired to perform a number of successive control operations (e.g., 3(T)). This may be specified by giving n before the control code. (A) and (B) should not have repeats specified. If it is desired to repeat carriage returns, the form n(R) should be used. Slashes cannot have repeats specified.

Form Feed

The programmer is responsible for form feed control. If the printed data is to be spaced properly on form paper, the user must insert the form feed control code (F) at the proper place in the message.

Print Red (Alert)

This control code is used primarily for important messages. When the first character in a message is (A), i.e., Print Red, the SIOC program immediately prints the message from core storage (alert messages are never put on disk storage). If a previous message is in the process of being printed, it is interrupted so that the alert message can be printed.

After the SIOC program has completed printing the alert message, it continues with the interrupted message. The user should end his alert messages with a Print Black, and a Return Carriage.

When a Print Red code is used within a message, it causes the printer to start printing in red. Black printing is not resumed until a Print Black code is executed.

NOTE: A Return Carriage should be programmed by the user at the beginning of an alert message so that the message will start on a new line. The R or slash which indicates the carriage return must follow the (A).

Length of Printed Line

At assembly time the user specifies, for all output printers, a maximum number of characters per line that should appear in the printed output (see Assembly Procedures).

In SIOC FORMAT statements, the user is responsible for returning the carriage when printing. The length of line specified at assembly time is simply a reasonable number of characters to prevent the output printer from continuously printing at its right margin setting.

The responsibility to return the carriage on the output printer is the user's in order to allow printing of one line of type in a DO loop.

For example, to print a line on the console typewriter the user may use the following statements.

```
PRINT 10, (X(5), J=1, 10)
    10 FORMAT (I4)
```

In order to print the same information on the SIOC output printer the following statements would be required.

```
      CALL SIOC (IEMUU, 1, 0)
 1    FORMAT (1H/)
      DO 99 I=1, 10
      CALL SIOC (IEMUU, 10, 1, X(J))
10    FORMAT (2HI4)
99    CONTINUE
```

NOTE: An additional control code is available for an end of message specification with the FORTRAN Executive formatting program. The character is written (E). This character is optional in the FORTRAN Executive system.

In no case should change mode (M) be used in a FORMAT statement.

Ending a FORMAT Statement

During output of data, the object program scans the FORMAT statement to which the relevant output statement refers. When a specification for a numerical field is found and list items remain to be transmitted, output takes place according to the specification, and scanning of the FORMAT statement resumes. If no items remain, transmission ceases and execution of that particular output statement is terminated. Thus, a numerical output operation will be brought to an end when a specification for a numerical field or the end of the FORMAT statement is encountered, and there are no items remaining in the list.

Printing the Message

After the SIOC program formats the message, the message is transmitted to a "disk buffer area." This area, specified by the user, can be up to five cylinders in length. When the system is delivered, one half cylinder is defined for messages. Changes in the length and location of the disk message buffer can be made when the system is defined. The procedure to be followed is described in the Assembly and Loading Procedures section.

If no previous messages are waiting in the disk buffer area and the SIOC is not busy, the first character of the message is printed immediately and control is returned to the calling program. If some previous message is in the process of being printed, the new message is stored in the buffer area "behind" all previous messages. If the buffer area is full when a new message arrives, the SIOC program "interlocks" and prints messages continuously until there is room for the latest message. Regardless of the size of the disk buffer area, there is a limit of 99 of the number of messages that may be stored on disk at one time.

The user must provide a 100-digit buffer area in core storage from which characters can be printed. When the first 100 characters of a message have been written, the next 100 characters are brought into the core buffer area from the disk buffer area. Upon completion of the message, the SIOC program checks the disk buffer area and starts outputting a new message if one is found to be waiting. The user may provide a buffer for each output printer or any number of buffers to be shared by all printers. These options are specified when the SIOC program is assembled.

While the output printer is actually operating on a character, the user's mainline program or interrupt subroutine is being executed. When the printer has completed the indicated operation, a signal is generated to initiate an SIOC interrupt which permits the next character to be transmitted to the printer.

Printer Errors. If an Alert (Print Red) message is being processed when a printer error occurs, the SIOC program will attempt to print the message, from the beginning, up to three times. If the error persists after three tries, the program will force the erroneous message to print out on the selected printer and will type the following message on the console typewriter:

THREE ERRORS ON ALERT MESSAGE

If a regular (non-alert) message is being processed when an error occurs, the SIOC program will attempt to print the message up to nine times. If the error persists, the selected unit will be logically disconnected and the secondary unit specified at assembly time will be used to print all subsequent messages sent to the faulty unit.

Unit Response Errors. The PSC program checks to see if unit responses are being received from the output printers in use. If a missing unit response is detected, the PSC program transfers control to the SAC program which determines the unit that is not responding. The SAC program types the character . (period) on the unit in error, types the message

NO RESPONSE UNIT XX

on the console typewriter, and then returns control to the PSC program.

Whenever the number of output printer messages (specified at assembly time) or the actual disk area reserved for messages is exceeded, the SIOC program will continue to print messages on one unit at a time until the space on the disk is sufficient to contain the messages that have been specified for printing. In the intervening time, the printing of each entire message will be done in the masked mode.

Each time a printer fails to respond when a character is sent to it, an error message will be printed on the console typewriter and another character will then be sent to the unit. The no-response condition is detected using a timing loop that will time out after several seconds. The error message printed will be in the format 60UU49ZZZZZ, where 60UU is the device indicator, 49 is an operation code, and ZZZZZ is an address that can be ignored. This message indicates that the unit is no longer reliable.

If many messages are specified for any output printer that continues to fail to respond, these messages will fill the user's disk message area. To continue system operation the system must be started again (see Assembly and Loading Procedures) and the faulty unit taken off-line to prevent the filling of the disk message area with messages that cannot be printed, and to permit the off-line repair of the printer. An alternate printer, if available, may be substituted for the defective printer to enable the continuing of the process-control system (see CE Interrupt Routine).

SYSTEM ALERT CONTROL PROGRAM

The System Alert Control (SAC) program takes control of the 1710 whenever an error condition is detected outside of IORT. SAC determines which error is present, records each error by type, analyzes the error with respect to operating conditions, and decides which of the following error procedures to execute.

1. Restart, using the program specified for restart in the Core Load map.
2. Branch to the Exception Core Load specified by the user in the Core Load map. (The Exception Core Load will be entered in the masked mode.)
3. Record the error and continue with the current core load.
4. Wait in the interruptible mode. The Any Check Interrupt will cause the SAC program to be entered.

Any Check Interrupt

When any of the errors listed in Table 11 occurs, the Any Check Interrupt brings the SAC program into use. The SAC program analyzes the error, prints an error message, and then determines which alternative procedure to follow.

Table 11. System Alert Control Program Error Checks

Name	Indicator Code
1620:	
Read Check	06
Write Check	07
MAR Check	08
MBR-E Check	16
MBR-O Check	17
1711:	
*Any Check	19
TAS Check	21
Function Register Check	22
Analog Output Check	23

* No error count kept.

Error Messages

The SAC program error message format is shown in Table 12. Up to 8 items can appear in a message; however, only the first 5 items appear in all messages. The error codes that appear as item 3 of each message are shown in Table 13.

CE Interrupt

In the FORTRAN Executive System, the CE Interrupt has three functions:

1. To type out a count of all errors that have occurred since the last depression of the CE Interrupt switch.
2. To allow the Customer Engineer to set or reset the AODOWN digit (this digit, when set to 1 makes the Analog Output Control program inoperative).
3. To allow the Customer Engineer to logically disconnect from, or reconnect to the system, any SIOC units.

Table 12. SAC Error Message Format

Item	No. of Characters	Message	Description
1	3	ERR	Each message from the SAC program starts with these three characters.
2	3	X̄XX	Current Core Load Identification Code
3	2	X̄X	Error code (Table 12)
4	7	RESTART or ERR MSK or SKIP TO (exception) or WAIT or RETURN	Alternative procedure to be taken
5	3	X̄XX	Identification code of the alternative procedure core load. This is either the Restart Procedure identification code or the Exception Procedure identification code.
6	22	0̄20̄0̄0̄0̄0̄1 0̄0̄0̄0̄0̄0̄10̄0̄0̄	Two digit count of all error indicators. This sample message indicates two Read Check errors, one MBR-O Check error and one Address Check error.
7	5	X̄XXXX	Core location of the instruction that caused the error condition. This is a TAS instruction.
8	12	X̄XXXXXXXXXX	TAS instruction that caused the error condition.

Typeout of Error Count

When the CE Interrupt switch is depressed, the CE Interrupt routine is called into use. The first function of this routine is to type out error codes and error counts. The format of the typeout is as follows:

Error Indicator	Error Count
XX	XX
XX	XX

Only nonzero error counts are typed out. For example, if no Read Checks have occurred since the last typeout, then neither 06 nor its count will be typed out. After each typeout the counts are restored to zeros.

AODOWN Status

The AODOWN digit is set to 1 whenever an Analog Output Check (analog output relay failure) occurs. Until this digit is reset to 0, the AOC program is inoperative. To allow the Customer Engineer to set the digit to 0 or 1, the CE Interrupt subroutine types out

Table 13. SAC Error Message Codes

Code	Description
TAS Errors:	
01	TAS Check occurred while TAS was not in use.
02	Function Register Check occurred while TAS was not in use.
03	Illegal OP code was detected during the execution of a TAS instruction.
04	Illegal function code (Q_7 digit) was detected during the execution of a TAS instruction.
05	Illegal terminal address was detected during the execution of a TAS instruction.
06	An 03, 04, and/or 05 code occurred but the TAS instruction was re-executed with a legal OP code, function code, and terminal address.
Analog Output Error:	
07	Analog Output Check - the analog output relays failed to unlatch. The AOC program cannot be used until the condition that caused this error is corrected (see C.E. Interrupt).
IORT Errors:	
10	Disk Error — Alert code in MCLM specifies a Wait condition.
11	Disk Error — Alert code in MLCM specifies branch to Exception routine.
Errors other than TAS, or AO:	
20	Less than three errors have occurred in this core load.
21	Three errors have occurred in this core load.
22	Less than three errors have occurred in this core load; a process interrupt is not being executed.
23	All error indicators have been interrogated and the Any Check indicator (19) will not turn off.
24	Error count overflow.
25	Three errors have occurred in this core load; a process interrupt (recorded) is being executed.
SIOC Errors:	
30	An error, other than a parity error, has been detected on a digital display unit. When this code appears in an error message, the digital value that could not be displayed follows the code in the message, e.g., 30 XXXX.
31	A parity error has been detected on a digital display unit. When this code appears in an error message, the digital value follows the code in the message (see Code 30).
32	This code is always used in conjunction with code 30. If a digital value will not display properly, the SIOC program will try to display 9999. If the nines display properly, code 32 is indicated. In an error message, this code is followed by the indicator number of the unit in error.
33	This code is similar to code 32, except that if the nines do not display properly, code 33 is indicated. In an error message this code is followed by the indicator number of the unit in error.

the following message after the error count typeout:

CHANGE STATUS OF AODOWN DIGIT
YES, INSERT 1. NO, 0. RS

NOTE: RS = Release and Start

If a 1 is inserted, the following message will be typed out:

INSERT 1 TO DISCONNECT OR
0 TO REACTIVATE. RS

If a 0 is inserted after a previous disconnect, the AO program will be reinitialized. This means that no AO operations will occur until a new call is executed.

If a 0 is inserted in response to the change-status message, the subroutine will proceed to the SIOC disconnect routine.

SIOC Disconnect Routine

During each execution of the CE Interrupt subroutine, the Customer Engineer is given the opportunity to logically disconnect or connect any SIOC units. The following message is typed:

CHANGE SIOC UNIT STATUS. YES, INSERT 1, NO, 0. RS

If 0 is inserted, the subroutine returns control to the MIC program. If a 1 is inserted, indicating that change is desired, a second message is typed.

ENTER UNIT NUMBER 70-89 RS

An SIOC unit code must be entered, after which the following message is typed:

TO DISCONNECT UNIT ENTER 1, CONNECT, 0. RS

If a 0 is entered, the message

UNIT CONNECTED XX

is typed and control returns to the MIC program. If a 1 is entered, indicating a printer unit, the message shown below is typed out.

ENTER SECONDARY PRINTER NUMBER 70-89. NONE 00. RS

This gives the Customer Engineer the opportunity to specify a different secondary printer unit than was specified in the SAC program at assembly time. After a number is entered, the subroutine transfers control to the MIC program.

NOTE: Any printer messages that are in progress when the CE Interrupt key is pressed will be completed on the original printer unit even if that unit is logically disconnected via the CE Interrupt subroutine.

Invalid Unit Indicator Numbers. If the Customer Engineer enters an invalid unit indicator number, i.e., one that is outside the range of 70-89, the message

UNIT CODE OUT OF LIMITS

is typed along with a repeated request for a unit number.

CORRECTIVE PROCEDURES

To a great extent, the corrective procedures previously mentioned are user-controlled by certain fields in the Core Load map. Some actions taken by the SAC program, however, are mandatory due to the type of error. Figure 1 shows the logic of the corrective procedure selection.

Restart Procedure

The automatic restart performed by SAC will cause the Mainline Core Load specified for restart in the MCL map to be read in and entered. A large portion of the Skeleton Executive will also be initialized at this time. The Core Load specified for restart must not expect data that was not on disk, or in the FORTRAN common area to be available, i.e., the program must expect initial conditions when it is entered. Two exceptions to the reinitialization of data on restart are the Analog Output records and program, if either in core or on disk, and the SIOC messages that are being outputted or are on disk in the SIOC message area. These will be retained and the output operations continued.

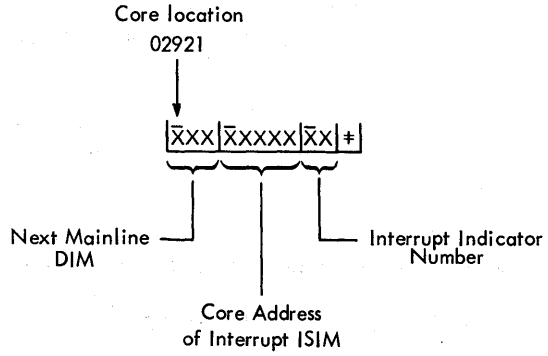
If the mainline program has initiated a call to SIOC to print a message, and the mainline program is not masked, the message will be lost if a restart is initiated before the message is formatted and stored on disk. This applies, even if the message is an alert message, up to the time the formatting is complete. To avoid the loss of any message, the call should be made while in the noninterruptible mode.

The FORTRAN areas and the MIC, ADC, PSC, and SAC programs will be reinstated when a restart is initiated. The math tables and the IORT will also be reinstated. The Executive Transfer Vector and ISIM will be reinstated, but information in ETV needed to continue SIOC output printer operation and Analog Output operation, and the indications of recorded interrupts that have occurred but have not been serviced, located in the ISIM will be saved. The error counts maintained by SAC and IORT will be saved. In general, every area of core within the Skeleton Executive will be reinitialized except those areas that are needed for SIOC message operation, Analog Output operation, and recorded interrupt service operation.

Exception Procedure

An Exception Core Load is provided with the system. This core load will handle error conditions in the following manner:

If the errors occurred while executing a recorded interrupt routine, the data in positions 02921-02930 are typed out on the console typewriter and the Next Core Load is called for execution. The format of the data is shown below:



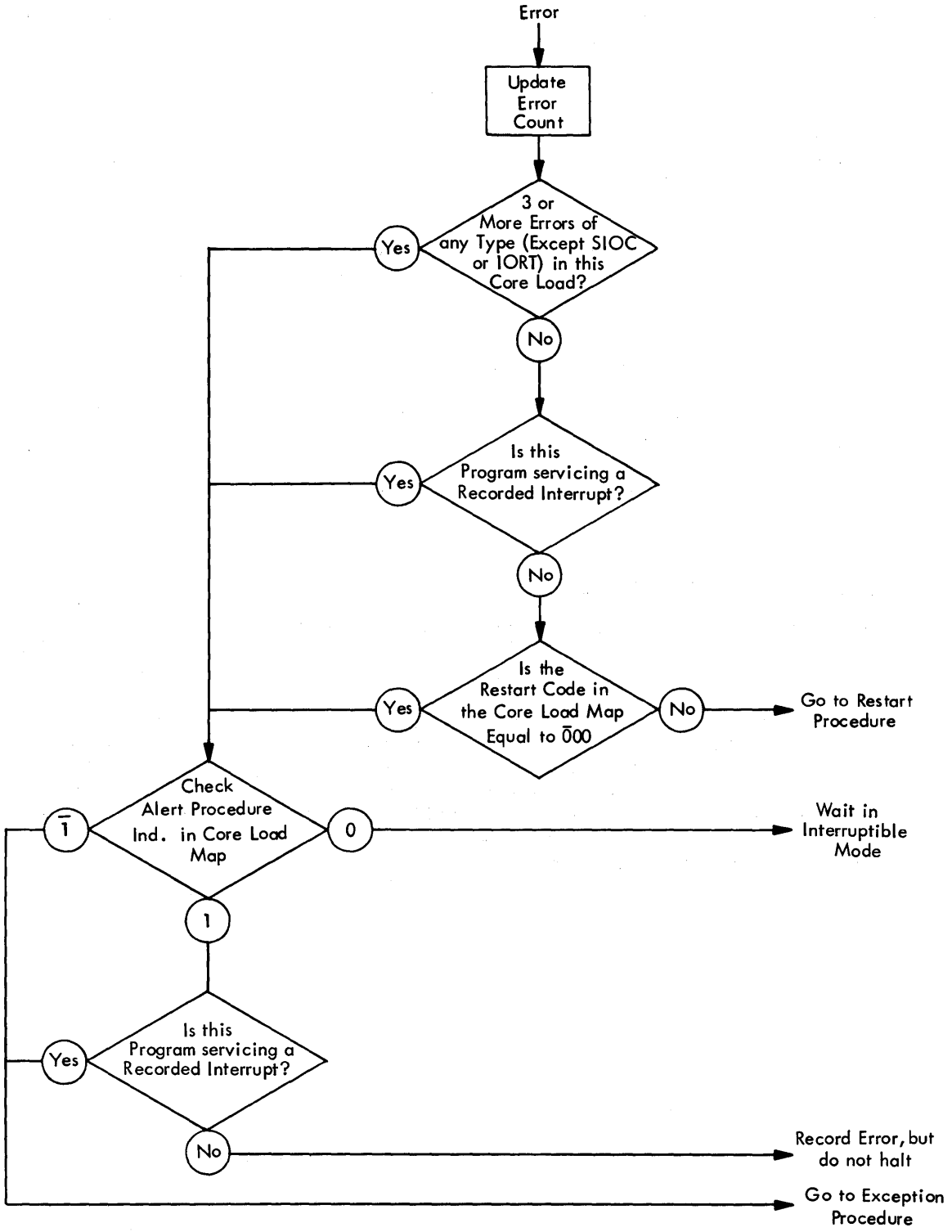


Figure 1. System Alert Control Program Corrective Logic

If the error occurred while executing a mainline core load, the Restart Core Load is executed after typing the following message:

$\overline{X}XX\overline{X}XX\overline{X}XX\overline{X}XX$

where the four fields are the DIM numbers of the Current, Next, Exception, and Restart core loads, respectively.

Error Count Retrieval

The error counts maintained by SAC and IORT will be printed when the CE interrupt is initiated. They will be set to zero after this typeout. The counts are maintained in core at locations 02138 to 02161 with a record mark in 02162. The format of these counts is shown below:

$\overline{X}X\overline{X}X\overline{X}X\overline{X}X\overline{X}X\overline{X}X\overline{X}X\overline{X}X\overline{X}X\overline{X}X\overline{X}X\overline{X}X$

The $\overline{X}X$ is the number of errors that have occurred for each error indicator. The indicators checked and the order of the count fields is shown below:

06 07 16 17 36 37 38 08 21 22 23 45

NOTE: 45 is used to identify Any SIOC error.

Checking Errors in Interrupt Routines

The SAC program cannot recognize an error condition that causes an interrupt when the 1710 is in the noninterruptible mode. The error interrupt will only be recognized after the interruptible mode is re-entered, i.e., when a Branch Out instruction is executed by MIC.

The user may permit SAC to service the error conditions that occur while his program is noninterruptible, but the errors that occur in interrupt routines may be incorrectly considered as mainline errors. Another method of handling errors while in the noninterruptible mode is to test the error indicator in the program, and take the corrective action that is needed without entering SAC. Indicators can be tested using the IF (SENSE SWITCH) statement in FORTRAN. Any two digit indicator number may be used as the sense switch to be tested.

Several diagnostic aids are included in the FORTRAN Executive System. These include a trace option, a set of five, short, specialized diagnostic routines (Quick Look Diagnostics), and a comprehensive Diagnostic Control program. The latter two aids are real-time diagnostics which attempt to detect errors before machine malfunctions occur.

TRACE OPTION

To aid the user in checking out his mainline programs and subroutines, the FORTRAN Executive System provides for a trace option within each Executive program. When used, this option traces the logical flow of the programs and types out messages when significant changes in flow occur (Table 14).

Table 14. Executive Program Trace Option Typeouts

Typeout	Description of Typeout
AIC	The Analog-Digital Control program has been "branched to" because of a call from either a mainline program or an interrupt subprogram.
AI	The Analog-Digital Control program has been "branched to" because of a Multiplex Complete interrupt.
AOC	The Analog Output Control program has been "branched to" because of a call from either a mainline program or an interrupt subprogram.
AO	The Analog Output Control program has been "branched to" because of an Analog Output Setup interrupt.
SIC	The Serial Input/Output Control program has been "branched to" because of a call from either a mainline program or an interrupt subprogram.
SI	The Serial Input/Output Control program has been "branched to" because of an Any SIOC interrupt.
CSC	The Contact Sense Control program has been "branched to" because of a call from either a mainline program or an interrupt subprogram.
PSC	The Program Schedule Control program has been "branched to" because of a call from a mainline program.
$\bar{X}$XXXX	The digits $\bar{X}$XXXX represent the RETURN address of a specific call sequence. This typeout occurs when the purpose of the specific call sequence has been fulfilled and the "calling program" (mainline program or interrupt subprogram) is again in control.

To use the trace option of the FORTRAN Executive, the user must specify the label DEBUG as 00001 when loading the System Symbol table. This statement, when the Executive program is assembled, will cause the trace instructions to be incorporated. For a more complete description of the DEBUG statement, see the section concerning Assembly Procedures.

When the program checkout is completed, the trace instructions can be removed by reassembling the Executive programs using a modification of the original control statement.

If the trace option is used with the AOC program, the trace message (AO) is typed after the AO points are set up to avoid output operations. (The delay would be caused by the trace message being typed during the 0.7 sec setup period, thereby leaving less time for output operations.) The message, AOC, is typed when a call to the Analog Output program is made. This message occurs as soon as the Analog Output program is entered.

NOTE: When the trace option is in operation, the typeouts can be suppressed by turning program Switch 4 off.

QUICK LOOK DIAGNOSTICS

The Quick Look diagnostics are a set of five routines designed to verify proper operation of selected machine circuits. Each diagnostic routine is short (approximately 200 core locations) and has a relatively fast execution time (less than 28 ms).

These routines are under direct control of the PSC program. The user does not have to load them nor be concerned about their execution.

The primary function of the routines is to determine that a block of circuitry is functioning normally. There is no attempt to provide any information of a descriptive or analytic nature. If a malfunction is detected, the more comprehensive Diagnostic Control program is called upon to further isolate the error.

Only one of the five routines is ever in core storage at any one time. It is brought into core and executed by the PSC program when a new core load is loaded. It is executed only between mainline core loads. A different routine is loaded when the next core load is brought into core storage. Thus, the five routines are rotated so that each is executed periodically.

Circuitry Tested

The Quick Look diagnostics concentrate on the circuitry in the computer that might fail and yet not cause an error check. The circuitry most susceptible to such failure is related to conditional branch and arithmetic operations. Consequently, the five diagnostic routines perform their tests in these areas. Specifically, the circuitry tested by each of the routines is apportioned as follows:

Routine 1 - Add, Add Immediate, Compare Immediate, Branch Not Equal, Branch No Overflow

Routine 2 - Add, Add Immediate, Subtract, Subtract Immediate, Compare Immediate, Branch No Overflow

Routine 3 - Transmit Digit, Branch On Digit, Branch No Record Mark, Branch No Flag, Set Flag, Clear Flag

Routine 4 - Branch On Indicator, Branch No Indicator, Branch No Overflow, Branch On Overflow

Routine 5 - Branch High, Branch Low, Branch Zero

DIAGNOSTIC CONTROL PROGRAM

The Diagnostic Control program is a comprehensive error detection program that is executed as a result of either a direct call from the user or a call from the Program Schedule Control program.

This diagnostic program performs all of the checks that the Quick Look Diagnostics perform plus analysis of the Divide special feature. The program determines whether a particular circuit is functioning properly. If it is not, a message is typed on the console typewriter.

Call Statement

The user may call the Diagnostic Control program by executing the following FORTRAN CALL statement.

```
CALL DIAG
```

The DC program may be called only from a noninterrupt routine but will probably find more use during times when the computer is "idling," i. e. , not performing any critical operations. The 1710 must be masked before a call to the Diagnostic Control program is executed.

Diagnostic Control Entry

The FORTRAN Executive System will execute the Diagnostic Control program under the following conditions:

1. One of the Quick Look routines executed by PSC between core loads has detected an error.
2. A call has been executed for the diagnostic program in a process program. (Probably in a user's Exception Procedure program.)

When the FORTRAN Executive System branches to the Diagnostic Control program, the following message is typed on the console typewriter:

```
ERR  ENTER  DCP
```

When the Diagnostic Control program is completed, the return to the user's program is denoted by the following message:

```
DCP COMPLETED XXXXXX
```

In this message, XXXXXX is the contents of a 6-digit communication area containing an indication of any errors that may have been detected. The six possible configurations of XXXXXX are shown below along with the condition they indicate:

100000 - Any error which indicates that the 1710 System is no longer reliable. The following message is typed along with this configuration:

```
CALL IBM
```

The computer re-enters and repeats the diagnostics.

010000 - The arithmetic tables contained an error but were corrected by the PSC program.

001000 - The High Positive indicator circuitry is not functioning properly.

000100 - The Overflow indicator circuitry is not functioning properly.

000010 - The multiply circuitry is not functioning properly.

000001 - The divide circuitry is not functioning properly.

If a disk error should occur while the Diagnostic program is attempting to restore core storage, the message DK OVLA ER is typed, and another attempt to restore is made.

The Supervisor Program performs off-line control and input/output functions for the FORTRAN Executive System. Through the use of control records, the FORTRAN compiler, the FEAP processor, and the Disk Utility Program can be called into operation by the Supervisor Program. It must be noted, however, that these Monitor-type operations (the operation of the Supervisor Program, FORTRAN compiler, FEAP processor, and the Disk Utility Program), must not be called for unless the 1710 is in the noninterruptible mode. (If an interrupt were to occur during a FORTRAN compilation, for example, there would not be any routine in core storage to recognize the interrupt.)

The Input/Output routine in the Supervisor Program is used by the FORTRAN nonprocess object programs and by the Disk Utility Program. Through the use of a macroinstruction in a FEAP nonprocess source program, cards and paper tape can be read and punched, data can be stored to and retrieved from disk storage, and data can be read and typed from the typewriter under control of the Supervisor I/O routine. Error checking and correction procedures are a part of the I/O routine.

CONTROL RECORDS

The control records recognized by the Supervisor program are described here in terms of cards, but the records can be in the form of paper tape records or typewriter input.

The input to the Supervisor Program consists of one or more "job decks." A job deck may be a program to be compiled or executed, a combination of these two (including data); or it may be a series of Disk Utility Program operations. The processing of each job deck is controlled by the Supervisor Program as specified by the Monitor control card that precedes it.

When a Monitor Control card is read, the program required to do the job is read into core storage from disk storage. The program then processes input until the end of the job is reached, a new Monitor Control card is encountered, or an error occurs. When the end of the job deck is reached or a new Monitor Control card is encountered, the Supervisor Program is reloaded into core storage from disk storage, and the process is repeated. If an error occurs, a message is printed to identify the error, and the remainder of the job will be processed. If it is not possible for the job to continue, the Supervisor Program will skip to the next job. All Monitor Control records, with the exception of those entered from the typewriter, are typed out on the console typewriter.

Eleven Monitor Control cards are used to indicate the processing required of the Supervisor Program. The manner in which the Supervisor handles each of these cards is shown in Figure 2.

Operation Codes

An alphabetic pseudo operation code, left-justified in columns 3-6, is used to identify each of the eleven control cards. By examining the operation code, the Supervisor Program determines what processing action is required. Columns 1 and 2 must contain record marks.

JOB

A JOB operation causes (1) the description or operating instructions contained in the JOB Control card to be typed, (2) modifies the disk module, if required, and (3) checks to ensure that the proper disk packs have been attached by the operator.

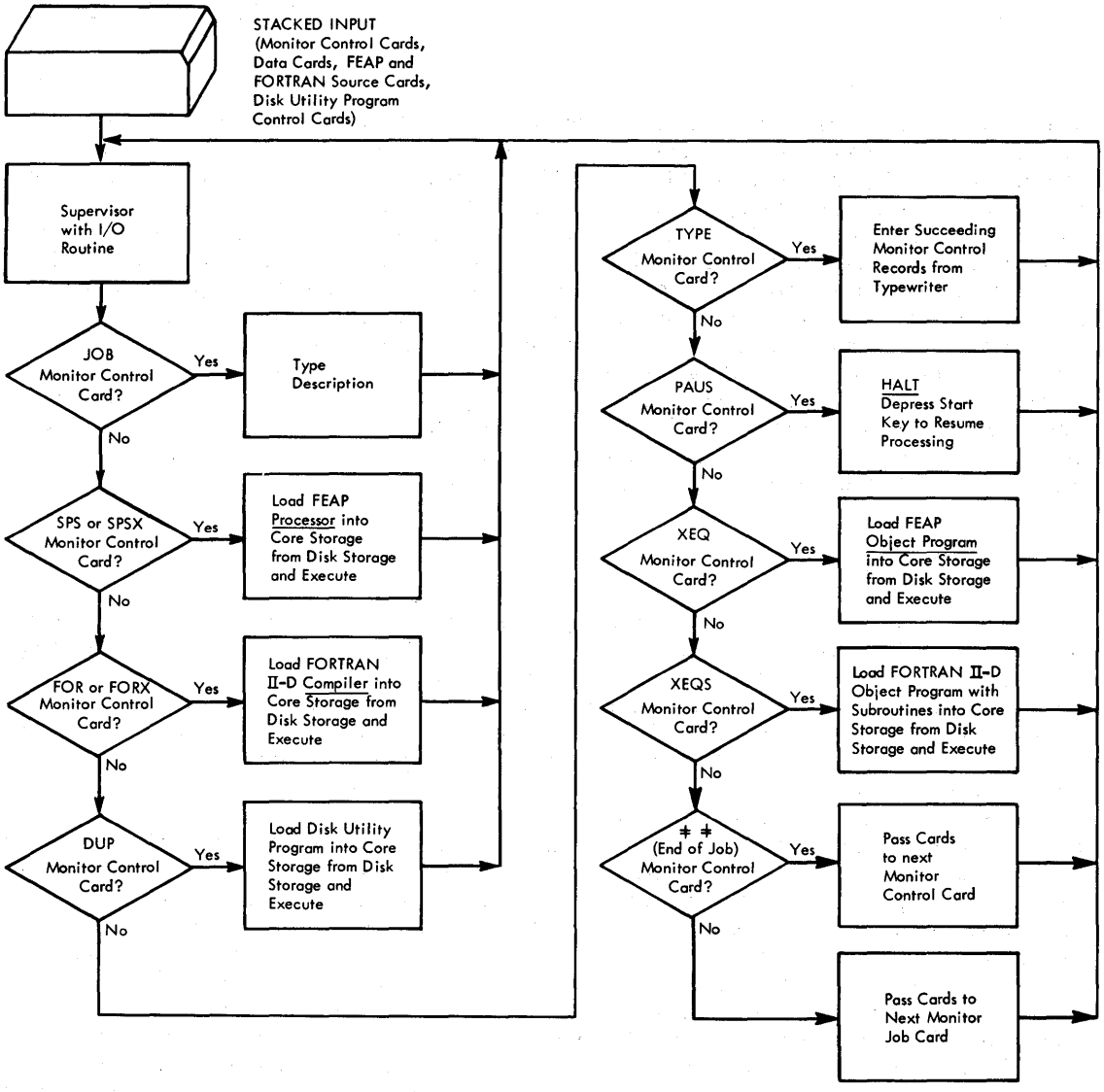


Figure 2. Supervisor Program Logic

SPS

This operation causes the FORTRAN Executive Assembly Program to be read into core storage from disk storage and to be executed. The assembled object program may be stored in disk storage and an entry made in the DIM (Disk Identification Map) table.

SPSX

The SPSX operation is similar to the SPS operation, with one exception; after the object program is assembled, it is then executed. The SPSX operation is valid only with non-process programs.

FOR

A FOR operation causes the 1710 FORTRAN II-D compiler program to be read into core storage from disk storage and to be executed. The object program can be stored in disk storage. If this occurs, an entry will be made in the DIM table.

FORX

The FORX operation is the same as the FOR operation, with one exception; the object program is executed after it is compiled. The FORX operation is valid only with non-process programs.

XEQ

The XEQ operation causes the FEAP object program, identified by the control card data, to be read into core storage from the input unit indicated in column 27, and then to be executed. Each disk-stored program, called by the XEQ operation, must have a DIM entry to enable the Supervisor program to find it. The XEQ record can be used only to call nonprocess FEAP programs.

XEQS

This operation causes a FORTRAN object program, identified in the XEQS card, to be read into core storage from disk storage, cards, or paper tape, and to be executed. Each disk-stored program called by the XEQS operation must have a DIM entry to enable the Supervisor Program to find it. The XEQS can only be used to call nonprocess FORTRAN programs.

DUP

The DUP operation causes the Disk Utility Program to be read into core storage from disk storage and to be executed.

TYPE

The TYPE operation causes a message — which requests the operator to enter the next Monitor Control record from the typewriter — to be typed, and the program stops to await keyboard input. After the operator enters the control record, the Release and Start key must be depressed to resume computer operation. All succeeding control records must be entered from the typewriter until a JOB record is entered to change the source of input.

PAUS

The PAUS operation halts the program to allow the operator to change paper tapes, load input cards, etc. Job processing is resumed by depressing the Start key.

\+ \+ (End-of-Job)

The \+ \+ (end-of-job) operation causes the message

END OF JOB

to be typed, if a job has actually started, and control is to be resumed by the Supervisor program. An End-of-Job record must follow each job. If this record is not present, erroneous results may be obtained.

Control Card Formats

JOB.	Columns	1-2	++(identification record marks)
		3-6	Operation (JOB, left-justified).
		7	Source of input, 5 = card. 3 = paper tape. 1 = typewriter.
		8-11	Module change numbers (for disk input only).
		12-31	Disk pack identification numbers (for disk input only), 12-16 drive 0. 17-21 drive 1. 22-26 drive 2. 27-31 drive 3.
		32-80	Description.

SPS, FOR, DUP, TYPE, PAUS.			
	Columns	1-2	+ +(identification record marks).
		3-6	Operation (FOR, etc., left-justified).
		7	Source of input, for FEAP, FOR, or DUP Monitor Control cards, 5 = card. 3 = paper tape. 1 = typewriter.

SPSX.	Columns	1-2	++(identification record marks).
		3-6	Operation (SPSX).
		7	Source of input, 5 = card. 3 = paper tape 1 = typewriter.

FORX.	Columns	1-2	++(identification record marks).
		3-6	Operation (FORX).
		7	Source of input, 5 = card. 3 = paper tape. 1 = typewriter.
		8	FORTRAN subroutine set identification number.

XEQ.	Columns	1-2	++(identification record marks).
		3-6	Operation (XEQ, left-justified).
		7-12	Name of user's program, to be executed (same name assigned in Equivalence table).
		13-16	DIM (Disk Identification Map) entry number. Note that either the name or the DIM entry number must be given (if program is in disk storage), but if both are given, the name takes precedence.
		17-21	Address where loading of user's program begins if program is not in core image. If not supplied, address 02402 is assumed.
		22-26	Address where execution of user's program is to begin if program is not in core image. This address must be relative to the start of the program if the program is relocatable; otherwise, the absolute entry address must be supplied.

		27	Source of input, Blank = disk. 5 = card. 3 = paper tape. (Note that card or paper tape input must be in System Output format.)
XEQS.	Columns	1-2	++(identification record marks).
		3-6	Operation (XEQS).
		7-12	Name of user's program to be executed (same name assigned in Equivalence table).
		13-16	DIM (Disk Identification Map) entry number. Note that either the name or the DIM entry number must be given (if program is in disk storage), but if both are given, the name takes precedence.
		27	Source of input, Blank = disk. 5 = card. 3 = paper tape. (Note that card or paper tape input must be in System Output format).
		28	Subroutine set identification.

++(End-of-Job).

Columns	1-2	++(identification record marks).
	3-4	Operation (++ end-of-job).

Comments Records

Comments records — in card, paper tape, or typewriter form — can be used to specify operating instructions and identify each job. Any number of these records may be inserted in front of a job in the stacked input. Usually they are inserted behind a JOB Monitor Control record.

When Comments records are encountered in the stacked input, they are typed out. The format of the Comments record in terms of cards follows:

Columns	1-2	++(identification record marks).
	3-6	Operation (blanks or any combination of letters and/or digits other than the eleven Monitor pseudo operation codes, JOB, SPS, etc.).
	7-80	Comments.

When the Supervisor Program reads a Comments card, it will pass subsequent cards until another card with + +, columns 1-2, is encountered. Therefore a Comments card should be followed by another Comments card or a Monitor Control card.

Module Change Numbers (Off-Line)

Module change numbers, punched in card columns 8-11 of the JOB cards, can be used by the operation on 1710 Systems with more than one 1311 Disk Storage Drive to alter the normal assignment of disk storage drives for any job. For example, a job that uses drive 0 in the execution of its program could use drive 1 instead of drive 0 by the entry of a JOB Monitor Control card with the appropriate module change numbers.

Card columns 8, 9, 10, and 11 of the JOB card represent disk storage drives 0, 1, 2, and 3, respectively. A change to the normal program assignment of a disk storage drive is made by punching the number of the substitute drive into the card column which represents the normal drive. Therefore, in the preceding example, a digit 1 would be punched into card column 8 to alert the program that drive 1 should be used for the job

instead of drive 0. Card columns 9, 10, and 11 could be left blank because only the assignment changes must be punched. The assignment of disk storage drives, placed in effect by a JOB card, remains in effect until changed by a succeeding JOB card.

To overlap the time required to change disk packs for one job with the processing time for a different job, the operator may choose to alternate the use of disk storage drives from one job to the next. Alternating drives is possible only when all drives are not in use for any one job. For example, assume that job A is to be followed by job B in the stacked input and the programs for both of these jobs use disk storage drives 0 and 1. Assume further that four disk storage drives are available to the 1710 System that is to perform these jobs. By entering a module change number in the JOB Monitor Control card for job B, the operator can use disk storage drives 2 and 3 for job B in place of drives 0 and 1. Therefore, while job A is being done, the operator could mount the disk packs for job B on drives 2 and 3, thus saving valuable operating time. The JOB card module change numbers should be punched

Card columns	8	9	10	11
Module change numbers	2	3		

so drives 2 and 3 will be used in place of drives 0 and 1, respectively, for job B.

If the system is redefined (DFINE control card) to utilize more than one module, a JOB control card with the module change numbers for all the drives in the system must be used. This precludes the occurrence of invalid MOD ERR messages when the additional drives are addressed. The JOB card need only be used once with these module change numbers unless substitution of one number for another, as described above, is desired.

The user must utilize the initial start procedure before the JOB card when entering module change numbers.

Using More Than One Disk Drive with the Process Control System

In order to utilize more than one disk drive with the on-line IORT, some additional operations must be performed (see DUP DFINE Description). These operations consist of reading a portion of the on-line IORT program into core from disk, changing it to specify the drive units that are on the particular system, and writing the data back on disk.

These changes can be made using the DUP DALTR routine. The sector that must be specified is 17020. The section that must be specified is 01. The contents of this section initially are 00$\bar{0}$1$\bar{0}$0$\bar{0}$0$\bar{0}$0. To introduce the ability to handle a second drive, this section must be changed to 00$\bar{0}$1$\bar{0}$3$\bar{0}$0$\bar{0}$0. Up to four drives can be used if the fields are changed to 00$\bar{0}$1$\bar{0}$3$\bar{0}$5$\bar{0}$7. An example of the typewriter log produced when DALTR is used to change to a four drive system is shown below.

```
++JOB

++DUP

*DALTR

SECTOR
117020R-S

1S T.HALF 00$\bar{0}$1$\bar{0}$0$\bar{0}$0$\bar{0}$0 $\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$1$\bar{0}$ 0502$\bar{0}$90$\bar{1}$70 00$\bar{1}$0336+$\bar{0}$0 $\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0 ORIGINAL
2ND.HALF $\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0 $\bar{0}$0+3340000 0001023902 1910010048 6241430+34 ORIGINAL

SECTION
01R-S
00$\bar{0}$1$\bar{0}$0$\bar{0}$0$\bar{0}$0 TYPE CHANGE
00$\bar{0}$1$\bar{0}$3$\bar{0}$5$\bar{0}$7R-S
1S T.HALF 00$\bar{0}$1$\bar{0}$0$\bar{0}$0$\bar{0}$0 $\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$1$\bar{0}$ 0502$\bar{0}$90$\bar{1}$70 00$\bar{1}$0336+$\bar{0}$0 $\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0 ORIGINAL
1S T.HALF 00$\bar{0}$1$\bar{0}$3$\bar{0}$5$\bar{0}$7 $\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$1$\bar{0}$ 0502$\bar{0}$90$\bar{1}$70 00$\bar{1}$0336+$\bar{0}$0 $\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0$\bar{0}$0 CORRECTED
```

(continued top of next page)

```
2ND.HALF  0000000000  00+3340000  0001023902  1910010048  6241430+34  ORIGINAL
2ND.HALF  0000000000  00+3340000  0001023902  1910010048  6241430+34  CORRECTED

SECTION
+R-S
DISK SECTOR 117020 CORRECTED
SECTOR
+R-S
END OF JOB
```

Module Change Numbers On-Line

The on-line IORT may be changed to substitute one drive for another using the procedure described under Using More Than One Disk Drive with the Process Control System. In this case the field containing 01 in the example would be changed to 03 if the first drive is to substitute for the second drive, to 05 if the first drive is to substitute for the third drive, or to 07 if the first drive is to substitute for the fourth drive. The IORT will then cause data for the logical drive (i. e. , the 2nd, 3rd, or 4th drive) to be read or written on the pack that is physically on drive one. Analagous substitutions of drives two, three and four can be made. Once the drive assignments have been made they remain in effect until the on-line IORT is reloaded from cards (paper tape) or until the user changes these assignments using the procedure described here. It is also necessary to follow the DUP DFINE procedure and the Monitor Module Change Numbers (off-line) to convert the entire FORTRAN Executive System to multiple disk drive operation.

Disk Pack Identification Numbers

Card columns 12-31 of the JOB Monitor Control card can be punched with the four 5-digit disk pack identification numbers, one identification number for each disk storage drive.

Disk Pack Identification Number	Disk Storage Drive
Card columns 12-16	0
17-21	1
22-26	2
27-31	3

The disk pack identification number from the JOB card is compared with the identification numbers recorded on the respective disk packs. If the proper disk packs are not attached by the operator, the Supervisor will halt for operator instructions. If it is desired to omit this check for any disk storage drive, the card field representing the disk storage drive may be left blank. When the operator enters a module change number, no change to the disk pack identification numbers (card columns 12-31) is required.

STACKED INPUT

Stacked input consists of control records (Monitor Disk Utility Program, FEAP, and FORTRAN), source programs, object programs, and data arranged logically by job. Each off-line job consists of phases which must fit into one of the following categories:

1. FEAP source program(s) to be assembled.
2. FORTRAN source program (s) to be compiled.
3. Disk Utility routine(s) to be executed.
4. FORTRAN or FEAP object program(s) to be called from disk storage and executed.

The order in which jobs are executed is not important, i.e., a Disk Utility routine may be executed before a FORTRAN compilation or vice versa. Jobs are executed in the order in which they are encountered in the stacked input. Each job must be preceded by a JOB card and followed by an end-of-job (++++) card.

Job Arrangement

A Disk Utility, FORTRAN or FEAP job is always represented by at least three Monitor Control records.

	Type	Purpose
1.	JOB	Identify beginning of job
2.	DUP, SPS, SPSX, FOR, FORX, XEQ, or XEQS	Transfer control to Disk Utility Program, FORTRAN compiler, FEAP or user's object program
3.	End-of-job	Identify end of job

In addition to the Monitor Control records, there may be one or more Disk Utility Program, FEAP, or FORTRAN system control records for each job. These records are a part of the input for the individual system.

TYPE or PAUS Control records may be inserted immediately preceding any of the records in the above sequence. Any number of Comments records may be inserted in front of Type 2 records. Source programs or input data can be entered immediately following Type 2 records.

The following three points must be taken into consideration when arranging the input for any job.

1. All Monitor Control records, with the exception of the records that follow a TYPE Control record, must be read from the same input source. The input source can only be changed at the beginning of each job or by a TYPE control record.
2. A job, with the exception of a Disk Utility job, may consist of several system functions possibly terminated by execution of a user's program. Execution of a user's program is considered as the end-of-job. If any cards remain in the stacked input for a job when it is ended in this manner, they will be passed without processing. Processing resumes with the first Monitor Control card of the next job in sequence.
3. If an error is detected in an FEAP assembly or FORTRAN compilation, the resulting object program or any programs that follow within the job cannot be executed.

MONITOR CONTROL RECORD ANALYZER ROUTINE

This routine, a part of the Supervisor Program, is used to read the Monitor Control records and Comments records (identified by + + in columns 1-2) to analyze these records, and to perform the operations or transfer control as directed by the pseudo operation codes. The first Monitor Control record is read from the input source that is specified by the operator when the Supervisor Program is originally loaded into core storage from disk storage to start the entire operation. Subsequent Monitor Control records and Comments records are read from the same input source until a JOB record changes the input source by specifying a different "source of input" or a TYPE card is encountered. Reading of Monitor Control records continues from the new source until again changed by another JOB Control record.

When the input source is the "typewriter," the Monitor Control Record Analyzer routine types the message:

The operator may then enter the next control record. The record is not typed out if the entry is made by the typewriter.

If the operator makes a mistake while entering the record, he may correct the error by turning on Program Switch 4 and depressing the R-S key on the typewriter. Switch 4 should then be turned off and the entire record re-entered.

When an SPS, SPSX, FOR, FORX, DUP, XEQ, or XEQS control record is read, control is transferred from the Analyzer routine to the individual program specified by the control record. Control is returned to the Analyzer routine after the program is executed. When control is returned, the Analyzer routine will pass records, provided the input source is other than the typewriter, until a Comments or Monitor Control record is encountered in the stacked input. Therefore, the last job to be executed should be followed by a TYPE or PAUS control card. If this control card is not present, the 1710 will stop on a Read Select instruction, expecting another control card.

Error Messages

During execution of the Monitor Control Record Analyzer routine, certain error messages may be typed. After typing a message, the 1710 will stop if any operator action is required. A list of these messages, the conditions which cause them, and the corrective actions required of the operator, follows.

Message	ERROR IN FIELD AT COL. XX. SET SW4 TO IGNORE, OFF TO RE-ENTER CARD
Cause	An illegal character has been detected in a JOB Record data field.
Action	To ignore the error turn Program Switch 4 on and depress the Start key. The message "CONDITION IGNORED" is typed and processing continues. To correct the error, turn Program Switch 4 off and depress the Start key. The Monitor Control record input source will be changed to the typewriter, and the operator may then re-enter the control record. If it is desired to read succeeding records from the original input source, column 7 must identify the input source.
Message	PACK NUMBER. ERROR ON MODULE X. SET SSW4 TO IGNORE OFF TO RECOMPARE
Cause	Disk pack identification numbers compare "unequal."
Action	To ignore the error, turn Program Switch 4 on and depress the Start key. The message "CONDITION IGNORED" is typed and processing is resumed. To correct the error, place the correct disk packs on the disk drives and depress the Start key. The disk pack identification number will again be checked by the program.
Message	END OF JOB
Cause	The end of a job has been reached. (This message will not be typed, if the input source is the typewriter.)
Action	None required.
Message	CANNOT RESTORE COMMON — RESET AND START TO RE-TRY
Cause	Common area does not read into core storage from disk storage correctly.
Action	Depress Reset and Start keys to retry the read operation.
Message	EXECUTION
Cause	Loading and execution of user's object program has started.
Action	None required.

Message	JOB CARD GROUP ONLY
Cause	Control Record Analyzer routine is expecting a JOB, TYPE, or PAUS Monitor Control record, but it does not find one.
Action	Enter JOB, PAUS, or TYPE, Monitor Control record from typewriter and depress the Release and Start keys.

Message	ERROR IN FIELD AT COLUMN XX. PHASE TERMINATED
Cause	A Monitor Control record contains an invalid code in the field starting at column XX.
Action	The phase is skipped and the supervisor will pass records from the control record source until it encounters the next Monitor Control record.

Message	EXECUTION IS INHIBITED
Cause	An error has occurred within a job which may prevent successful execution of the user's object program.
Action	None required. No user object program can be executed until the next JOB Monitor Control record is encountered.

Message	OBJECT DIM ERROR PHASE TERMINATED
Cause	The Supervisor is unable to find the DIM entry specified by an XEQ or XEQS control record.
Action	None required. The phase in which the error occurred is terminated and processing continues.

Message	OBJECT NAME ERROR PHASE TERMINATED
Cause	The Supervisor is unable to find a name in the Equivalence table which corresponds to the name supplied in an XEQ or XEQS control record.
Action	None required; the phase is terminated.

Message	ENTER MONITOR CONTROL RECORD
Cause	A "$$TYPE" Monitor Control record has been encountered.
Action	Enter a Monitor Control record from the typewriter. (Monitor input source is changed to the typewriter.)

Message	SYSTEM DIM ERROR PHASE TERMINATED
Cause	Supervisor is unable to find DIM entry for FEAP, FORTRAN compiler, or Disk Utility Program.
Action	None required; the phase is terminated.

I/O Routine

The I/O routine is designed to relieve programmers of the necessity for writing input/output subroutines. The I/O function is performed automatically by the I/O routine. Therefore, the programmer can concentrate on describing his files and disregard the actual operation of the I/O function. Provision is also made in this routine for error detection and correction. If Parity, Wrong-Length Record Check, or Address Check disk errors occur in a disk operation, the routine will repeat the operation which had the error, up to nine times, in an attempt to correct the error. The Monitor System uses this routine for I/O operations.

The I/O functions performed by the I/O routine include reading and punching cards or paper tape, reading or writing typewriter, reading or writing disk records, and seeking disk cylinders. These functions may be used in a FEAP object program by entering I/O macro-instructions (GET, PUT, SEEK, or CALL) in the user's source program. These macro-instructions, as well as the associated declarative statements for defining declarative constants (DTN, DTA, etc.), are described in the FEAP section.

All linkages for I/O routines are generated automatically through the use of macro-instructions in SPS source programs or the I/O statements (e.g., FIND, RECORD, FETCH, PUNCH, READ, etc.) in FORTRAN source programs. The data and addresses supplied in a macro-instruction or the parameters in a FORTRAN statement are incor-

porated into the linkage instructions where they are made available for use by the I/O routine.

The I/O routine to be used with the Executive Control programs for on-line (process control) is different from the off-line (nonprocess control) I/O routine in several ways. The error counts that are maintained by the on-line I/O routine are coordinated with the System Alert Control program error counts and are outputted for CE examination when the CE interrupt is initiated. The error messages are shortened to conserve 1710 machine time, and the error procedures are changed to conform to use with a continuously running control program.

The entry addresses along with other important addresses and constants have been maintained constant between the two routines. The on-line routine cannot be used to load a program that is in System Output format, nor can it call in the Control Card Analyzer routine.

The differences in methods, error messages, and error procedures are described in the following paragraphs. Unless a function is specifically excluded, it is valid for both the on-line and off-line routines.

Each time the I/O routine is entered as the result of an FEAP macro-instruction or FORTRAN statement, the read, write, and parity check indicators are turned off. If a read or write error occurs that cannot be corrected without operator intervention during off-line operation, an error message is typed and the program halts. A restart procedure is specified for all off-line error conditions (see I/O Error Routine). An error count is maintained by both the on-line and off-line I/O routine for inspection by the user or for diagnostic analysis by an IBM Customer Engineer.

In addition to using the I/O routine with FEAP macro-instructions and FORTRAN statements, the routine may be used by coding the general form of I/O routine linkage directly in the user's program.

I/O Routine Linkage

General Form:

```
TFM     IORT, * + 23
B       ENTRY, DEF, 7
```

IORT is the address (00565) of a 5-position storage area in the I/O routine.

ENTRY may be any one of the four possible entry points in the I/O routine represented by the following symbolic addresses:

Symbolic Address	Actual Address	Function
IORBC	00520	Write record into disk storage with Read-Back Check.
IOPT	00532	Write a record to an output device.
IOSK	00554	Seek a disk record.
IOGT	00566	Read a record from an input device.

DEF can be the address of any I/O declarative constant (see I/O Constants).

CALL LINK or CALL LOAD Linkage. These linkages are usually used to call programs from disk storage, with or without execution. Linkages may be in either a short or long sequence form. Both forms are alike with the exception that the long sequence form contains a relocation address.

Short Sequence

```
TFM          IORT, * + 19
B7           IOCAL
DC           1, M0
DC
 or  }       1, M1
DSC
DSC          1,0
DC           5, IIII @
```

Long Sequence

```
TFM          IORT, * + 19
B7           IOCAL
DC           1, M0
DC
or           1, M1
DSC
DC           1,0
DC           4, IIII
DSA          LLLLL
DSC          1, @
```

IOCAL is an entry to the I/O routine
M_0 M_1 is a constant $\bar{3}2$ for CALL linkages, M_1 is flagged for CALL LINK only.
IIII is the DIM entry number of the program to be called.
LLLLL is the relocation core storage address where the program is to be loaded.

If the short sequence is used to call a relocatable program, LLLLL is assumed by the I/O routine to be the address contained in the "high" indicator field of the Communications Area. If the long sequence is used to call a core image program, the I/O routine will disregard LLLLL.

CALL EXIT Macro-Instruction Linkage.

```
B7          MONCAL
```

MONCAL (core storage address 00796) is an entry to the I/O routine which will call in the Monitor Control Record Analyzer routine.

CALL EXIT linkage is used at the end of the execution of an off-line object program to return control to the Monitor Control Record Analyzer routine to read another Monitor Control record. If, during execution of an object program, an error is encountered which will not allow normal exit to the Analyzer routine, the operator may manually branch to MONCAL (00796) to resume processing.

I/O Constants

An I/O constant for card, paper tape, or typewriter consists of eight digits.

$$\bar{C}CCCC\bar{M}_0M_1 \ \ddagger$$

CCCCC is the address of an I/O area.
M_0 M_1 is one of the following codes which identifies the operation:

$\bar{0}0$	Typewriter Numerical
$\bar{0}2$	Paper Tape Numerical
$\bar{0}4$	Card Numerical
$\bar{0}6$	Typewriter Alphameric
$\bar{0}8$	Paper Tape Alphameric
$\bar{1}0$	Card Alphameric

Disk I/O constants may be in any of the following four forms:

1. M_0 M_1 $\bar{D}$DDDD $\ddagger$
2. M_0 M_1 $\bar{D}$DDDD$\bar{L}$LLLL $\ddagger$
3. M_0 M_1 0 $\bar{I}$III $\bar{L}$LLLL $\ddagger$
4. M_0 M_1 0 $\bar{I}$III $\ddagger$

$\overline{D}$DDDD is the address of the leftmost position of the associated disk control field.
$\overline{L}$LLLL is a relocation core storage address of a program to be called.
$\overline{I}$III is the DIM entry number of a program to be called.
M_0 and M_1 provide various disk options for the user. A list of these codes and their associated options follows:

M_0 (code)	Option
0	Add the starting address of the work cylinders from the Communications Area (core positions 422-425 off-line, and 402-405 on-line) to the sector address in the disk control field. (Used with constant types 1 and 2 only.)
1	Same as option zero, except the "high" indicator in the Communications Area will also be updated for disk read operations only. This indicator is merely a field which contains the core storage address of the highest position to be loaded plus one.
2	Use the sector address in the disk control field for the disk operation (SEEK, READ, or WRITE).
3	Use the sector address in the disk control field for the disk operation. Also, update the "high" indicator in the Communications Area for read operations only.
$\overline{n}$	A flag over the code n, ($\overline{n}$ = 0, 1, 2, or 3) causes the read/write heads to be repositioned to an assigned cylinder (specified in the Communications Area) after any disk I/O operation, except seek.

M_1 (code)	Option
0	Disk read or write in sector mode with WLRC. NOTE: The user must place a group mark (≢) in the core storage location following the last character position of the last sector of the record.
2	Disk read or write in sector mode without WLRC.
4	Disk read or write in track mode with WLRC. NOTE: The user must place a group mark (≢) in the core storage location following the last character position of the last sector of the record.
6	Disk read or write in track mode without WLRC.
$\overline{n}$	A flag over code n ($\overline{n}$ = 1, 2, 4, or 6) causes the I/O routine to branch to a given address after a disk read operation. The given address will be the "execution address" if an extended disk control field is used. Otherwise, it will be the "core address" of the disk control field. If code n is unflagged, the I/O routine will branch to the first instruction following the disk operation calling linkage in the object program. If the entry address is not specified, the entry is made to the (possibly relocated) first card address of the deck to be loaded.

Disk Control Field

The disk control field, associated with I/O constants, types 1 and 2, may be in either of the following formats:

$$D\bar{D}DDDD\bar{S}SS\bar{C}CCCC\pm$$
$$D\bar{D}DDDD\bar{S}SS\bar{C}CCCC\bar{E}EEEE\pm$$

DDDDDD is the first sector address of the data or program.

SSS is the number of sectors to be read or written.

CCCCC is the core address (must be an even-numbered address) of the data or program.

EEEEE is the execution address where program execution is to continue after a disk read operation is completed. The second disk control field, known as an extended disk control field, is used when M_1 of the I/O constant is flagged.

Card I/O

Cards are read or punched in alphameric or numerical form from a user-specified constant (generated from an I/O declarative statement) designated in general linkage. If a punch error is detected during a write instruction, the instruction is again executed to correct the error. If the error persists or a parity error occurs during a write operation, an error message is typed and the program halts if off-line, or continues if on-line (see I/O Error Routines). Error messages will be typed for all read errors.

Typewriter I/O

A specified I/O declarative constant designated in general linkage will be used by read or write typewriter instructions (alphameric or numerical). If a read error or a parity error occurs during reading, the program will type an error message and branch back to the read instruction and await entry of data. The operator can then type in the data and return control to the program. If a parity error or write check occurs during writing, it will be counted and the indicators will be turned off — but the program will not halt. Control operations (RCTY, SPTY, TBTY) are not executed in the I/O routine. These must be handled in the main program coding.

Paper Tape I/O

Paper tape is read or punched in alphameric or numerical form from a specified I/O declarative constant designated in general linkage. If a parity read error occurs during a read operation, an error message is typed and the program halts if off-line or continues if on-line.

Disk Storage I/O

Disk storage will be read or written as specified by the I/O declarative constant designated in general linkage. Also, disk seek operations will be initiated to disk addresses contained in the I/O declarative constant designated in general linkage. For a CALL macro-instruction, disk data records or programs will be written in the area of core storage designated by the relocatable address in CALL linkage. If this address is not present for a relocatable program, the processor selects the address. If the relocation address is present but the program is an on-line program or is not relocatable (i.e., it is in either Absolute or Core Image format), the relocation address is ignored and the program is stored at the core address specified by the DIM entry.

Disk storage indicators are reset by the I/O routine. If the Cylinder Overflow indicator (38) is turned on before the sector count reaches zero, a seek to the next cylinder is initiated and reading or writing is resumed. If read, write, or parity indicators, or

indicators 36 or 37 are turned on, the instruction associated with the error will again be executed up to nine times. If the error persists, an error message is typed and the program halts if off-line or branches to SAC if on-line.

The seek only linkage

```
TFM   IORT, * + 23
B     IOSK, DEF, 7
```

will allow computing time and seek time to overlap. DEF refers to any disk I/O constant.

The cylinder location of the arm on each drive is entered into a list by the I/O routine. Every time the I/O routine executes a disk operation, the current entry cylinder is compared to the previous cylinder and the arm is instructed to SEEK only if it is not already located at the current cylinder.

Repositioning of Access Arms

The I/O routine contains four 2-digit cylinder indicators that can be used to reposition the access arm on each of the four possible disk drives to a new cylinder following a read or write operation. The four cylinder indicator core storage locations and their associated drives follow:

Indicator Addresses	Drive
00512 - 00513	0
00514 - 00515	1
00516 - 00517	2
00518 - 00519	3

These indicator positions are reset to $\overline{0}\overline{0}\overline{0}\overline{0}\overline{0}\overline{0}\overline{0}\overline{0}$ by the Monitor Control Record Analyzer routine. Therefore, a program which uses other cylinders for repositioning must provide for changing the indicators.

Repositioning the access arm following a GET or PUT macro-instruction is optional. If M_0 of the I/O constant used by a GET or PUT is flagged, the read/write heads will be repositioned.

Full Track Operation

If any I/O operation is to be attempted with the Write Address light on, the programmer must set a flag at OLDDA + 14 (core position 00455) before entering the I/O routine. The flag will prevent accumulating error counts (which is a write disk sector operation). The flag must be cleared before terminating the routine in which the Write Address light "on" condition is present.

If an I/O operation is attempted with the Write Address light on, no flag present at OLDDA + 14, and an indicator 06, 07, 16, 17, 36, 37, or 38 on or turned on by the I/O operation, the program will stop with the instruction at 00728. To save the error count, the operator must (1) turn the Write Address light off, (2) depress the STOP/SIE key, (3) turn the Write Address light on, if desired, and (4) depress the Start key to resume automatic operation.

I/O ERROR ROUTINE

Each time the I/O routine begins execution, it tests indicator 19 (any check) to determine if an error had occurred prior to entry. If the indicator is on, the I/O error routine will be called into core storage and executed. This routine records a count of errors by type (for indicators 06, 07, 16, 17, 36, 37, and 38), and provides the necessary error messages and corrective operating options. In addition, the error routine turns off the individual indicators (06, 07, 16, 17, 36, 37, and 38) by testing them.

After an I/O function is executed, indicator 19 is again tested. If it is on, the I/O error routine is entered to process the error.

Error Detection and Correction

During execution of the I/O error routine, error messages are typed to describe errors and the operator is allowed to intervene to decide how errors should be treated by the program. A list of error correction options available to the operator for use with off-line programs follows.

Error Correction Code	Option
00	Ignore the error. When this option is used, the I/O routine will finish processing the I/O operation as though the error had not occurred.
05	Re-execute the I/O operation. If an error recurs during the next execution, an error message is again typed, the computer stops, and the operator can exercise the same option or another option.
10	Skip this phase of the job if error occurs at system time (FEAP assembly, FORTRAN compiling, Disk Utility Program, or Supervisor Program execution time) and return control to the Monitor Control Record Analyzer routine and pass records to the next Monitor Control record.
15	Discontinue execution and return control to the Monitor Control Record Analyzer routine and pass records to the next JOB Monitor Control Record.
20	Continue processing by branching to a specified core storage address without further processing of the I/O request. When this option is exercised, the operator enters the 5-digit branch address from the typewriter.

After each error message is typed, the computer halts. The operator then depresses the Start key, enters a 2-digit error correction code from the typewriter, and depresses the R-S key to resume processing.

If an error is made while entering a 2-digit correction code, it may be corrected by turning Program Switch 4 on, depressing the typewriter R-S key, turning Program Switch 4 off, and re-entering the 2-digit code.

The I/O error routines have the facility to handle any of the following errors.

Entry Check
Typewriter write
Typewriter read
Paper tape read
Card write
Card read
Cylinder overflow
Write error count
Illegal DIM entry
System
Unavailable disk drive

In addition, the off-line I/O routine will detect Monitor Control records read by user programs. Error messages, conditions, and corrective operator action associated with

each type of error are described as follows:

Entry Check. If indicator 19 (any check) is on, when tested in the preprocessing phase, the message

ENT ERROR $\bar{0}6\bar{0}7\bar{1}6\bar{1}7\bar{3}6\bar{3}7\bar{3}8$

is typed on the console typewriter for off-line programs. Each pair of indicator numbers is flagged in the leftmost digit. If an indicator was on when tested, the rightmost digit will also be flagged.

A shorter message will be typed in the on-line program. This message consists of an N followed by the numbers 2 through 8, if all seven indicators are on, or just as many numbers, corresponding to the indicators above, that are on when the I/O routine was entered.

Typewriter Write. For this error, no error message is typed; however, the error is automatically indicated by over-printing the error character(s) with a horizontal line.

Typewriter Read. For this error, the message

TYP ERR $\bar{0}6\bar{0}7\bar{1}6\bar{1}7\bar{3}6\bar{3}7\bar{3}8$

is typed for off-line programs. To restart the computer, the operator exercises one of the error correction options. On-line, the message

E9

is typed, and the program continues.

Paper Tape Read. The message

PTR ERR $\bar{0}6\bar{0}7\bar{1}6\bar{1}7\bar{3}6\bar{3}7\bar{3}8$

is typed for off-line programs. The operator must backspace the tape to the beginning of the record before exercising error option 05. On line, the message

E12

is typed, and the program continues.

Paper Tape Write. The off-line program prints

PTP ERR $\bar{0}6\bar{0}7\bar{1}6\bar{1}7\bar{3}6\bar{3}7\bar{3}8$

and the program stops. The on-line program prints

E13

and continues.

Card Write. For this error, the I/O error routine retries the operation once for a write check error (indicator 07). If the error is corrected by the retry, control is returned to the I/O routine; if the error is not corrected, the message

CDP ERR $\bar{0}6\bar{0}7\bar{1}6\bar{1}7\bar{3}6\bar{3}7\bar{3}8$

is typed in the off-line program. The error option can then be exercised by the operator. On-line, the message

E11

is typed, and the program continues.

Card Read. For this error, the I/O error routine retries the operation once for a read check error (indicator 06). If the error is corrected by the retry, control is returned to the I/O routine; if the error is not corrected, the message

CDR ERR $\bar{0}6\bar{0}7\bar{1}6\bar{1}7\bar{3}6\bar{3}7\bar{3}8$

is typed in the off-line program. The error option can then be exercised by the operator. On-line, the message

E10

is typed, and the program continues.

Cylinder Overflow. For this error, the I/O error routine tests to determine if a legitimate overflow has occurred during a disk read or write operation. For a legitimate overflow, a seek operation to the next cylinder is automatically initiated and reading or writing continues. A maximum of three seek operations will be performed if sufficient core storage is available to accommodate the data being read or written.

If the disk read or write operation results in an attempt to read or write data beyond the highest sector address of the addressed disk module, the message

DSK OFL

is typed in the off-line program. All error correction options, except 05 are available to the operator. If 05 was inadvertently entered, it would have the same effect as error correction option 00. On-line, the message is

E14

and the program branches to the System Alert Control program which executes the users error procedure for the mainline core load in progress.

For a disk error, which is other than a legitimate overflow, the disk read or write operation causing the error is retried up to nine times; if this fails to correct the error, the message

DSK ERR XXXXX $\bar{0}6\bar{0}7\bar{1}6\bar{1}7\bar{3}6\bar{3}7\bar{3}8$

is typed in the off-line program where XXXXX is the 5-digit return address to the object program. A flag is typed over the leftmost position of each pair of indicator numbers. The indicator(s) which identifies the type of error will also be flagged in the rightmost position. If indicator $\bar{3}\bar{8}$ is flagged in its rightmost position, it may mean either of two things:

1. A legitimate overflow did occur, but another type of error occurred in attempting to transmit data to or from the succeeding cylinder.
2. A machine malfunction occurred.

If the operator exercises error correction option 05, but this does not correct the error, he should turn the Disk, Parity, and I/O Check switches to STOP, and again exercise option 05. The console lights may then be examined to determine the nature of the error.

The on-line routine types the same message as for an entry error (N2345678) except that an E appears instead of an N. The program then branches to the System Alert Control program for further corrective action.

If a legitimate cylinder overflow condition occurs, a seek operation to the next cylinder is automatically initiated and reading or writing continues.

In the on-line program, any disk error which is not corrected by retrying the operation will cause an entry to SAC. To permit the user to distinguish such IORT disk errors from other causes for entry to SAC, a single digit (a one) is placed in the last core storage position of the computer. The FORTRAN program can detect this indicator by testing the first variable of COMMON using an IF statement. This variable must be a fixed point variable and must be included in any definition of COMMON for on-line use. If a FORTRAN test is made of the variable, the variable must be re-established as zero by the user. Since only a single digit is placed in core storage when the disk error is detected, it is necessary for the user to set the variable to zero when the on-line program is started, as well as after the variable has been changed because of a disk error. The indicator is set for any error detected by on-line IORT which results in entry to SAC from IORT.

Write Error Count. If an error occurs while the I/O error routine is writing the error count in disk storage, the message

BAD DISK WRITE. RESET START

will be typed in the off-line program. In this case, the operator does not exercise an error correction option, but he must:

1. Clear the Select-Lock light, if it is on.
2. Depress the Reset and Start keys.

On-line the error message is

E18

and the program continues.

Illegal DIM Entry. If the user supplies an illegal DIM number (not in DIM table) in a CALL statement, the I/O routine will transfer control to the I/O error routine and message

MAP ERR XXXXX IIII

is typed in the off-line program where XXXXX is the core storage position immediately following the call linkage, and IIII is the illegal DIM entry number. The operator then enters error correction code 00 and depresses the Release and Start keys. The computer will again halt. The operator must then type in a corrected 4-digit DIM entry number and depress the Release and Start keys. When this error occurs in an on-line program, the message

E17

is typed. In addition to an illegal DIM entry indication, this message indicates an illegal disk control field in the on-line program. After the message the I/O routine branches to the System Alert Control program for further error analysis.

Form C26-5879-1
Page Added 5/14/65
By TNL N26-0120

System. If the I/O error routine cannot interpret the nature of the error, the message

IMP ERR

is typed in the off-line program and control is returned to the Monitor Control Record Analyzer routine without stopping to allow operator intervention. On-line, the message

E15

is typed, and control is returned to the calling program.

In addition to the system error just described, the computer may halt in the I/O routine at core storage address 00467 without typing a message. This halt occurs if a read error occurs while the I/O routine is reading one of its subroutines from disk storage. To retry the operation, the operator should:

1. Clear the Select-Lock light if it is on.
2. Depress the Reset and Start keys.

If this error persists, it may mean either of two things.

1. The user inadvertently altered the I/O routine in core storage.
2. A machine malfunction occurred.

In the on-line program the message

E1

is typed and the operation is repeated.

Unavailable Disk Drive. If the programmer specifies a logical module for which there is no physical disk storage drive, the message

MOD ERR XXXXX

is typed in the off-line program where XXXXX indicates the return address to the object program. The operator must enter the error correction code 00 and depress the Release and Start keys. The computer will again halt. The operator must then enter a corrected 1-digit drive code and depress the Release and Start keys to continue.

Illegal Drive Code. If the user gives an illegal drive code in the disk control field for a disk operation, the message

MOD ERR XXXXX

is typed where XXXXX is the 5-digit return address to the object program. To continue, the operator should enter an error correction code 00 and depress the Release and Start keys. The computer will halt to allow the operator to enter a corrected 1-digit drive code from the typewriter. Depress the Release and Start keys to resume operation.

In the on-line program, the message

E16

is typed for either an unavailable disk drive or for an illegal drive code, and the routine branches to the System Alert Control program for further analysis.

Control Record Trap. To prevent the I/O routine from inadvertently reading a control record as a data record in the off-line program, the I/O routine is designed to trap control records, if they are read from the Supervisor input source. Each record read is tested for ++ in its first two positions. If present, control is transferred to the I/O error routine and the message

TRP ERR

is typed. If the control record was read in numerical mode and it was not an end-of-job record (++++), an additional message is typed:

MUST RELOAD

The operator then depresses the Start key and re-enters the record. The Monitor Control Record Analyzer routine assumes control and processes the trapped control record. If the control record was read in alphameric mode, it is processed in the normal manner by the Supervisor.

Loading Error. If the on-line IORT is assembled and loaded after the SAC program has been loaded, the message

SAC

is typed and the machine halts. It is necessary to reload the Skeleton Executive (see Assembly and Loading Procedures).

ERROR COUNT RETRIEVAL ROUTINE

Each time an error is detected by the off-line I/O error routine, an error count is incremented by one. An error count is maintained for each of the following error indicators:

06	Read Check
07	Write Check
16	MBR-E Check
17	MBR-O Check
36	Address Check
37	WLR-RBC
38	Cylinder Overflow

The error counts can be typed out and reset to zeros by entering the following instructions and data from the typewriter:

34 00032 00701
36 00032 X0702
49 00070 0
11975400100046 (disk control field)

X is the drive code for the FORTRAN Executive System.

The Release and Start keys are depressed to start the operation. The seven indicator counts are then typed in sequence in 14 consecutive positions with a flag over the leftmost position of each count.

X̄X̄X̄X̄X̄X̄X̄X̄X̄X̄X̄X̄X̄X̄

the error counts are reset to zeros after the typeout.

On-line, the error count is coordinated with the System Alert Control program error counts and typed out via the CE interrupt program.

Loader Routine

The Loader routine, a part of the Supervisor program, is used to load user's object programs into core storage from cards, paper tape, and disk storage. This routine is only available for use by off-line programs. To perform the loading function, the Loader routine is called into core storage whenever an object program is to be loaded into core storage. The user's object program could be any program in System Output format.

Programs are sequence checked as they are loaded if input is from cards. This check is performed on the last five digits of each input record. If any records are out of sequence, an error message is typed and the operator is allowed to intervene to correct the sequence error. Patch cards may be interspersed with other cards of an object program to be loaded.

The sequence number of card input appears in columns 76-80. Sequence numbers start with 0̄0001 and must have a flag over their leftmost position in order to be sequence checked.

SYSTEM OUTPUT FORMAT

The general output format for FORTRAN and FEAP programs is shown below.

Columns	1-5	Address of data
	6	Indicator code
	7-8	Length of data
	9-75	Data
	76-80	Sequence number

Patch cards should be prepared in the same format. However, the sequence number must not include a flag in column 76. All patch cards must precede the card that defines the end of a relocatable program (Indicator Code 6). The entries shown above are described as follows:

NOTE: The descriptions given here are in terms of cards; however, paper tape and disk formats are the same with the exception of the sequence number.

Address of Data. This entry will always refer to the location where the first digit of data on the card is to be loaded. This entry will appear in columns 1 through 5 of a reloadable card. The address is an absolute address, i.e., no relocation increment (presuming this program is relocatable) has been added yet.

Indicator Code. This 1-digit entry is used to either define the type of data that is to follow or to convey certain loading instructions to the loader. There are thirteen different indicator codes that may be used; some are applicable to SPS II-D only (see Indicator Codes).

Length of Data. This field is used in conjunction with certain indicator codes (1, 2, $\bar{2}$, 3, $\bar{3}$, 4, $\bar{4}$) to specify how many digits of data are to follow. With other indicator codes, this field becomes part of a larger field and assumes a different role.

Data. This field contains actual data to be loaded. Data may be instructions, constants, etc., depending upon the indicator code. All instructions for relocatable programs will contain flags over O_0, O_1 of the operation code to specify if the P and Q addresses, respectively, should be incremented by the relocation address. If patch cards are prepared, these flags must be punched for addresses to be adjusted. Instructions of programs in absolute format must not be flagged.

Indicator Codes

Although the output format is shown divided into specific fields, these same fields do not always make up the columns that are indicated. As will be seen by the descriptions of the various indicator codes, the format varies considerably as the type of data on the card changes.

The indicator codes are described as follows:

NOTE: Those codes marked with an asterisk are used in FEAP output only.

- ± - This digit indicates that a change is being made in the sequence of loading addresses for the program. The five digits that follow the record mark denote the new address or origin. After the 5-digit address, there will be another indicator code to define the data that follows.
- $\bar{\pm}$ - This digit is used whenever the data that follows is an instruction or relative address which cannot be fully contained before the seventy-fifth column of the card has been reached.

* 0 - This digit is used when a TRA-TCD declarative combination is assembled in any relocatable FEAP program. The five digits that follow the zero constitute a branch address for entrance to a routine.
* $\bar{0}$ - This digit is used in the same manner as 0 above, except the flag denotes an SPS program with absolute addresses.
  1 - This digit indicates that the data to follow after the "length of data" field are instructions.
  2 - This digit indicates that the data following the "length of data" field are constants that are to be relocated.
  $\bar{2}$ - This digit indicates that the data following the "length of data" field are conconstants that are not to be relocated.
  3 - This digit indicates that the data following the "length of data" field are relative addresses to be relocated.
* $\bar{3}$ - This digit indicates that the data following the "length of data" field are relative addresses that are not to be relocated.
* 4 - This digit is used to supply numeric blanks when a relocatable program is loaded. The 2-digit "length of data" field following the indicator specifies how many numeric blanks are desired. Thus a 4$\bar{1}$2 will cause twelve numeric blanks to be inserted into core storage when loading.
* $\bar{4}$ - This digit is used in the same manner as the digit 4 above except that the flag indicates the program is in "absolute" form.
  6 - This digit indicates the end of a relocatable program. In FEAP, the five digits immediately preceding a 6 or $\bar{6}$ (described below) will be the highest address plus one or two to yield an even number for the number of core positions needed for this program. In FEAP or FORTRAN, the card that follows a card containing a 6 or $\bar{6}$ will contain five 9's in columns 1-5, a record mark in column 6 and zeros in columns 7-75.
* $\bar{6}$ This digit indicates the end of an "absolute" program.

Error Messages

If an error occurs during execution of the Loader routine, an error message will be typed and the 1710 will stop to await operator action. A list of error messages, the conditions which cause them, and the corrective action required, follows:

Message — XXXXX LD1 (XXXXX is the sequence number of the last card read in correct sequence).
Cause — Card sequence error.
Action — Correct the order of input cards, starting with the card following XXXXX, and place them in the card reader. Depress the Start key.

Message — LD2
Cause — Card read error.
Action — Reread card by depressing the Check Reset and Start keys on the card read punch.

Message — LD3
Cause — Disk read error.
Action — Depress Start key and retry.

Message — LD4
Cause — Disk read error while reading Loader routine into core storage.
Action — Depress Start key to retry.

DISK UTILITY PROGRAM

The Disk Utility Program is composed of a group of generalized routines designed to assist the user in the off-line operation of the FORTRAN Executive System. By means of these routines, certain frequently required operations, such as loading or unloading portions of disk storage (data or programs), can be performed with a minimum of programming effort by the user.

The routines described in this section are:

1. Write Addresses. This routine writes disk pack sector addresses as specified by the user. Data on the disk pack can be replaced by zeros or left unchanged.
2. Alter Sector. This routine uses the typewriter to change data in a sector of disk storage. In most cases, only the digits to be changed must be typed.
3. Disk-to-Output. This routine unloads disk storage containing data or programs into cards, paper tape, or on the typewriter.
4. Load Programs. This routine loads one or more programs from cards or paper tape to disk storage at either a specified address or an address selected by the load routine itself, and checks for an overlap of previously stored programs.
5. Replace Programs. This routine implements the changes or additions necessary to update a program or disk storage. Input can be in either card or paper tape form.
6. Disk-to-Disk. This routine copies data or programs from one area of disk storage to another.
7. Delete Programs. This routine effectively deletes programs from the system by deleting their associated DIM entries and Equivalence table entries without actually removing the programs themselves.
8. Define Parameters. This routine allows the user to enlarge or shorten the disk storage areas assigned for the work cylinders, DIM, ISIM, Sequential Program table, and the Equivalence table and to specify certain options to be used as system standards. If used, the Defined Parameters routine must be used prior to the loading of any user's programs to disk storage.
9. Define Disk Pack Label. This routine writes the "label sectors" (first and last sectors, cylinder 99) and establishes a Sequential Program table on a disk pack. It can be used to initialize new disk packs.
10. Define FORTRAN Library Subroutine Name. This routine generates an entry in the Equivalence table for FORTRAN subroutines that have multiple entries. Thus, a name can be assigned to all entries in a subroutine. Each routine can be entered and executed by means of control records read by the Disk Utility program. In addition, the routines are used by both FEAP and FORTRAN to output assembled programs into cards or paper tape and to load and replace programs in disk storage.

The Equivalence table, DIM table, and Sequential Program tables are used and modified by the Disk Utility program in the execution of its routines. These tables are updated automatically for each disk storage change when the user adds, deletes, or replaces a program. Entries are created in the tables whenever a new program is loaded to disk storage.

OPERATION

When a DUP Control card (‡‡DUP in card columns 1-5) is recognized by the Supervisor Program, the Disk Utility Program will take control and select the appropriate Disk Utility routine as identified by the next card in sequence, which should be a Disk Utility Program Control card. This card is identified by an asterisk in card column 1. Card columns 2-6 contain a code word to identify the Disk Utility routine desired; such as,

Alter Sector, Load Programs, etc., and the remaining card columns provide additional control information to be used by the Disk Utility routine itself. The user supplies the control information which describes the function he desires. Because the control information for each type of Disk Utility Program Control card is different, the format of each is described separately in the separate routine descriptions. After the execution of a Disk Utility routine is completed, control is returned to the Monitor Control Record Analyzer routine.

A DUP Monitor Control card, as well as a Disk Utility Program Control card, is required each time a Disk Utility routine is to be executed. These cards are stacked with the other input cards to be processed by the System. This stacked input may be in card or paper tape form or it may be entered from the typewriter.

If the code word contained in a Disk Utility Program Control card is not one of the ten legitimate codes (DWRAD, DALTR, DDUMP, DLOAD, DREPL, DELET, DFINE, DCOPY, DLABL, or DFLIB) an error message will be typed and the computer will halt. This message will be comprised of the data from the control card and a constant, ERR CONTROL. When the Start key is depressed, the Disk Utility routine will return control to the Monitor Control Record Analyzer routine which will pass all cards until the next Monitor Control card is reached.

If the control record is entered from the typewriter, the message

ENTER DUP CNTRL REC

is typed and the computer halts. The user may then enter the next Disk Utility Control record from the typewriter and depress the R-S key to continue processing. Records are entered in the alphameric mode.

The Disk Utility program uses the I/O routine of the Supervisor program to perform its I/O functions. Therefore, error messages associated with that routine will be typed if an I/O error occurs. The I/O error messages, as well as operating options for I/O errors, are described under, I/O Error Routine, in the Supervisor Program section.

Whenever a Disk Utility routine is instructed by a control record to write read-only flags with sector addresses, the message

DUP * TURN ON WRITE ADDRESS KEY, START

is typed and the program halts. The operator should turn on the Write Address key (to allow read-only flags to be written) and depress the Start key to continue processing. After the program has been completely loaded, the message

DUP * TURN OFF WRITE ADDRESS KEY, START

is typed and the program again halts to allow the operator to turn off the Write Address key.

Whenever a program of less than 200 sectors is assigned to disk by any Disk Utility routine, it will always be placed in consecutive sectors of one cylinder. However, a program can be assigned by the user, to any available disk storage area as described under Load Programs Routine. Programs as large as 999 sectors long can be processed by the Disk Utility program.

After a program is loaded to disk by any Disk Utility Program routine, the message

DK LOADED AAAAAA IIII DDDDDD
SSS CCCCC EEEEE ≠

is typed to inform the user about the assigned DIM entry.

AAAAAA is the assigned program name, IIII is the assigned DIM entry number and the remainder of the message is the DIM entry itself.

WRITE ADDRESSES ROUTINE

The Write Addresses routine is used to write sector addresses on a disk pack. Addresses may be written with or without read-only flags over the leftmost positions. Data positions of each sector may be changed to zeros or left unchanged.

When the Write Addresses routine is executed, the Write Address key must be turned on. The message:

DUP * TURN ON WRITE ADDRESS KEY, START

is typed to signal the operator to turn the switch on. After the routine has been executed, the message:

DUP * TURN OFF WRITE ADDRESS KEY, START

is typed to signal the operator to turn the switch off.

The format of the control card follows.

Control Card (DWRAD).

Columns	1	Asterisk (*)
	2-6	Code word, DWRAD.
	7-12	Disk sector address where writing is to start (seek address).
	17	Letter P if read-only flags are to be written over addresses; otherwise leave blank.
	18	Letter Z if data positions are to be changed to ZEROS; letter S if data positions are to remain unchanged.
	21-26	Address to be written at sector where writing is to start.
	27-32	Final address to be written.

When the DWRAD Control card is read, control is transferred to the Write Addresses routine and the message

WRITE AND SAVE
__SEEK START STOP
XXXXXX XXXXXX XXXXXX

or the message

WRITE AND ZERO
__SEEK START STOP
XXXXXX XXXXXX XXXXXX

is typed to allow verification of control record data and the computer is halted. (Note that the second form of the message indicates that sector data positions are to be changed to zeros.) Six X's indicate the respective seek, start, and stop addresses. Depressing the Start key causes execution of the routine. The routine seeks the disk sector address specified in columns 7-12. The address specified in columns 21-26 is written in that sector; the address is then incremented by one and written in the next sector. Writing continues in this manner until the incremented address is equal to the final address (columns 27-32) and the final address has been written.

If the program is unable to find the starting address (columns 7-12), or any address that should be on the specified track in disk storage, an error message ER SK XXXXXX will be typed. XXXXXX is the disk address on the last sector examined when no equal comparison could be made with the sector addresses that should be on the track that has been read. In addition, the 20 sector addresses from the selected track will be typed and the program will halt. When the Start key is depressed, control is returned to the Monitor Control Record Analyzer routine to read the next Monitor Control record.

ALTER SECTOR ROUTINE

This routine allows the user to alter the data in any selected sector of disk storage. The sector data to be changed is typed out. All, or selected portions of the sector may then be updated. After the changes have been made, the old and the new data are typed out for visual comparison and verification. If the changes are satisfactory, the new data is stored on the disk pack. As many sectors as desired may be altered each time this routine is used. Control is tranferred to the Alter Sector routine when the control card is read.

Control Card (DALTR).

Columns	1	Asterisk (*)
	2-6	Code word, DALTR.

After the control card is read, the message

SECTOR

is typed and the program halts.

The operator types in the 6-digit address of the sector to be altered and depresses the Release and Start key. If more or less than six digits are typed, the message

SECTOR ADDRESS ILLEGAL,
START TO RE-ENTER *DALTR

is typed and the machine halts. Pushing the Start key will restart all operations on the given sector. The routine reads the sector and types it out in the following format.

1ST. HALF xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx ORIGINAL

2ND. HALF xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx ORIGINAL

Note that the two halves of the sector are identified by the phrases, "1ST HALF" and "2ND HALF," respectively. (Typewriter margins must be at least 70 spaces apart to permit this format.)

Each group of ten characters is assigned a section number by the routine. The first five groups are assigned numbers 01-05; the last five groups are assigned numbers 06-10. After the sector data is typed out, the routine requests the number of the section in which the first change will be made.

The message typed out is shown below:

SECTION

The user now types in one 2-digit section number between 01 and 10. After depressing the Release and Start key, the message

SECTION NUMBER ILLEGAL,
START TO RE-ENTER *DALTR

is typed if the section number is greater than 10. If the section number is correct (between 01 and 10), the selected section is typed out for verification as shown below.

XXXXXXXXXX TYPE CHANGE

The changes can now be entered directly under the typed section. If a particular character does not require changing, an "X" may be typed under that character, or the character itself may be retyped. Although only one section is typed out for any one selection, succeeding sections may be altered by continuing to type changes. Spacing is optional except that the number of characters (including spaces) cannot exceed 100. Spaces (alpha blanks) will not become part of the sector data. For example, assume that section 3 is selected and is typed out as shown below:

3574246798

The operator desires to make some changes in section 3 and section 4. For this example, the typewritten page may look like this:

3574246798 TYPE CHANGE
XXX7625X82 75234XX479

Typing may be terminated as soon as the last digit to be changed is typed; that is, if the fifth digit in section 4 (previous example) is the last change, the last five digits of section 4 do not have to be typed in.

If the user does not type changes but simply depresses the Release and Start key, the message

CORRECTIONS HAVE NOT BEEN ENTERED

is typed and the computer will halt on a read alphameric instruction to allow the user to enter the changes.

If more digits are entered than the sector can contain, the message

TYPE-IN EXCEEDS SECTOR LENGTH, START

is typed. Depressing the Start key allows the operator to begin again and enter a new sector address. Spaces are optional and do not become part of the sector data; however, they are counted toward the maximum allowable number of characters which is 100.

After all changes have been made, the operator depresses the Release and Start key. The routine then types out the original sector data along with the changes that were typed in. The output appears as shown below:

1ST. HALF 1234567890 1234567890 3574246798 8654213212 0987654321 ORIGINAL

1ST. HALF 1234567890 1234567890 3577625782 7523413479 0987654321 CORRECTED

2ND. HALF 7265417623 0176421432 8543217290 5482797654 8243176521 ORIGINAL

2ND. HALF 7265417623 0176421432 8543417290 5482797654 8243176521 CORRECTED

At this point the routine will again type the word

SECTION

If other changes must be made, the operator enters a new (or possibly the same) section number and depresses the Release and Start key. The Section change routine is now repeated. When all the changes prove satisfactory, the operator enters a record mark instead of a section number and depresses the Release and Start key. The routine then writes the updated sector back on the disk pack and types the message

DISK SECTOR DDDDDD CORRECTED

DDDDDD is the sector address that was selected to be changed. The routine then branches to the part of the routine that types the message "SECTOR" to allow the user to choose another sector address and change another sector.

When all desired sectors have been altered, this routine is concluded by typing a record mark instead of a sector address after the word Sector has been typed out. This will cause control to be returned to the Supervisor Program, which will read another Monitor control statement.

Operating Notes

When the routine is ready to accept the new data (after the section number is typed in), it positions the console typewriter in the "alphameric shift" mode. Therefore, typing numerical data requires the operator to manually shift into numerical mode.

Flagged digits 1-9 may be inserted by typing the corresponding alphabetic letters J-R. Flagged zeros, numeric blanks, flagged record marks, and flagged group marks may be entered by using minus (-) key, @ key, W key, and G key, respectively. Alpha blanks do not become a part of the sector data.

DISK-TO-OUTPUT ROUTINE

The Disk-to-Output routine transfers data from selected portions of disk storage to cards, paper tape, or the typewriter. This routine enables the user to preserve original records before they are updated or changed, thus providing an audit trail. The card or paper tape output will be in numerical form and will contain record marks and group marks. Numerical blanks result in blank card columns.

This routine can be used to obtain any of the following items of output as directed by the user.

1. Program or data identified by name.
2. Program or data identified by DIM number.
3. Data between sector limits.
4. DIM table.
5. Equivalence table.
6. Availability list (extracted from Sequential Program table).
7. Sequential Program table.

The routine is executed whenever a DDUMP control card is read by the Disk Utility program or wherever a FORTRAN compilation or FEAP assembly requires punching into cards or paper tape.

The Disk-to-Output routine can be used to transmit any number of disk sectors to cards, paper tape, or typewriter. Transmission will start with the first sector specified in a DIM entry or with a beginning sector specified by the user. Transmission will end when the sector count in the DIM entry reaches zero or when a specified ending sector is found. The output following compilation or assembly is terminated by a "9's" trailer record. The trailer record format is five 9's followed by a record mark, 69 zeros and a sequence number. This record always follows all FEAP and FORTRAN object programs. During execution of the Disk-to-Output routine, resulting from DUP control records, error messages 01, 04, 06, or 20 may be typed (see ERROR DETECTION AND CORRECTION).

The control card used to transfer control to the Disk-to-Output routine is punched in the following format.

Control Card (DDUMP)

Columns		
	1	Asterisk (*).
	2-6	Code word, DDUMP.
	7-12	Alphabetic name of program or data to be punched or typed (same name that appears in Equivalence table).
	13-16	DIM entry number of programs to be punched or typed. (If the letter M is present in column 18, either a name or DIM entry number must be present, but both need not be present.)
	17	Output unit, C = card P = paper tape T = typewriter
	18	Identify output I = Disk Identification Map (DIM) E = Equivalence table A = Availability entries from the sequential program table from the disk module specified by column 19 (typed output only). S = Entire sequential program table from the disk module specified by column 19. M = Program identified by columns 7-12 or 13-16 of this card. L = The sectors between limits as specified by columns 21-26 and 27-32 of this card.

19	Module number (0, 1, 2, or 3) to be used if output options S or A are exercised (see column 18 above).
21-26	Beginning disk storage address of output (lower limit).
27-32	Ending disk storage address of output (upper limit).
76-80	Beginning sequence number (less one) of output deck. If no number is specified, the first card will have a sequence number of 00001.

Output Format

Card

Each 300 positions of disk storage (three sectors) will be punched into four successive cards; 75 columns of disk data followed by a five-column sequence number in each card. When 2 sectors are to be outputted, 3 cards are punched. When 1 sector is outputted, 2 cards are punched. Therefore, all disk data is punched from 1 or 2 sector outputs.

A special trailer card containing 9's in columns 1-5, a record mark in column 6, zeros in columns 7-75, and a sequence number in columns 76-80 will be punched following the last output card. This record is used to terminate loading when the output is reloaded by the Load Programs routine or Replace Programs routine. If the output deck is reloaded by the Load Programs routine or Replace Programs routine, the trailer card must remain behind the deck for control purposes. The 9's trailer record will load into the work cylinders along with the balance of the data; however, the trailer record will not require extra disk storage positions if the data is moved to another disk location. If the output is a program to be reloaded by the loader routine, the entire program must be outputted. The System Output Loader routine requires an entire program, with all of its indicator codes, in order to operate.

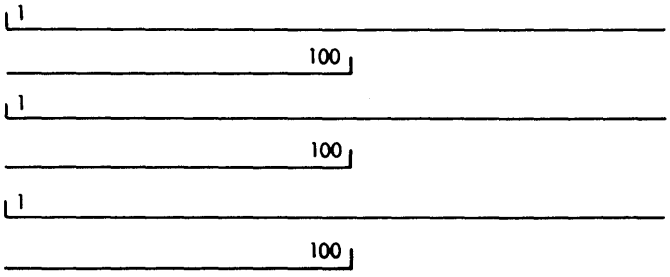
When either the DIM table or Equivalence table is punched out, they will be in a loadable format; i.e., alphameric characters will be in 2-position alphameric coding form.

Paper Tape

This output will be in standard loadable format, i.e., it may be reloaded to disk storage by the Replace Programs routine. The output will be identical to the card format described above, except that the sequence number will not be punched with paper tape records. The last output sector will be followed by a trailer record for control in the event that the output is reloaded into disk storage.

Typewriter

With the exception of the DIM table, Availability list, and Equivalence table, all typewriter output will be in a standard format. Each 100-character sector will be typed on two lines as shown below.



DIM Table

The output of the DIM table is in one of two formats. If Console Switch 3 is turned on at any time the shorter format is typed. The complete DIM entry is typed if Switch 3 is turned off. The user may change the format at will during the time the output is being printed.

Short format	$\bar{X}XXX$
Complete format	$\bar{X}XXX\ X\bar{X}XXXX\bar{X}XX\bar{X}XXXX\bar{X}XXXX+$

Unused (available) DIM entries are not printed.

Availability Lists. Availability lists are typed as follows:

```
AAAAA
BBBBB CCCCC
BBBBB CCCCC
BBBBB CCCCC
```

AAAAA is the disk pack identification number. BBBBB is the starting disk address of an unused area of storage. CCCCC is the ending disk address for the unused area.

These entries, one per line, are extracted from the Sequential Program table by the Disk-to-Output routine.

Equivalence Table. Equivalence table entries are typed in the following format with five entries per line.

NNNNNN $\bar{I}$III NNNNNN $\bar{I}$III NNNNNN $\bar{I}$III NNNNNN $\bar{I}$III NNNNNN $\bar{I}$III

NNNNNN $\bar{I}$III NNNNNN $\bar{I}$III NNNNNN $\bar{I}$III NNNNNN $\bar{I}$III NNNNNN $\bar{I}$III

NNNNNN is the alphameric name. $\bar{I}$III is the DIM entry number. Available FORTRAN library entries will appear as RRRRRR $\bar{9}$999 in place of NNNNNN $\bar{I}$III.

LOAD PROGRAMS ROUTINE

The Load Programs routine is used initially to load FEAP or FORTRAN object programs from the work cylinders or from cards or paper tape into disk storage. Overlap of occupied areas of disk storage is prevented by the routine. Programs cannot be loaded in the work cylinders with this routine. Programs will be loaded into areas of disk storage selected by the routine itself, if the user does not specify a storage area preference. If the routine selects the storage area, it will always store the program on a single cylinder, without overlapping cylinders, unless it is longer than an entire cylinder. If the user selects the storage area, it will be stored in the selected area regardless of cylinder overlap conditions.

This routine provides the following program loading options.

1. A name may be assigned to the program and placed in the Equivalence table.
2. A DIM entry may be assigned to the program.
3. The disk storage location can be specified and permanently assigned (fixed).
4. An entry address (execution address) can be assigned in the DIM entry to the program.
5. Read-only flags can be written in the sector addresses.
6. The disk storage location for the program can be specified by cylinder(s) without causing permanent assignment. Thus, several associated programs can be assigned to the same cylinder or group of cylinders by the user without actually specifying sector addresses.
7. Programs in either core-image or system output formats can be loaded; and programs in system output format can be converted to core image while loading.
8. Process Control programs are loaded as mainline core loads or interrupt core loads.
9. Mainline and interrupt core loads are automatically loaded in core image format and permanently assigned.
10. Core Load maps and status tables are assembled from data selected by DUP and data supplied in the control cards, and are loaded to disk storage with the core load.
11. A library subroutine transfer vector area, and an interrupt subprogram transfer vector area are loaded automatically with mainline core loads.
12. An Interrupt Subprogram Identification Map entry is assembled automatically when interrupt core loads are loaded. This entry is loaded into the ISIM for the system.
13. A library subroutine transfer vector area is loaded (if library subroutines are called) to disk storage automatically as part of the Interrupt Core Load.

It is possible, by exercising option 3, to permanently assign the sectors where the program is to be loaded in disk storage. This capability is provided in this routine only. All process core loads are automatically permanently assigned. When using this option, any programs already in the specified load area, but not permanently assigned, will be moved. The overlapped program is moved to the area which immediately follows the new program. If this in turn would result in additional overlapping of other programs, the process of moving programs continues until available space is found. If any program, in this move, is a permanently assigned program, or contains read-only flags in its sector addresses, no programs are moved and the new program will not be loaded.

A program is considered immovable if it is either permanently assigned by the Load Programs routine or if it contains read-only flags in any of its sector addresses. Permanently assigned programs can be

1. Deleted by the Delete Programs routine.
2. Copied by the Disk-to-Disk routine.
3. Changed by the Alter Sector routine, except sectors containing read-only flags.
4. Dumped into card or paper tape or printed on the typewriter.
5. Read for any purpose with normal read commands.
6. Changed or overlayed (except sectors containing read-only flags).

A permanently assigned program will not be moved in disk storage by the Disk Utility program.

Programs being loaded can be "file protected" by writing read-only flags over the disk addresses of the storage sectors. All loading options are indicated by the control card.

Control Card (DLOAD).

Columns		
	1	Asterisk (*)
	2-6	Code word, DLOAD.
	7-12	Alphabetic name (left-justified) of program to be loaded into disk storage.
	17-20	A DIM entry number to be given the program to be loaded. (This number will not be used by the routine if it is already assigned to another program)
	21-26	Beginning disk sector address in the work cylinders that contains the program to be permanently loaded. The first digit of the sector address selected must be 1,3,5, or 7. (This field is not used if column 58 contains an I or an M.)
	27-32	Ending disk address in the work cylinders that contains the program to be permanently loaded. The first digit of the sector address selected must be 1, 3, 5, or 7. (This field is not used if column 58 contains an I or an M.)
	33-38	Assigned disk storage address of the program to be loaded. (If this address is included, the program will be permanently assigned to the given address.) The first digit of the sector address selected must be 1, 3, 5, or 7. (This field is not used if column 58 contains an I or an M.)
	39-43	Core storage address for a program that is placed in disk storage in core-image format. This address will be placed in the CCCCC portion of the associated DIM entry. (This field is not used if column 58 contains an I or an M.)
	44-48	Entry address (address of the first instruction to be executed) for a program that is being loaded. This address is placed in the EEEEE portion of the associated DIM entry. This address is used for reading programs from disk in core-image format with the I/O routine. (This field is not used if column 58 contains an I or an M.)
	49	Input unit C = card. P = paper tape. D = disk storage work cylinders.
	50	Letter I, if program to be loaded in in Core Image format. S, if program to be loaded is in System Output format. M, if program to be loaded is in System Output format, and it is to be converted to Core Image format before loading to disk storage. (This field is not used if column 58 contains an I or an M.)
	51	Letter P, if read-only flags are to be written over disk addresses of storage sectors; otherwise leave blank. If the read only option is included when a mainline core load is being loaded, the MCL Map (first 3 sectors) will not be protected with read-only flags to permit PSC to change the map (see PSC).
	52-54	Beginning cylinder (three digits XYY, where X is the module, 0, 1, 2, or 3, and YY is the cylinder number 00-99) to define lower limit where program can be loaded. (This field is not used if column 58 contains an I or an M.)
	55-57	Ending cylinder (three digits XYY, where X is the module 0, 1, 2, or 3, and YY is the cylinder number 00-99) to define upper limit where program can be loaded. (Note that both the upper and lower limits will be ignored by the routine if columns 33-38 of this card are punched. This field is not used if column 58 contains an I or an M.)

Control Card (DREPL)

Columns	1	Asterisk (*).
	2-6	Code word, DREPL.
	7-12	Alphabetic name of program to be loaded.
	13-16	The DIM entry number which identifies the program to be loaded if the program is from another assigned disk storage area. (The program to be loaded will be deleted from its original disk storage location.)
	17-20	The DIM entry number which identifies the program to be replaced.

58	Code for process core loads. I is Interrupt Core Load M is Mainline Core Load Interrupt Core Loads will have $\overline{7}$77XX placed in the DIM entry (entry address portion), where $\overline{7}$77 is a code to indicate an interrupt core load, and XX is the indicator number taken from card columns 61-62. Mainline core loads will have a code of $\overline{8}$88 in the entry address portion of the DIM entries.
60	Any non-blank (non-special) character, if program to be loaded is a FORTRAN object program. (This field is not used if column 58 contains an I or an M.)
61-62	Interrupt indicator number. (For Interrupt Core Loads only.)
63-65	Three-digit DIM number of next mainline core load. Column 65 must be flagged if all recorded interrupts are to be processed before the next core load is executed. (For Mainline Core Loads only.)
66-68	Three-digit DIM number of exception mainline core load. (For Mainline Core Loads only.)
69-71	Three-digit DIM number of restart mainline core load. (For Mainline Core Loads only.) If no restart mainline is indicated, $\overline{0}$00 will be substituted and the user's Alert code will be interrogated instead of performing the restart. (See SAC.)
72	Alert code. (For Mainline Core Loads only. See SAC.) 1 Record error and continue. 0 Wait in the interruptible mode. $\overline{1}$ Enter exception core load.
76-80	Beginning sequence number (minus one) of the program being loaded.

REPLACE PROGRAMS ROUTINE

The Replace Programs routine is used to replace programs in disk storage with updated, changed, or new programs. Programs can be loaded to a disk storage area from cards, paper tape, or from another assigned disk storage area. In addition to loading disk stored programs, identified by DIM entries, programs can be loaded from work cylinders.

A program can be given another name in the Equivalence table by reloading the program over its original assigned disk area using a different name. The program can then be called by either name since both names are maintained in the Equivalence table.

With this routine, it is possible to load a program to itself adding read-only flags to the disk addresses. A permanently assigned program in disk storage cannot be replaced by this routine. To replace a permanent program, (1) delete the program with the Delete Programs routine and, (2) load the replacement program with the Load Programs routine.

The format of the control card for this routine follows. All fields are optional with the exception of columns 1-6, 17-20, and 49.

21-26	Beginning disk sector address if the program to be loaded is in the work cylinders. The first digit of the sector address selected must be 1, 3, 5, or 7.
27-32	Ending disk sector address if the program to be loaded is in the work cylinders. The first digit of the sector address selected must be 1, 3, 5, or 7.
39-43	Core storage address for a program that is to be placed in disk storage in core-image format. This address will be placed in the $\bar{C}$CCCC portion of the associated DIM entry.
44-48	Entry address (address of the first instruction to be executed) for a program that is being loaded. This address is placed in the $\bar{E}$EEEE portion of the associated DIM entry.
49	Input unit C = card. P = paper tape. D = disk storage.
50	Letter I, if program to be loaded is in Core Image format. S, if the program to be loaded is in the System Output format. M, if program to be loaded is in System Output format, and it is to be converted to core image format before loading to disk storage.
51	Letter P, if read-only flags are to be written on disk addresses of storage sectors; otherwise leave blank.
60	Any non-blank, alphameric character, if program to be loaded is a FORTRAN or SPS object program which requires subroutines.
76-80	Beginning sequence number (minus one) of the program being loaded.

DISK-TO-DISK ROUTINE

This routine can be used to copy data or programs in disk storage to any available (unoccupied) disk storage area including the work cylinders. A program to be copied should be specified by a DIM entry, an alphabetic name that is in the Equivalence table, or a sector address given by the user. The program cannot be copied into an area which is already identified by a DIM entry number, except the work area (DIM entry 0001). Read-only flags may be written with the disk sector addresses of the copy, except in work cylinders, at the option of the user. When this routine is used, the DIM table and the original program remain unchanged. It is not possible to copy a program over a portion of that same program. It is not possible to copy a program into the work cylinders if that program exceeds the work cylinder limits. Data can be copied from one portion of the work area to another; however, no check will be made for overlapping of data, within the work area. If a program or data to be copied is less than 100 sectors, there is no danger of overlap.

If any read-only flags are encountered in sector addresses within the copy area, an I/O routine error message is indicated. The program will be copied up to the point of the error.

The options offered by this routine are identified in the control card that follows.

Control Card (DCOPY)

Columns	1	Asterisk (*).
	2-6	Code word, DCOPY.
	7-12	Alphabetic name of program to be copied.

Columns	13-16	The DIM entry number which identifies the program to be copied.
	21-26	Beginning sector address of program or data to be copied.
	27-32	Ending sector address of program or data to be copied. (The beginning and ending sectors will always be used if present.)
	33-38	Beginning disk sector address of the new copy. This address must be that of work cylinders or available disk storage. This field must always be punched.
	51	Letter P, if read-only flags are to be written on disk sector addresses at the new location of the program; otherwise leave blank.

The sectors that are to contain the copy must not have read-only flags in the sector address initially or an error will be indicated and copying will be terminated.

After the data is successfully copied, the message

N̄NNNN SECTORS OF DATA
COPIED FROM X̄X̄X̄X̄X̄X̄ TO ȲȲȲȲȲȲ

is typed, where N̄NNNN specifies the number of copied sectors and X̄X̄X̄X̄X̄X̄ and ȲȲȲȲȲȲ are the beginning sector addresses of the From and To areas, respectively. If the copied data is written with read-only flags, an additional message is typed:

AND FILE PROTECTED

To move a program or data from one disk area to another, it should be: (1) copied to the working cylinders from the original area, (2) deleted from the original area, and (3) loaded to the new area from the working cylinders. This can be accomplished by using the Disk-to-Disk, Delete Programs, and Load Programs routines, in that order. Therefore, a *DCOPY control record is used to copy the program into working cylinders; a *DELET control record is used to make the original storage area available by deleting its DIM entry and Equivalence table entry; and a *DLOAD control record is used to load the program to a specified sector address and to generate the new DIM entry and Equivalence table entry.

DELETE PROGRAMS ROUTINE

This routine can be used to delete a program and its associated DIM entry, Sequential Program table entry, Interrupt Subprogram Identification Map entry, and Equivalence table entry or entries (where the program has more than one name) from disk storage. When a program is deleted, read-only flags are removed.

If the program to be deleted is a mainline core load or an interrupt core load, the DELET program performs an additional function in order to maintain the user's Mainline Core Load map and Interrupt Subroutine Identification Map.

If an interrupt core load is deleted, the ISIM entry that specifies that core load is replaced with an ISIM entry that specifies a short program to type the message:

E 20

This message is typed if a new interrupt routine to replace the missing interrupt routine has not been loaded before an attempt is made to service the interrupt. The interrupt indicator is turned off by MIC. The same message and action will result if an interrupt routine was not loaded initially.

If a mainline core load is deleted, the mainline core load map entries of all mainline core loads already on disk are checked. All of the core loads that call the deleted

mainline core load as the next core load have the actual disk address in their Mainline Core Load Map. The DELET program will replace the first position of this disk address for these core loads with a record mark. This code will be used at the time that Program Schedule Control calls the next core load to cause the disk address portion of the Mainline Core Load Map to be filled with the correct disk address from the DIM map. The DIM number in the Mainline Core Load map is not changed by the DELET routine, so the user must retain the same DIM number for the new next-core-load that was used by the deleted next-core-load. If no new program has been loaded to the next core load DIM, the Input/Output routine will indicate an illegal DIM entry error and will transfer to the System Alert Control program to execute the user's error procedure for the core load that calls the missing program.

The format of the control card follows.

Control Card (DELET).

Columns	1	Asterisk(*)
	2-6	Code word, DELET.
	7-12	Alphabetic name of program to be deleted (same name that appears in Equivalence table).
	13-16	DIM entry number of program to be deleted. (Either a name or DIM must be present, but both are not required.)
	18	Any alphameric character to indicate that a core load is to be deleted.

DEFINE PARAMETERS ROUTINE

This routine can be used to alter the assignment of work cylinders, DIM table, Equivalence table, Sequential Program table, or certain system specifications in the System Communications Area. The tables can be enlarged or shortened or relocated as desired. Also, this routine can be used to indicate that more than one disk storage drive is to be used with the 1710.

When the size of the DIM table is changed, the Equivalence table will be moved to immediately follow the DIM table. When redefinition of an area (working cylinders, DIM table, or Sequential Program table) is attempted, the area must be available; i.e., it must not be occupied by programs with assigned DIM entries. If an area is unavailable, it will not be redefined and the message

DUP * ERROR 08

will be typed.

The normal assignment of disk storage for the above mentioned tables is as follows.

Description	Cylinder Assignment
Work Cylinder	8-23
DIM table	24
Equivalence table	25(first eighty sectors)
Sequential Program table	99(second through eighty-first sector)

To make any alterations to these assignments, or to the system specifications, the user must enter a control card containing the new parameters. Only the parameters to be changed need be punched. The parameters from a control card are processed from left to right by the routine. If any parameter is invalid, those parameters to its right will not be processed. The number of modules in the system must be defined on a separate card before the work area is redefined to a different module.

The DIM table and the Sequential Program table cannot be moved. The DIM table

can be shorted or lengthened and, in either case, the Equivalence table will automatically be moved so that it is immediately behind the DIM table. The Sequential Program table cannot exceed 80 sectors in length.

Control Card (DFINE).

Columns		
	1	Asterisk (*)
	2-6	Code word, DFINE.
	7-12	Beginning disk sector address of work cylinders (must be first address of a cylinder).
	14-16	Number of cylinders to be reserved for work cylinders (11 minimum, 99 maximum).
	18	Number of disk storage drives on system (1-4).
	20-22	Number of sectors to be reserved for DIM table (35 minimum, 999 maximum).
	24-26	Number of sectors to be reserved for Equivalence table (9 minimum, 999 maximum).
	28-30	Number of sectors reserved for Sequential Program table (80 sectors maximum). (Note that the same number of sectors is reserved for each disk storage drive on the system as defined in column 18.)
	45-46	Standard length of mantissa (any number 02-08) for FORTRAN programs (08 when the system is delivered). This value must be the same for all process control programs.
	48-49	Standard fixed-point word length (any number 04-10) for FORTRAN programming system (disk sector positions 47-48 of Communications Area, 04 when the system is delivered). This value must be the same for all process control programs.
	51	Source of Input, other than disk input, for FORTRAN subprograms (disk sector position 73 of Communications Area; 5 when the system is delivered). 3 = paper tape 5 = card
	53	Core storage capacity of object machine (disk sector position 76 of Communications Area; 1 when the system is delivered). 1 = 20,000 3 = 40,000 5 = 60,000
	57	FORTRAN Arithmetic and I/O subroutine set identification number (disk sector position 83 of Communications Area; 1 when the system is delivered). 1 = disk storage version 2 = core storage version 3 = disk storage version for machines equipped with the Automatic Floating Point feature. 4 = core storage version for machines equipped with the Automatic Floating Point feature.
	59-63	The core address for the start of mainline core loads. If it is not an even-hundred address already, the address will be reduced to the next lower even-hundred core location. When the system is delivered, this address is 12000.

The number of disk storage drives on the system may be 1, 2, 3, or 4. The Supervisor program and the Disk Utility program will need to know this number in order to utilize all available disk storage. The system will utilize only the first disk storage drive

unless additional drive availability is specified by a DFINE control card. Therefore, it may be necessary for the user to process a DFINE control card immediately after initially loading the Monitor System. When loading programs and assigning addresses, the Monitor System will start with the first available sector on the first available disk drive and proceed sequentially higher to available drives. Also, the user may want to change some of the other parameters of the system before any actual processing is initiated. If any errors are found in any data on a DFINE control card, all data to the left of the data in error will have been processed and data to the right will be ignored. See Error Detection and Correction for a description of possible DFINE errors.

When the routine is used to enlarge or shorten the tables or to change the number of disk storage drives for the system, the *DFINE record should be followed by a ++ PAUS record. After the routine is executed, ++ PAUS record will halt the computer to allow the operator to reinitialize the System.

DEFINE DISK PACK LABEL

This routine can be used to initialize a new disk pack by writing the disk pack identification number in the label sectors (first and last sectors of cylinder 99) and the Sequential Program table in cylinder 99. Also, the five-digit disk address of the Communications Sector (19663) is written in the last five positions of the first sector on cylinder 99.

All disk packs used by the Monitor System must be labeled and must contain a Sequential Program table. The disk pack identification number is written in the first five positions of the first sector in cylinder 99 and a read-only flag is written over the corresponding sector address. The same number is also written in the 31st through 35th positions of positions 1-100 of the last sector of the disk pack. This sector address is changed to 00199 regardless of the addressing scheme used for the remainder of the disk pack.

Note that it is necessary to initialize the disk pack which contains the FORTRAN Executive system because the system pack is not automatically initialized when the system is loaded. Label sectors on a pack which contains the system may be changed by this routine; however, the Sequential Program table will not be reintialized. The FORTRAN Executive System disk pack is identified by $\bar{0}$4800 in the sector address portion of DIM entry 3.

The format of the control card follows.

Control Card (DLABL).

Columns	1	Asterisk (*).
	2-6	Code word, DLABL.
	7-11	Disk pack identification number to be assigned.
	12	Disk drive number (0, 1, 2, or 3) of the disk drive that contains the disk pack to be labeled.

Both a disk pack identification number and disk drive number must be given. If either one is missing, the message

DUP * ERROR 01

is typed and the computer halts without writing label sectors. To correct the error, the operator may enter a corrected control card in the stacked input. Depressing the Start key will return control to the Monitor Control Record Analyzer routine to read the next Monitor Control record in the stacked input.

Only numerical characters may be entered for the disk pack identification number. If this number is all zeros or any position contains a letter or special character, the message

DUP * ERROR 10

is typed and the computer halts without writing a label sector. The restart procedure is the same as that given for ERROR 01.

DEFINE FORTRAN LIBRARY SUBROUTINE NAME

This routine permits the user to assign additional names (synonyms) for the FORTRAN library subroutines or to assign names to user-written library subroutines. Any user-written library subroutine with more than one entry will require this routine in order to place the name of the additional entries in the Equivalence table. These names are added to the Equivalence table within the first 50 entries of the system disk pack.

The control card format follows.

Control Card (DFLIB).

Columns	1	Asterisk (*).
	2-6	Code word, DFLIB.
	7-12	Name of library subroutine, left-justified.
	14-15	DIM number. This number was originally specified by the user when the subroutine was added to the system disk pack. The abbreviated 2-digit DIM numbers for the sixteen standard library subroutines may be found in the FORTRAN Library Subroutines table in the FORTRAN section.

When a name is entered in the Equivalence table, the message

FORTRAN LIB NAME ENTERED NNNNNNĪIII

is typed, where NNNNNN is the name specified in columns 7-12 and ĪIII is the DIM number specified in columns 14-15 (preceded by two zeros).

Both a Name and DIM entry number must be given. If either is omitted, error message 01 will be typed.

Error message 10 will be typed for any of the following conditions.

1. The Name is all numbers, its first character is not alphabetic, or it contains special characters, including nonterminating blanks.
2. Columns 7-15 contain a record mark or group mark in any position or column 13 is not blank.
3. The DIM number is outside the range 10-39 or it contains letters or special characters.

Error message 54 will be typed if no space is available for the Name within the first 50 entries of the Equivalence table. Also, the Name itself will be typed.

Error message 51 will be typed if the name is a duplicate of another name in the Equivalence table. If operator action is required for any of the above messages, refer to Error Detection and Correction.

ERROR DETECTION AND CORRECTION

In addition to the messages described with the individual Disk Utility routines, other numbered error messages may be typed. These messages, described here, may be common to more than one Disk Utility Program as well as FORTRAN output operations that follow compilation. Table 15 indicates, by message number, the error messages that may be generated by each routine. A list of the error messages and their cause, and a list of operator actions for the associated messages follow.

Table 15. Numbered Error Messages Generated by Disk Utility Routines

ROUTINE	ERROR MESSAGE NUMBER																																				
	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	50	51	52	53	54	55	56	57	58	59	60	61	62
Write Addresses (DWRAD)														X		X																				
Alter Sector (DALTR)																																				
Disk-to-Output (DDUMP)	X			X		X								X		X			X																	
Load Programs (DLOAD)	X				X	X		X						X		X	X	X			X		X	X	X	X	X	X	X						X	
Replace Programs (DREPL)	X	X	X	X	X	X	X	X						X		X	X	X			X				X		X	X	X							
Disk-to-Disk (DCOPY)	X			X		X		X	X					X	X	X			X																	
Delete Programs (DELET)	X					X								X					X	X		X														X
Define Parameters (DFINE)	X							X		X	X	X	X				X																			
Define Disk Pack Label (DLABL)	X								X																											
Define FORTRAN Library Subroutine Name (DFLIB)	X								X																X		X									
FORTRAN or SPS Output																	X				X				X		X	X		X	X	X	X	X		

Error Message	Cause
DUP * ERROR 01	Field missing from control card.
DUP * ERROR 02	"TO" DIM entry number specified in DREPL control card is not in use in DIM table.
DUP * ERROR 03	"TO" DIM entry number, specified in a DREPL control card refers to permanently assigned program.
DUP * ERROR 04	"FROM" DIM entry number specified in a DDUMP, DREPL, or DCOPY control card is not in use in the DIM table.
DUP * ERROR 05	Work cylinders illegally specified for program storage by DLOAD or DREPL contol card entry.
DUP * ERROR 06	DIM entry number specified in a DDUMP, DLOAD, DREPL, DCOPY, or DELET control card is out of range of DIM table entry capacity.
DUP * ERROR 07	"FROM" DIM entry number in a DREPL control card refers to an immovable program.
DUP * ERROR 08	Insufficient available storage space at location specified by a DLOAD, DREPL, DCOPY, DFINE control card.
DUP * ERROR 10	Field in DFLIB, DCOPY or DLABL control card contains invalid data.
DUP * ERROR 11	Number of modules specified in DFINE contol card is greater than 4, or less than 1.
DUP * ERROR 12	Beginning disk sector address of work cylinder, in DFINE control card, is not first address in cylinder.
DUP * ERROR 13	Insufficient available storage for specified work cylinder (DFINE control card).
DUP * ERROR 14	Number of sectors specified by DFINE control card for Sequential Program table exceeds 80 sectors.
DUP * ERROR 15	Sector address is non-numerical in a DWRAD, DDUMP, DLOAD, DREPL, DCOPY or DELET control record.
DUP * ERROR 16	Storage loaction specified by a DCOPY control card would cause program storage to overlap work cylinders if allowed.

Error Message	Cause
DUP * ERROR 17	Starting sector address is greater than ending address for DWRAD or DCOPY control card.
DUP * ERROR 18	Sequential Program table is defined as less than required by the present contents of that table. The Sequential Program table is full (DLOAD or DREPL control card or when loading FORTRAN or FEAP output).
DUP * ERROR 19	Core storage address of a program to be placed in disk storage in core image format is less than 02302. A blank address will be treated as 02402. If the program is a Library function (DIM entries between 10 and 130), the message will not be indicated.
DUP * ERROR 20	Name specified by DDUMP, DCOPY or DELET control card is not used in Equivalence table.
DUP * ERROR 21	DIM number specified by DELET control card is not in use.
DUP * ERROR 22	An attempt to load more than 999 sectors has been made.
DUP * ERROR 23	An attempt has been made to delete a core load but no digit was found in column 18.
DUP * ERROR 24	Cylinder limits specified in *DLOAD control card are greater than allowed by System parameters.
DUP * ERROR 50	The interrupt indicator specified in the DLOAD card is not in the Interrupt Indicator Table. The program is loaded to disk but the ISIM is not changed.
DUP * ERROR 51	Name specified is a DLOAD, DREPL, or DFLIB control card or a FORTRAN or FEAP control card has been rejected because a duplication name exists in the Equivalence table.
DUP * ERROR 52	"TO" DIM entry number specified in DLOAD control card is in use in DIM table. (The routine will load the program and assign the DIM entry.)
DUP * ERROR 53 AAAAAA	Name specified in a DLOAD or DREPL control card or a FORTRAN or FEAP control card has been rejected because The Equivalence table (with the exception of the first 50 entries) is full. AAAAAA is the rejected name.
DUP * ERROR 54 AAAAAA	Name specified in a DLOAD, DREPL, or a DFLIB control card or a FORTRAN or FEAP control card has been rejected because the first 50 entries of the Equivalence table are full. AAAAAA is the rejected name.
DUP * ERROR 55 CARD SEQUENCE NNNNN	Sequence error has been found while reading a program to be loaded to disk storage. NNNNN is the sequence number of the card that is out of sequence. Only cards with an eleven punch over the leftmost position of the sequence number (column 76) are sequence checked; therefore, patch cards are excluded from the check.
DUP * ERROR 56	DIM number supplied in FORTRAN or FEAP control card is in use in DIM table, and Name specified in the same card has a different DIM number in the Equivalence table.
DUP * ERROR 57	DIM number supplied in FORTRAN or FEAP control card is in use in DIM table, and Name specified in the same card has no matching name in the Equivalence table.
DUP * ERROR 58	"TO" DIM entry number specified in FORTRAN or FEAP control card refers to a permanently assigned program storage area.
DUP * ERROR 59	DIM entry number specified in a FORTRAN or FEAP control card is out of range of DIM table entry capacity.

DUP * ERROR 60	Insufficient available storage space at location specified by FORTRAN or FEAP control card.
DUP * ERROR 61	DIM table full.
DUP * ERROR 62	Equivalence table DIM entry (specified in DELET control card) is not in use.

Operator Action

Messages 1-24. After the message is typed, the computer halts. The operator may then enter a corrected control record. Depressing the Start key returns control to the Monitor Control Record Analyzer routine to read the next Monitor Control record.
Message 51. No operator action required. The routine continues and loads the program without placing the name in the Equivalence table.
Message 52. The routine continues and assigns a DIM entry and loads the program.
Message 53, 54. No operator action required. The routine continues and loads the program without placing a name in the Equivalence table.
Message 55. After the message is typed, the computer halts. To restart:

1. Remove the cards from the hopper.
2. Depress the Nonprocess Runout key.
3. Remove the last two cards from the stacker.
4. Arrange cards from steps 1 and 3 in correct sequence and place them in the hopper.
5. Depress the Reader Start key. Note that paper tape contains no sequence number, therefore, it can never generate this type of error.

Message 56. No operator action required. The routine continues and loads the program, generating a DIM entry, without placing the name in the Equivalence table.
Message 57. No operator action required. The routine continues and loads the program, generating a DIM entry, and places the name in the Equivalence table.
Messages 58, 59. No operator action required. The routine continues generating a DIM entry and loading the program to disk storage.
Messages 60, 61. After the message types, the computer halts without loading the programs, providing no other output is requested. (If a FORTRAN or FEAP control record is included with the source data to indicate that the compiled or assembled object program is to be punched, the program is outputted without halting the computer.) To correct the error, a DLOAD, DREPL, or DDUMP Monitor Control record can be entered in the stacked input to load the program to a different location from the work cylinders or to output the program. Depressing the Start key returns control to the Monitor Control Record Analyzer routine to read the next Monitor Control record.

FORTRAN AND FEAP OUTPUT

The Disk Utility program contains the output routines for both FORTRAN and FEAP. These routines load object programs to disk storage and punch them out into cards or paper tape.

Following compilation or assembly, the message

DK LOADED AAAAAA IIII DDDDDD
SSS CCCCC EEEEE ‡

is always typed to inform the user about the assigned DIM entry. AAAAAA is the supplied name, IIII is the DIM entry number, and the remainder of the message is the DIM entry.

For programs being loaded into disk storage, the user may select the DIM entry

and/or name. Names and DIM numbers are supplied in FORTRAN (*LDISK) or FEAP (*NAME, *ID NUMBER) control records.

Processing of the Equivalence table and DIM table and the actual loading is dependent upon whether the user supplies a Name and DIM number, Name only, or DIM number only.

Name and number supplied by user.

1. If DIM entry is in use in DIM table.
   a. and Name is already in Equivalence table.
      1) If Name in table references number supplied, replace old program with new program.
      2) If Name in table references another number, type error message 56, and load program with available DIM entry number and no Name.
   b. and Name is not in Equivalence table, type error message 57 and load new program with available DIM entry number and add Name in Equivalence table.
2. If DIM entry is not in use in DIM table.
   a. and Name is already in Equivalence table, type error message 51 and load program with assigned DIM entry number without assigning Name.
   b. and Name is not in Equivalence table, load program and place Name in Equivalence table.

Name only supplied by user.

1. If an identical Name is in the Equivalence table, the program is loaded without the supplied name (error 51 will be indicated).
2. If an identical Name is not in the Equivalence table, the object program is loaded, an available DIM entry is assigned, and the name is added to the Equivalence table.

DIM entry number only supplied by user.

1. Program is loaded. If number was in use, object program replaces old programs. Names referencing the old program are deleted from the Equivalence table.

FORTRAN EXECUTIVE ASSEMBLY PROGRAM

The FORTRAN Executive Assembly Program (FEAP) is a disk-oriented symbolic assembly program designed to simplify the preparation of symbolic programs for use with the FORTRAN Executive System. It is similar to the 1620 SPS II-D program described in the IBM 1620 Monitor I System Reference Manual (Form C26-5739). The differences include additional mnemonics for the 1620 Model 2, index registers and binary capabilities features, and new control records which facilitate coordination of symbolic and FORTRAN programs. Only the operating procedures and the differences between 1620 SPS II-D and 1710 FEAP are described in this section. Appendix B contains a complete listing of all statements and mnemonics available for use with FEAP.

The FEAP program can be called for execution using a Monitor SPS or SPSX control record, however, if it is used on a 1710 Control System, the computer must be operated in the noninterruptible mode.

There are no SPS subroutines (FSQR, FEX, FA, FM, etc.,) available with FEAP. However, the user does have access to the FORTRAN library subroutines and arithmetic and I/O routines when programming for use with FORTRAN. In addition, by using the FEAP language, the user can write subroutines and add them to the FORTRAN library. The procedures to follow, as well as programming notes, are given in this section under Subroutines.

LANGUAGE

All SPS language specifications described in the 1620 Monitor I System Reference Manual are valid for use in the FORTRAN Executive Assembly Program. Table 16 lists the mnemonics for the 1620 Model 2, Index Register feature, and Binary Capabilities feature. In addition, two new declarative operations have been added and are described in the following paragraphs.

DSCI - Define Scan Image

The DSCI statement is used to set up an input area for the reading of High-Level Analog Input points. The statement must have at least two operands; the first designates the core storage address where the scan operation is to begin (if this operand is omitted, the processor will assign the next available core storage location); the second and succeeding operands define the addresses (00-19) of the multiplexer channels. All operands may be actual or symbolic.

For each multiplexer address defined in the statement, the processor generates a 1-digit code plus four zeros. Addresses 00-09 will generate the codes 0-9; addresses 10-19 will generate the codes $\bar{0}$-$\bar{9}$. Thus, the assembled result of a DSCI statement is a constant equal in length to 5N plus ≠, where N is the number of multiplexer addresses defined in the statement.

Example:

```
LABEL DSCI 15000, 01, 02, 19, 04, 17
```

This statement will generate the following constant:

$$1\bar{0}0002\bar{0}0009\bar{0}0004\bar{0}0007\bar{0}0000\neq$$

↑

Location 15000

Table 16. Mnemonics for 1620 Model 2 Index Registers, and Binary Capabilities Features

Mnemonic	Description	Assembled Instruction
BKTY	Backspace (typewriter)	34 xxxxx x01x3
BS	Branch and Select	60 xxxxx xxxxx
BSIA	Branch and Select Indirect Addressing	60 xxxxx xxxx9
BSNI	Branch and Select No Indirect Addressing	60 xxxxx xxxx8
TRNM	Transmit Record, No Record Mark	30 xxxxx xxxxx
BNG	Branch No Group Mark (with 1311 Disk Drive Unit)	55 xxxxx xxxxx
IXTY	Index (typewriter)	34 xxxxx x01x4
BANS	Branch Band A Not Selected	47 xxxxx x31xx
BBNS	Branch Band B Not Selected	47 xxxxx x32xx
BBAS	Branch Band A Selected	46 xxxxx x31xx
BBBS	Branch Band B Selected	46 xxxxx x32xx
BEBS	Branch Either Band Selected	47 xxxxx x30xx
BNBS	Branch Neither Band Selected	46 xxxxx x30xx
BCX	Branch Conditionally, Modify Index Register	63 xxxxx xxxxx
BCXM	Branch Conditionally, Modify Index Register Immediate	64 xxxxx xxxxx
BLX	Branch and Load Index Register	65 xxxxx xxxxx
BLXM	Branch and Load Index Register Immediate	66 xxxxx xxxxx
BSBA	Branch and Select Band A	60 xxxxx xxxx1
BSBB	Branch and Select Band B	60 xxxxx xxxx2
BSNX	Branch and Select No Index Register	60 xxxxx xxxx0
BSX	Branch and Store Index Register	67 xxxxx xxxxx
BX	Branch and Modify Index Register	61 xxxxx xxxxx
BXM	Branch and Modify Index Register Immediate	62 xxxxx xxxxx
MA	Move Address	70 xxxxx xxxxx
ANDF	AND to Field	93 xxxxx xxxxx
BBT	Branch on Bit	90 xxxxx dxxxx
BMK	Branch on Mask	91 xxxxx dxxxx
CPLF	Complement Octal Field	94 xxxxx xxxxx
DTO	Decimal to Octal Conversion	97 xxxxx xxxxx
EORF	Exclusive OR to Field	95 xxxxx xxxxx
ORF	OR to Field	92 xxxxx xxxxx
OTD	Octal to Decimal Conversion	96 xxxxx xxxxx
RBPT	Read Binary Paper Tape	37 xxxxx x33xx
WBPT	Write Binary Paper Tape	39 xxxxx x32xx

d = D operand for BBT and BMK

When a Scan Synchronized or Scan Free instruction is executed, the four zeros for each address are replaced by the converted analog signal.

DFLC - Define Floating Constant

The DFLC statement is used to enter a floating-point constant into the object program. The label is equivalent to the low-order address of the constant.

The DFLC statement contains three operands. The first operand is the mantissa length. This operand may be actual or symbolic, but any symbols used must have been previously defined. If the length is omitted, the mantissa length defined as ff in a MAIN or SUBR statement is used. If a MAIN or SUBR statement was not used, ff is taken from the System Communications Sector. If an attempt is made to specify a length greater than 28, or less than 2, an error message is typed, and the length is set to 8. The maximum mantissa length is 8 for programs used with the FORTRAN Executive System.

The second operand is the floating-point constant. The constant may be in either of two forms:

1. A number consisting of 1 to any number of decimal digits with a decimal point at the beginning, at the end, or between two digits. A preceding plus sign is optional for positive numbers. Zeros to the left of the decimal point are permissible. If the number of digits in the constant exceeds the size of the mantissa, the low order digits of the constant are truncated.

Examples:

```
17.
5.0
-.000003
0.0
```

2. An integral decimal exponent of one or two digits preceded by an E (or E and sign) may follow a floating-point constant. The magnitude thus expressed must be between the limits of 10^{-100} and 10^{99}, or must be zero.

Examples:

```
5.0E3
5.063E-5
0.1E-99
```

Blanks are ignored in either type of constant.

The third operand is the assigned address. The address may be omitted in which case the processor assigns the next available locations.

Example:

```
LABEL DFLC 5, 148.00, ADDR, COMMENTS
```

FORTRAN-Oriented Statements

To facilitate the coordination of Symbolic programs and FORTRAN programs, five new statements are included in the FORTRAN Executive Assembly program.

	Mnemonic	Purpose
	MAIN	Defines a symbolic mainline program
	SUBR	Defines a symbolic subprogram
Invalid unless MAIN or SUBR is used	LIBF	Calls a FORTRAN Library subroutine
	COMN	Defines a work storage area
	CALL	Calls a FORTRAN subprogram

MAIN

A MAIN statement identifies a symbolic program that is to be used as the mainline program in the FORTRAN Executive System. This statement must be the first statement of the FEAP source program; that is, it must immediately follow the SPS Monitor control record in the assembly operation. The format of the statement is:

Mnemonic	Operands
MAIN	N_1, N_2, F, K

N_1 and N_2 are identical to the parameters used in the FORTRAN DEFINE DISK statement described in the FORTRAN section. F and K are the desired floating-point mantissa and fixed-point word lengths, respectively. If F and K are omitted, the system standards are used.

SUBR

An SUBR statement identifies a symbolic subprogram that is to be used with the FORTRAN Executive System. This statement must be the first statement following the SPS Monitor control record. The format of the statement is:

Label	Mnemonic	Operands
NAME	SUBR	F, K

F and K are the desired floating-point mantissa length and fixed-point word length. This statement overrides any name assigned by an *NAME aaaaaa control record.

LIBF

An LIBF statement is used wherever a FORTRAN Library subroutine is desired in the FEAP program. The format of the statement is:

Mnemonic	Operands
LIBF	Function name, Argument

The Function Name is the symbolic name of a FORTRAN Library subroutine. The subroutines and their names are:

Subroutine	Name
Logarithm (Natural)	LOGF
Exponential	EXPF
Cosine	COSF
Sine	SINF
Arctangent	ATANF
Square Root	SQRTF
Absolute	ABSF
Disk Write	RECORD
Disk Read	FETCH
Disk Read or Write List	DKLIST
Disk Read or Write Array	DKARAY
Disk Read or Write Complete	DKEND
Subscript 1 dimension	ENTSC1
Subscript 2 dimension	ENTSC2
Subscript 3 dimension	ENTSC3

The absolute value function is a library subroutine that is not relocatable. It is loaded into core along with the arithmetic subroutines.

COMN

This statement is used to define the size of a block of core storage to be used as a work area. The format of the statement is:

Label	Mnemonic	Operands
WORK	COMN	A, B, C

A, B, and C are the three dimensions of an array. Any operands that are omitted are assumed by the program to be one.

The amount of storage that is reserved equals A*B*C* (f + 2) where f equals the mantissa length defined in the MAIN or SUBR statement. If the first letter of the label is I, J, K, L, M, or N, denoting fixed-point mode, the amount of storage reserved is equal to A*B*C* (k) where k is the fixed-point word length defined in the MAIN or SUBR statement.

The location of the reserved area is at the high end of core storage. For example, if 20,000 positions of core storage are available for execution of the program, then the area would extend from location 19999 downward. If more than one area is defined (via subsequent COMN statements), the second, third, etc., areas will begin in the first location following the preceding area.

The label of the statement refers to the units position of the first element in the array, that is, the element with the lowest core address.

Notice that the use of this statement corresponds to the use of a DIMENSION and a COMMON statement in FORTRAN. Thus, the same work area can be defined and used in both a FORTRAN mainline program and a related FEAP subprogram.

CALL

The CALL statement allows FORTRAN subprograms to be called from a FEAP program. The format of the statement is:

	Mnemonic	Operands
	CALL	NAME, A, B, ... N

NAME is the name of the subprogram being called, and A, B, ... N are arguments. The arguments must be in symbolic form, and must be defined in the FEAP program. A typical call might be:

```
        CALL    CO, N250
        .
        .
N250    DC      K, 250
```

CO is the name of a Contact Operate subprogram and 250 is the number of a contact to be closed. Note that the 250 is written symbolically as N250 and later defined as a constant. K is the fixed-point word length and is also specified in the MAIN or SUBR statement.

The instructions generated in-line are BTM * +6, *+11, 67 and DSA P_1 the first time a particular subprogram is called. The BTM instruction generated in-line on all successive calls to the same subprogram will have a P address which specifies the low-order (units) position of the P address of the BTM that was used in the first call. At the end of the FEAP program output, the name of each subprogram called and the address of the low-order position of the P address of the first BTM generated for the subprogram are outputted. At the time that the FORTRAN loader gets the subprogram and loads it, the actual address of the subprogram called is placed in the P operand of the first BTM instruction generated for each different subprogram called. The names and address are then overloaded by the next routine to be loaded.

CALLING EXECUTIVE CONTROL PROGRAMS

All the Executive control programs that can be called using the FORTRAN language can be called also from FEAP-written FORTRAN routines. Four of these require an additional parameter not required when writing in FORTRAN, three require different entry addresses when called from interrupt and from noninterrupt routines, and two others can be called in the same way that FORTRAN requires, but the function desired can be more efficiently performed using a single FEAP instruction. In addition, the user can utilize the SIOC output printer to print DMES formatted messages but must observe a slightly different convention with respect to SIOC formatting.

The entry points for PSC, ADC, AO, SIOC, and DIAG are in the Executive Transfer Vector (ETV). The symbols for these entry points must be the addresses in the short form (subroutine sets 1 and 3) ETV. These addresses are automatically corrected at load time if the long form is used so the user need not concern himself. ADC, AO, and SIOC must be entered at different addresses when called from an interrupt and from a mainline program. Different names should be used for the interrupt calls (i. e., ADCI, AOI, and SIOCI). These addresses must be preloaded into the System Symbol Table by the user unless they are defined with each assembly. In addition, the first four of these control programs requires an additional identification parameter that is automatically generated when writing in FORTRAN language. This parameter must be placed before any other parameters for the calls that require it. A sample of each call of this type will illustrate the way the call is written and also show the content of the parameter for each call.

```
                CALL PSC, IDPSC, IDOPT, J
                .
                .
IDPSC           DC    K, 3,, EXTRA PARAM CONTENTS
IDOPT           DC    K, 2
J               DC    K, 678

                CALL ADC, IDADC, JNO, MFRST, I
                .
                .
IDADC           DC    K, 2,, EXTRA PARAM CONTENTS
JNO             DC    K, 10
MFRST           DC    K, 110
I               DSB   K, 10

                CALL AO, IDAO, IFRO1, LSPP, ISLEW
                .
                .
IDAO            DC    K, 1,, EXTRA PARAM CONTENTS
IFRO1           DC    K, 1
LSPP            DC    K, 68
ISLEW           DC    K, 5

                CALL SIOC, IDSIOC, IDLIT
                .
                .
IDSIOC          DC    K, 0,, EXTRA PARAM CONTENTS
IDLIT           DC    K, 2171
```

In each example, the contents of the extra parameter required to identify the routine has been specified with the remark EXTRA PARAM CONTENTS.

NOTE: A mask operation must be performed before calling the SIOC, AO, or ADC programs from a recorded interrupt routine. The return to MIC from a FEAP-written interrupt routine is done using an indirect branch to 03252.

The user may write CALL MASK or CALL UNMASK in FEAP. The output will not be the same as with FORTRAN, but it will perform the function desired. The output will be BTM * +6, *+11, 67, and the name of the routine called (MASK or UNMASK) will be outputted with the address of the *+6 operand. The FORTRAN loader will load the address of the MASK (or UNMASK) subprogram into the P operand. These subprograms exist permanently in core and consist of these instructions:

```
          MASK                          UNMASK

          DC  5, 0                      DC  5, 0
MKSR11    MK                  UMKSR1    UMK
          BB2                           BB2
```

To accomplish the function, the user can use an FEAP instruction (MK or UMK) in the same way it is used in-line when CALL MASK or CALL UNMASK instructions are given in FORTRAN.

The SIOC output printer call includes (when using FORTRAN) a statement number. This must be replaced with a label in FEAP language. The user may use the format feature of the SIOC program or format his own messages at assembly time using the DMES statement. In the SIOC formatting feature the label must address the position of storage that is nine core storage positions before the format data to be used by the SIOC formatting routine.

In addition, a five-digit DC and a three-digit DC must be included preceding the user's format data to specify the FORTRAN format routine address and twice the number of format characters. These DC's should be as shown below:

```
        DC    5, 4676
        DC    3, XXX
```

where XXX is two times the number of characters in the DAC that follows. The DAC is used to specify the format using the same specifications as are used in the body of the H-type FORTRAN FORMAT statement described in the SIOC section. Only the portion of the FORTRAN FORMAT statement after the first H should be in the DAC.

Example:

```
                CALL  SIOC, IDSIOC, IDPRT, NUMF, NO, I
                .
                .
IDSIOC          DC    K, 0
IDPRT           DC    K, 072
NO              DC    K, 1
I               DC    K, 1234
                DC    5, 04676
                DC    3, 012
LAB             DAC   6, I6, (R)
NUMF            DS    , LAB -10
```

This statement causes the contents of I to be printed with two spaces preceding, i.e., bb1234. The K in the above example is the value of the fixed-point word length.

NOTE: If no variables are to be printed with a given call to SIOC, the DC that is labeled NO must contain a constant of zero and the list must be empty (i.e., NO must be the last parameter).

When messages are not to be formatted by SIOC (i.e., are written using DMES), the user must include a three-digit sector count of the message to be printed immediately before the DMES statement. This sector count must be calculated by the user. Messages will automatically begin in the numeric mode and the user must change mode if alphabetic output is to be printed. Since several DMES statements may be used in succession to compose one message, user formatted messages may be as long as desired provided the message will fit in the message area on disk (100 sectors initially).

Example:

```
                CALL  SIOCI, IDSIOC, IDMES, LAB
                .
                .
IDSIOC          DC    K, 0
IDMES           DC    K, 0172
SECNT           DC    3, 001
LAB             DMES  ,, (A) (R) (M) I AM AN ALERT MESSAGE (M) (B) (R) (E)
```

The call sequence above will cause the message

I AM AN ALERT MESSAGE

to be printed on the output printer. IDSIOC specifies that this is an SIOC call. (This parameter is used only in FEAP-written SIOC calls.) IDMES specifies that the output printer is to be selected, that the message is already formatted, and that device code 72 is used for the output printer desired. These three specifications are indicated in the constant with the label IDMES using the 0, the 1, and the 72, respectively. SECNT is 001 to specify that the message requires one sector. This constant is not used with alert messages and could have been omitted since alert messages are never stored on disk before printing. Since the message was formatted at assembly time, the printing of this alert message will start immediately.

Since Contact Sense (CALL CS), Contact Operate (CALL CO), time reading (CALL CLOCK) and 1711 manual entry reading (CALL MEOP) are used like normal FORTRAN calls, the use of these statements has been covered under the description of subprograms called in FEAP programs (see CALL Statement in FEAP section).

USING FORTRAN SUBROUTINES

To call a FORTRAN Arithmetic or I/O subroutine, the user executes a BTM instruction referencing the symbolic address of the subroutine. The symbolic names of the available subroutines are given in Table 5.

The second operand (Q field) should be the address of the A or I referred to in Table 5. FAC, the FORTRAN accumulator, may be accessed by the subroutines . TOFAC and . FMFAC; or the user may access FAC directly by writing TF (fixed-point) or TFL (floating-point) and using the symbol . FAC as the P or Q operand. If the latter method is used, the testing for overflow and underflow is the user's responsibility. The names of FORTRAN subroutines and the symbol . FAC are stored in the System Symbol Table; therefore, any FEAP assembly which refers to these symbols must be preceded by a *SYSTEM SYMBOL TABLE control record. Although the names of the FORTRAN I/O routines and subscripting routines are included for completeness, it is not recommended that they be called from a FEAP-assembled program unless the user has first acquired a thorough understanding of them.

Adding Subroutines to the FORTRAN Library

The user may write library subroutines in FEAP language and have them placed in the FORTRAN library. The subroutine must be assembled using FEAP and may be loaded to disk storage at assembly time, or at a later time using the Disk Utility Program.

When the subroutine is loaded, a special DIM entry number which is reserved for FORTRAN library subroutines must be used. This number is specified using the FEAP control statement, ID NUMBER, or is punched in the DLOAD control card. Only 30 DIM entries are reserved, and, of these, the first 15 are in use when the system is delivered. The first 10 must not be replaced because they are required for correct operation of the FORTRAN system, but the user is free to replace any of the others if he desires.

If a subroutine has multiple entries, the first DIM entry will define the subroutine, but a DIM entry is required for each entry point, and no subroutine may have more than 9 entry points.

All entry points for a subroutine must be indicated in the following manner at the beginning of the source statements.

```
SUB         DSA  ENTRY1, ENTRY2, . . . , ENTRY9
            DORG SUB-4
ENTRY1
```

where ENTRY1 is the name of the first entry point, and ENTRY2 is the name of the second, etc.

The user must provide a 5-position area immediately preceding each entry point. This space will be used to contain the address of the parameter for the subroutine when the subroutine is entered. Only one parameter will be transferred to the subroutine. If more than one in-line parameter is required, the user must utilize a subprogram.

The symbolic name for each entry point must be specified in a DSA statement at the beginning of the source program when it is assembled (even if only one entry point is desired). Also, the operand of the DEND statement must specify the number of entries to the subroutine.

Working Areas

In writing the subroutine, the programmer may first move the argument into one of the working areas such as FAC, BETA, or SAVE. In arithmetic subroutines, the exponent of a floating point result is usually stored in SAVE before being moved to FAC. A careful study of the arithmetic subroutines may reveal that the relocatable subroutines to be added can share the normalization, sign determination, overflow, underflow, and error typeout sections. The value calculated by the subroutine must be left in FAC. Even if no value is calculated, it is advisable to place a constant in FAC.

When programming a subroutine with variable length floating-point numbers, it may be necessary to use certain addresses and constants available in the arithmetic and input/output subroutines. A reference to the listings of these subroutines will yield the necessary information. The symbolic names of the constants are listed in the System Symbol Table.

Most of the symbolic addresses normally required have been placed permanently in the Systems Symbol Table. As the mode of operation (fixed- or floating-point) is determined by the argument of the subroutine, the FORTRAN processor does not distinguish fixed-point subroutines from floating-point subroutines. It is up to the user to have a thorough knowledge of the added subroutines and to use them correctly.

Loading the Library Subroutine

The Disk Utility Program can be utilized to add a subroutine to the FORTRAN library or it can be added at assembly time.

An example of the control records required for adding a subroutine directly to the FORTRAN library at assembly time follows:

```
+ +JOB
+ +SPS
*LIBR
*NAME HCOS
*ASSEMBLE RELOCATABLE
*STORE RELOADABLE
*ID NUMBER 0026
```

} Monitor and SPS control records

```
SUB     DSA   HCOS,    HSIN
        DORG  SUB-4
        .
        .
        DC    5, 0
HCOS    OP
        .
        DC    5, 0
HSIN    OP
        .
        .
        DEND  2
```

The User-written library function SPS instructions

The Disk Utility Program can be utilized to load the subroutine to the library; in which case, the NAME, STORE, RELOADABLE and ID NUMBER statements can be omitted, but an OUTPUT CARD or OUTPUT PAPER TAPE statement would have to be included.

The DLOAD (or DREPL) Control record must contain the information as shown in the following example.

```
* DLOAD     HCOS        0026        0101200002        C     I
Col         7           17          39   44           49    50
```

Columns		
	1-6	Code word *DLOAD.
	7-12	Alpha name of program to be used in FORTRAN arithmetic statements.
	17-20	DIM entry number.
	39-43	The length of the subroutine. This number must be even.
	44-48	The number of entry points.
	49	Input unit (P for paper tape).
	50	Core image format.

Other options, such as read-only protection, are available if they are desired (see DLOAD Control Record in the Disk Utility Program section of this manual).

Additional Entries and Synonyms

A DFLIB Control record must be entered if the subroutine contains more than one entry point or if one entry point is to be called by more than one name. The format for the DFLIB Control record to add the second entry for the preceding examples is as follows:

Columns		
	1-6	*DFLIB
	7-12	HSINbb
	13	Not used
	14-15	27
	16-80	Not used

where HSIN is the user-assigned name, left-justified, of the subroutine being added, and 27 is the next consecutive DIM entry number, after the DIM entry number used for the original entry. The DIM entry number must be between 20 and 39 and must correspond with the sequential position of the entry as it is written in the operands of the DSA statements in the source program. As delivered, the system already makes use of DIM entry numbers 10 through 24 for FORTRAN library subroutines. However, the last 6 may be

removed, if desired. If no subroutines are removed from the FORTRAN library set, the available DIM entry numbers for additional library subroutines are 0025 through 0039.

CONSTRUCTION OF I/O REQUESTS AND FORMAT STATEMENT OUTPUT

To simulate the FORTRAN input statement, ACCEPT N, A_1, A_2, ... A_m, the user will include the following sequence of linkage instructions to be assembled.

```
BTM   .RATY, ADRFMT (N)
BTM   .SWC, LOC (A1)
.
.
.
BTM   .SWC, LOC (AM)
BTM   .COMPLT, 0 0
```

The first instruction branches to the RATY I/O subroutine macro and transmits to that routine the base address (ADRFMT) of format specifications that would result from a FORTRAN FORMAT statement. The symbols . RACD or . RAPT may be used in place of . RATY if card or paper tape input is desired. The symbols . WATY, . WACD, or . WAPT should be used to output data on the typewriter, card punch, or paper tape punch, respectively. Base address is the address of the last character prior to format specifications.

The BTM instructions transfer control and transmit the data address for each element of the list. The final instruction informs the I/O subroutine that it is the last element in the list. (Flag on low-order of LOC if field is fixed.) The low-order position of LOC is flagged if data is in fixed form.

MATRICES

Where it is desired to read in or write out an entire matrix, the output from the source statement will have the following form:

```
BTM   .RACD     ,ADRFMT(N)
TFM   .PAR      ,NOELS
BTM   .MATRX    ,LOC(1)
BTM   .COMPLT   ,00
```

The first instruction branches to the proper I/O macro and transmits to that macro the base address of the format specifications that resulted from the FORMAT N statement. The base address is the address of the last character prior to format specifications.

The following two instructions transmit the total number of elements in matrix (NOELS) to a parameter area (PAR) and the address of the location of the first element of the matrix LOC(1) is transmitted to the matrix routine while branching to the matrix routine (. MATRX). These two instructions may appear by themselves in a list or in front of or back of any single BTM, SWC, LOC(A) statement of the previous example.

When these linkage instructions are included in a FEAP assembly, a SYSTEM SYMBOL TABLE statement must also be included.

CONSTRUCTION OF DISK I/O REQUESTS

To simulate the linkage for recording or fetching a list of variables, the user may write the following instructions:

```
LIBF RECORD, IRECNO
LIBF DKLIST, LOC (A1)
.
.
.
LIBF DKLIST, LOC (AM)
LIBF DKEND, 0
```

The variables at LOC (A1) through LOC (AM) will be written in the records starting with the record specified in IRECNO. The same instructions, employing FETCH instead of RECORD, will read the variables in the records specified in IRECNO into LOC (A1) through LOC (AM). If the variable specified by LOC is fixed point, LOC should be negative (-LOC). To simulate the linkage for recording of arrays the user may write the following instructions:

```
LIBF RECORD, IRECNO
TFM .PAR, NOELSA
LIBF DKARAY, LOC (A1)
TFM .PAR, NOELSB
LIBF DKARAY, LOC (B1)
.
.
.
LIBF DKEND, 0
```

The array A and the array B with the number of variables specified in NOELSA and NOELSB, respectively, will be written onto the disk starting at the record specified in IRECNO. A set of instructions using FETCH instead of RECORD would read the array A and the array B. As many sets of TFM, .PAR, NOELS, and LIBF DKARAY, LOC (A1) instructions as there are arrays to be read or written should be included before the LIBF DKEND, 0 instruction. When using these routines with FEAP, N1 and N2 must be defined as described in the FORTRAN section. When the program that calls FETCH or RECORD is assembled, a SYSTEM SYMBOL TABLE statement must also be included.

Simulation of FETCH and RECORD in FEAP

The user may wish to write or read data using his own routine, and to use this data with the FORTRAN disk I/O routines. If this is done several considerations are important.

1. All listed variables must occupy the same number of positions on the disk. This number equals the larger of k or (f + 2). Shorter variables must be right-justified in this space.
2. No more than N1 variables may be placed on a single disk record.
3. Records must be one or two sectors in length as in FORTRAN.
4. All arrays must be written without wrong-length record check, they must include a group mark immediately after the array when written, and, if the array starts in an odd core position, a record mark must precede the array. The array must be written from the position that contains the record mark.
5. All arrays must be read with wrong-length record check. The group mark will come into core after the array and the wrong-length record check indicator will be turned on.
6. Using IORT for reading arrays requires the following instructions to facilitate correct error check operation:
   a) Place the instruction TFM DIO+35, YTURN before the IORT PUT macro.
   b) Include the following instructions somewhere in the program:

```
YTURN    BI     *+12, 3700
         BNI    ERRET, 1900
         TFM    INOUT, MYER
         B7     IOERR
MYER     BI     *+12, 3700
         BI     CHECK-12, 1900
         B7     EEXIT
DIO      DS     ,00816
ERRET    DS     ,00602
INOUT    DS     ,02098
IOERR    DS     ,00624
CHECK    DS     ,01262
EEXIT    DS     ,01630
```

7. All arrays read into even core locations must be moved one position to the left if the first character read is a record mark. A transmit record instruction should be used.
8. All arrays to be read into odd core locations must instead be read into the next higher core address, which is even. If a record mark is found in the first character position, indicating that the array was originally written from an odd position, the array must be moved two positions down in storage using a transmit record instruction. If no record mark is found in the first position, the array must be moved one position down in storage.
9. The user must save and restore any required data overlayed by arrays that are read in this way.
10. The user must determine that no incorrect writes to disk are performed.

Input/Output Format Control

The user should attempt to utilize the data about constructing I/O FORTRAN statements to create FEAP written linkage and FORMAT statements only if he completely understands them. Since calls to the system IORT for input/output are more straight forward and faster to execute, the user would be wise to utilize this alternative to simulating FORTRAN I/O.

SLASH (. SLASH) (5) a five-digit absolute address corresponding to . SLASH in the I/O subroutine.

HOLLERITH (. H TYPE) (5,3,2,2) a five-digit absolute address corresponding to . HTYPE in the I/O subroutine. This is followed by a three-digit field and W two-digit fields. (W is the number of Hollerith characters.) The three-digit field will contain the width times 2 followed by W two-digit fields containing the alphameric representation of the Hollerith data.

REPETITION (. REP) (5,5,2) a five-digit absolute address corresponding to . REP in the I/O subroutine, followed by a five-digit relocatable address specifying the format location to which I/O program must return to secure next element of format specification. These two five-digit fields will be followed by a two-digit field specifying the total number of times the preceding format field is to be used.

(. REP3) A five-digit absolute address corresponding to . REP3 in the I/O subroutines and performing the same function as . REP but for several specifications that are to be repeated. Several references to REP may be "nested" within a reference to . REP3.

X TYPE (. XTYPE) (5,3) a five-digit absolute address corresponding to . XTYPE in the I/O subroutine. This is followed by a three-digit field, containing twice the number of alphabetic blanks desired.

E TYPE (. ETYPE) (5,3,2) a five-digit absolute address corresponding to . ETYPE in the I/O subroutine. This is followed by a three-digit field w w and a two-digit field d d, where w w is the width specification and d d is the decimal specification.

$$w\,w \geq d\,d+6$$

F TYPE (. FTYPE) (5,3,2) a five-digit absolute address corresponding to . FTYPE in the I/O subroutine. This is followed by a three-digit and a two-digit field w w and d d, where w w is the width specification and d d is the decimal specification.

$$w\,w \geq d\,d+2$$

I TYPE (. ITYPE) (5,3) a five-digit absolute address corresponding to . ITYPE in the I/O subroutine. This is followed by a three-digit field w w which is the width specification times 2.

$w \leq K-2$	(ARG NEG)
$w \leq K-1$	(ARG POSITIVE)

A TYPE (. ATYPE) (5,3) a five-digit absolute address corresponding to . ATYPE in the I/O subroutine. The following field will contain three digits for the width specification times 2.

$w \leq (F/2)$	For variable length (character will be set to 00)
$w \leq (K/2)$	For fixed length

REDO (. REDO) (5,5) a five-digit absolute address corresponding to . REDO in the I/O subroutine. The following field will contain the first address (less five) in the format field that is to be repeated. This will be used specifically to terminate each FORMAT statement.

SUBSCRIPTING ROUTINE LINKAGE

The subscripting routines utilize the following instructions to link from the calling program:

```
BTM   ENTSCX, *+12, 67
DSA   BASE, D4, D1, I, D2, J, D3, K, 0=
```

ENTSCX is one of the three subscripting routine entries. If the Q address is flagged (negative) the array is a formal parameter. That is, the address of the array is actually the address of an address of the array. If D1 is negative, the call is within an input/output call linkage. If only one or two dimensions are handled, the last four or two addresses, respectively, may be omitted. The zero and record mark must be after the last address included.

D1, D2, D3 and D4 are calculated values based upon the following:

GIVEN:	$(C_1 * V_1 + B_1,\ C_2 * V_2 + B_2,\ C_3 * V_3 + B_3)$
Where	C_1, B_1, C_2, B_2, C_3 and B_3 are fixed point constants
and	V_1, V_2 and V_3 are fixed point variables
and	I and J are the number of rows and columns, respectively;

THEN: $D1 = C_1$
$D2 = I*C_2$
$D3 = I*J*C_3$
and $D4 = B_1 - 1 + I*(B_2 - 1) + I*J*(B_3 - 1)$

OPERATING PROCEDURES

Assembly

To assemble a FEAP source program, proceed as follows: load the source program preceded by the applicable Monitor Control record (SPS or SPSX) and the desired FEAP control records. The source program must end with a DEND statement. Following the DEND statement is either the first card of the next "job," or data if the program is to be executed immediately after assembly.

FEAP Control Records

FEAP control records must be provided to control the assembly of the programs. These records may be in card, paper tape, or typewritten form, and are inserted in the stacked input behind the Monitor Control record (SPS or SPSX). The control records are typed out when they are encountered in the stacked input. The format of a FEAP control record in terms of cards is as follows:

Columns	1	*
	2-75	Control Statement

Only one control statement may be entered in each control card. The statements must be written exactly as given, except for blanks which are permitted anywhere in the control statement. Control statements may be followed by remarks. Any statement, other than those listed below (e.g., an identification statement) will be typed, but will have no effect on the assembly. The processor will indicate an identification statement by typing (ID) to the right of such a statement.

Intervening blanks between the letters of a control statement do not invalidate the statement.

TWO PASS MODE. This control statement causes the object program to be produced by entering the source program twice (two passes). Two passes are required when the space allotted for work storage is too small to contain the intermediate output from the assembly. (The space required for a on-pass operation is approximately one sector per source statement.) If one-pass assembly is attempted with too large a source program, an error message is typed out.

OBJECT CORE n. This statement specifies the core storage capacity (20,000, 40,000, or 60,000) of the object machine (machine on which the object program will run). If the storage capacity of the assembly machine (machine on which the object program will be assembled) and the object machine are the same, this statement is not needed. The n digit of the statement is one of the coded digits 2, 4, or 6 which represents 20,000, 40,000 and 60,000, respectively.

The FEAP program can be called for operation on a 1710 Control System only when the 1710 is operating in the noninterruptible mode.

ERROR STOP. This statement instructs the processor to stop whenever a source statement containing an error is encountered. When this occurs, an error message will be typed (see Error Messages). The operator can then enter a corrected source statement

and continue assembly (see On-Line Error Correction). If an Error Stop control statement is omitted, the processor will not stop for erroneous source statements; however, an error message will still be typed.

ASSEMBLE RELOCATABLE. This statement causes the processor to assemble a relocatable object program in System Output format. If this statement is omitted, the processor will produce an "absolute," nonrelocatable program.

BEGIN CARD INPUT, BEGIN PAPER TAPE INPUT, BEGIN TYPEWRITER INPUT. These three statements cause the loading program to begin reading input from the newly designated unit. These statements can be used as "last" control statements when the source program is to be entered from a different input medium than were the control statements.

TYPE SYMBOL TABLE. This statement causes the symbol table to be typed after all source statements have been read (see Typeout of Symbol Table).

PUNCH SYMBOL TABLE. This statement causes the symbol table to be punched into cards after all source statements have been read. These cards may be listed 80-80 on an IBM 407 to obtain a printed symbol table.

LIST TYPEWRITER. This statement causes a program listing (containing both source and object data) to be typed as the program is being assembled.

LIST CARD. This statement causes the program to be punched into cards which may be used to make a program listing. If desired, both the LIST TYPEWRITER and LIST CARD statements may be used in one assembly.

OUTPUT CARD. This statement causes the object program to be punched into cards in a reloadable format (see System Output Format in the Supervisor section). This output will occur after any symbol table and listing outputs.

OUTPUT PAPER TAPE. This statement causes the object program to be punched into paper tape in a reloadable format (see System Output Format in the Supervisor section). Either an OUTPUT CARD or an OUTPUT PAPER TAPE statement may be used for an assembly, but not both.

STORE CORE IMAGE. This statement causes an assembled program to be permanently stored on the disk in a format that is identical to the format of an executable program in core storage. This statement may not be used in an assembly that also contains an ASSEMBLE RELOCATABLE statement.

STORE RELOADABLE. This statement causes the assembled program to be stored on the disk in a reloadable format. This format is identical to that described under System Output Format. If neither Store Core Image not Store Reloadable is specified, the assembled program will not be permanently stored on the disk. However, the program will remain in the work cylinders until destroyed by another job.

SYSTEM SYMBOL TABLE. This statement allows the source program to use symbols stored in the System Symbol table without defining them in the source program itself. There is a provision in FEAP for defining user symbols in the System Symbol table (see FEAP Modification Program).

ID NUMBER dddd. This statement assigns a four-digit DIM entry number (dddd) to a program being assembled. Exactly four digits, including leading zeros, must be entered.

NAME aaaaaa. This statement can be used to assign a Name in the Equivalence table for an assembled program which is to be stored in disk storage. aaaaaa is a 6-character alphameric name. At least one of these characters must be alphabetic.

LIBR. This statement must be entered when assembling a user-written subroutine that is to be added to the library subroutines.

PUNCH RESEQUENCED SOURCE DECK. This statement causes the processor to punch a new source deck in sequence by page and line number. The page and line field will contain a five-digit number starting with 00010 and will increase by ten for each successive card, e.g., 00010, 00020, etc. The resequenced deck is punched while the old source cards are being read. The output appears in the punch stacker ahead of any other punched output. When operating in two-pass mode, the resequenced source deck should be used for the second pass. Corrections to source statements made from the typewriter will not appear in a resequenced source deck.

PUNCH SELF-LOADING CARDS. This record causes the object program with a loader to be punched in cards for use with another system. This deck may be loaded by inserting in the card reader and pressing the LOAD key.

PUNCH SELF-LOADING TAPE. This statement causes the object program with a loader to be punched in paper tape for use with another system. The tape may be loaded from the tape reader by pressing the INSERT key and entering 36 00000 00300 R/S where R/S is the Release and Start key.

Only one of the above two control records may be present in an assembly. If either one is used, the control records in the following list will be ignored and execution of the assembled program will be inhibited.

```
*ASSEMBLE RELOCATABLE
*STORE CORE IMAGE
*STORE RELOADABLE
*OUTPUT CARD
*OUTPUT PAPER TAPE
```

FEAP ERROR MESSAGES

The error message codes that might be typed out on the typewriter during an assembly are listed in Table 17.

Error messages take the following general form:

```
PPPPP          ALABEL + CCCC ERn
```

where PPPPP is the page and the line number of the statement in error, ALABEL is the last label used, and CCCC is the number of statements from that label to the statement in error.

When ER5 (see Table 17) is typed out, the erroneous symbol is also typed.

Table 18 shows what action the processor will take for each error if no Error Stop control statement has been included in the assembly.

Error Correction

One-Pass Mode. If the operator wishes to correct source program errors during the assembly process, he must use the Error Stop control statement. When an error occurs, the appropriate error message is typed out along with one of the following instructions to the operator:

Table 17. FEAP Error Codes

ERROR CODE	CAUSE OF ERROR
ER1	The capacity of the machine on which the object program is to be executed has been exceeded. The processor does not take subroutines into account when determining this error.
ER2	Invalid label or record mark is in a label field.
ER3	Invalid OP code or record mark is in an OP code field.
ER4	A label is defined more than once.
ER5	1. A symbolic address contains more than six characters. 2. An actual address contains more than five digits. 3. An undefined symbolic address is used in an operand. 4. A HEAD character ($) is improperly specified.
ER6	A DSA or DSCI statement has more than ten operands.
ER7	A DSB statement has the second operand missing.
ER8	1. A DC, DSC, or DAC has a specified length greater than 50. 2. A DVLC has a length greater than 50. 3. A DMES has a length greater than 100. 4. A DNB has a length greater than 99. 5. DFLC has a length greater than 28 or less than 2.
ER9	A DC, DSC, DAC, DVLC, or DMES statement has no constant specified.
ER10	1. A DC or DSC statement has a specified length which is less than the number of digits in the constant itself. 2. A DAC statement has a specified length which is less than or greater than the number of digits in the constant itself.
ER11	An invalid character is used as a HEAD character in a HEAD statement.
ER12	A HEAD operand contains more than one character.
ER13	A DMES statement contains an invalid starting mode character.
ER14	1. A DMES statement contains a control character which is incorrectly specified. 2. A DMES statement has an invalid format, i.e., stray parenthesis, etc.
ER15	1. A DMES statement contains an alpha character in a numerical field. 2. Format error in constant of DFLC.
ER16	A DMES statement contains an invalid mode change.
ER17	1. A relocatable assembly contains either a relocation error (see Rules of Relocatability) or, 2. A DORG with an absolute operand.
ER18	1. A symbolic name used in a CALL LINK or CALL LOAD statement is not in the Equivalence table. 2. Name in LIBF not in the Equivalence table or refer to an Invalid subroutine number.
ER19	The storage area allotted for the symbol table has been exceeded.
ER20	Intermediate output has exceeded disk storage work area (program requires two passes).
ER21	Object output has exceeded disk storage work area.
ER22	Improper "select" operand is in a CALL statement; i.e., neither LINK, LOAD, nor EXIT is specified.

Table 18. Disposition of FEAP Errors When No Error Stop Statement is Used

ERROR CODE	DISPOSITION
ER1	No disposition.
ER2	The label is ignored.
ER3	A NOP is assembled.
ER4	The second definition of the label is ignored; the first definition of the label is used in the assembly.
ER5	The operand is assembled as an absolute 00000.
ER6	The first ten operands are assembled; any remaining operands are ignored.
ER7	The number of elements is set to 1.
ER8	1. Length is set to 50. 2. Length is set to 50. 3. Length is set to 100. 4. Length is set to 99. 5. Length is set to 8.
ER9	A field of zeros is generated, equal to the size of the length operand for the DC, DSC, DAC, or DVLC constant. In the case of a DMES, an end of message (++) is assembled and the address counter is increased by 100.
ER10	For a DC or DSC, the length of the constant is used as the length operand; for a DAC, the specified length is used, and the programmer-assigned address, if present, is ignored.
ER11	The HEAD character is set to blank.
ER12	The first character of the operand is used as the HEAD character.
ER13	The starting mode is assembled as the alphabetic mode.
ER14	An end of message (++) is inserted into the constant.
ER15	1. An end of message (++) is inserted into the constant. 2. Floating 0.0 is assembled.
ER16	A $\bar{0}$ is placed in the next available location following the mode change.
ER17	1. The operand is assembled as an absolute 00000. 2. The DORG is ignored.
ER18	1. A DIM number of 00000 is assembled. 2. A P address of 00000 is assembled.
ER19	Processing continues but no more labels are stored. After completion of the intermediate phase, processing stops, the following message is typed, and control returns to the Supervisor Program. DISK AREA TOO SMALL. ASSEMBLY DELETED
ER20	Processing continues, but no more intermediate data is sent to disk storage. After completion of the intermediate phase, processing stops, the following message is typed, and control returns to the Supervisor Program. DISK AREA TOO SMALL. ASSEMBLY DELETED
ER21	Processing stops immediately and control is returned to the Supervisor Program.
ER22	The statement is ignored.

NOTE: Assembly and outputting continues in all cases except ER 19, 20, and 21.

RE-ENTER STATEMENT or
RE-ENTER OPERANDS

At this point, the processor returns the typewriter carriage and types the full erroneous source statement. If only the operands are to be re-entered, the processor will then re-type the source statement up to the operand field. The processor at this point requires that the operator enter either an entire corrected source statement or a corrected operand field. The operator should use the previously typed original statement as a guide to the positions of the Page, Line, Lable, Op code, and Operand fields.

Two-Pass Mode. The error correction procedure in two-pass mode is identical with that of the one-pass mode, with one exception. During the second pass, the processor might type an error message containing "ER XX." This message always refers to a statement corrected during the first pass. The operator should scan the typewritten record of the corrections made during the first pass to find the one identical in page and line number, label, and increment. When the processor types RE-ENTER STMT and returns the carriage, the operator must re-enter the entire corrected statement, exactly duplicating the statement entered during the first pass.

Post Assembly Phase

After assembly is completed and listings, if desired, have been outputted, the following messages are typed:

END OF ASSEMBLY
XXXXX CORE POSITIONS REQUIRED
XXXXX STATEMENTS PROCESSED

In the above typeouts, XXXXX is a 5-digit number.

Symbol Table Output

If either of the statements Type Symbol Table or Punch Symbol Table are present in an assembly, the symbol table will be typed or punched during assembly. This output, if punched, will precede the list deck in the punch stacker.

All 6-character labels are listed first in reverse alphameric order, i.e., 9 to 0, Z to A. All other labels follow in normal alphameric order with their head characters. In the case of an assembly in which the number of symbols exceeds 235 (some symbols will then have to be stored on disk), the listing is broken into two or more blocks, each of which is sorted as described above.

The format of the symbol table output is as follows:

Typewriter. The typed output lists all labels and their numerical equivalences, five to a line. The format is as shown.

Label	Equivalence
LLLLLL	AAAAA(—)

Here LLLLLL refers to a 6-character label or a 5 or fewer character label with a head character. AAAAA refers to the numerical equivalence of the symbol. The minus sign, if present, denotes a negative quantity. If the program is being "assembled relocatable," the minus sign is replaced by an R to denote a relocatable quantity.

Card. The card output format of the symbol table is as follows:

Columns	1-13	1st label plus equivalence
	17-29	2nd label plus equivalence
	33-45	3rd label plus equivalence
	49-61	4th label plus equivalence
	65-77	5th label plus equivalence

Formats of Typewriter Listing and Punched Deck

If desired, the operator can obtain a typewriter listing and/or a punched list deck of an assembled program. The formats of each type of output are described here.

Typewriter. A typewriter listing consists of a source statement together with its associated assembled machine language instruction.

Card. A card list deck usually consists of one card for each source statement. The format is as follows:

Columns	1-5	Page and line number.
	6	Blank.
	7-12	Label as on source card.
	13	Blank.
	14-17	Op mnemonic as on source card.
	18	Blank.
	19-78	Operand fields as on source card. If the fields extend beyond column 59, the object information (normally found in columns 61-80 of first card) is placed on a subsequent card or cards.
	61-65	Actual address of assembled instruction or constant.
	66	Blank.

NOTE: The data in columns 67-80 is peculiar to the type of statement assembled.

Imperative Statements.

Columns	67-68	Op code in machine language.
	69	Blank.
	70-74	P operand in machine language.
	75	Blank.
	76-80	Q operand in machine language.

Non-imperative Statements.

Columns	67-71	Length of assembled data.
	72	Blank.
	73-80	If these columns are punched, they will contain actual assembled data.

Error Messages After Assembly

The following error messages are applicable after assembly.

EXCEEDED SPECIFIED CAPACITY BY XXXXX

The above message indicates that the object program would exceed the available core storage if the program were to be executed. The available core storage is determined by the user's Object Core control card.

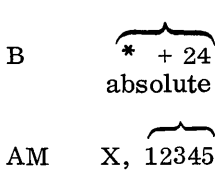
MORE THAN 5 CYLINDERS OF
RELOADABLE OUTPUT SSW4
ON TO DUMP OUTPUT OFF
TO CONTINUE, NO OUTPUT

The above message is typed when the reloadable object output would occupy more than 999 sectors on disk storage (approximately 5 cylinders). This situation is an error because programs greater than 999 sectors cannot be specified in the Disk Identification Map.

After the message is typed out, the computer halts. At this time the user can either turn Program Switch 4 on and depress START to have the program outputted on a pre-chosen output unit, or turn Program Switch 4 off and depress START to continue, in which case the program is not outputted. In either case the program is not stored on disk storage.

Rules of Relocatability

When a program is relocated, as specified by an Assemble Relocatable statement, certain addresses within the program are adjusted relative to the relocation (starting) address. Only relocatable quantities are adjusted. Absolute quantities are not adjusted. Examples of both relocatable and absolute quantities follow:



The processor recognizes relocatable and absolute quantities by applying the following rules:

1. An integer (e.g., 1, 12345, etc.) is an absolute value.
2. A processor-assigned address, which is associated with a label (i.e., the address of an instruction or constant with an associated label), is a relocatable quantity. An asterisk address (*) is also relocatable.
3. A symbol defined as equal to some quantity has the same relocation property as the associated quantity. An example follows:

```
SYMBOL  DS    ,QUAN
```

4. The product of two absolute quantitites is an absolute quantity.
5. The sum or difference of two absolute quantities is an absolute quantity.
6. The sum or difference of a relocatable quantity and an absolute quantity is a relocatable quantity.
7. The difference between two relocatable quantities is an absolute quantity.

The processor will recognize any of the following situations as "relocation errors."

1. The sum of two relocatable quantities.
2. The product of a relocatable quantity and any other quantity.
3. An operand below the relocatable address of 00000. For example, RELOC - 10000, where RELOC is a relocatable quantity of less than 10000.

NOTE: The exact negative of a valid relocatable quantity is a valid relocatable quantity.

Although the quantity defined by an operand may be either positive or negative, a symbol may be equivalent to a positive quantity only. If a symbol is defined equal to a negative quantity, any reference to that symbol by the assembler will produce the absolute value of the quantity.

FEAP MODIFICATION PROGRAM

This program allows the user to modify the SPS II-D assembler by: (1) Adding or deleting operation codes from the System Op code table, and (2) Adding or deleting symbols from the System Symbol table. The FEAP Modification program is loaded into disk storage as part of the Monitor System. It is identified in the Equivalence table by the name "SPSLIB".

An XEQ Monitor Control record with the assigned name SPSLIB punched in columns 7-12 is used to call the modification program for execution. To specify the type of modification desired, the user places modification control records following the XEQ record. These records and any other input data to the Modification program must be entered from the same input device that was used to enter the XEQ record.

Modification program control records, in terms of cards, use the same format as that used for SPS control records. The five modification control statements must be written exactly as given (DEFINE OP CODE, DELETE OP CODE, DEFINE SYSTEM SYMBOL TABLE, LIST OP CODE, ENDLIB). Only one statement may be included in a control record. These statements are typed when they are read. A description of the five control statements follows.

DEFINE OP CODE. This statement causes user-assigned Op (operation) codes, specified in Op code definitions cards, to be added to the System Op code table. The Op code definition card(s) must follow the control record in the stacked input. The format of the Op code definition card follows:

Columns	12-15	New mnemonic Op code (left-justified)
	16-75	A 3-digit code which determines the instruction generated by the Op code. (The code may be preceded by a minus sign.)

The allowable 3-digit codes that may be entered in columns 16-75 are shown in Table 19.

Table 19. FEAP Modification Program Codes

CODE	ASSEMBLED DATA	USE
XY0	XY PPPPP QQQQQ	Any instruction
-XY2*	8X PPPPP YQQQQ	SIOC instructions
-XY4	4X PPPPP Q00QY	Mask and Unmask instructions
XY2	3X PPPPP Q07QY	Disk instructions
XY3	3X PPPPP Q0YQQ	I/O instructions
XY4	34 PPPPP Q0XQY	Control instructions
XY6	46 PPPPP QXYQQ	Branch Indicator instructions
XY7	47 PPPPP QXYQQ	Branch No Indicator instructions

*If the first character of the Op code contained in columns 12-15 is the letter "R," position O_1 of the assembled instruction will be flagged.

The digits X and Y may be any number 0-9. A separate Op code definition card should be entered for each Op code that is to be defined. If an attempt is made to define an Op code that is already present in the Op code table, the message

ALREADY DEFINED

will be typed and the new Op code will be ignored. If space is unavailable in the Op code table for a new Op code, the message

NO ROOM IN TABLE

will be typed and the new Op code will be ignored.

DELETE OP CODE. This statement causes Op codes, specified in Op definition cards, to be deleted from the System Op code table. The Op code definition card(s), which must follow the control record in the stacked input, specifies in columns 12-15 the code to be deleted; columns 16-75 may be blank. Only one Op code may be specified per card. If an attempt is made to delete an Op code that is not in the Op code table, the message

NOT IN TABLE

will be typed and no change will be made to the table.

DEFINE SYSTEM SYMBOL TABLE. This statement is used to modify the System Symbol table. The System Symbol table consists of certain symbols that were defined when the Monitor System was assembled plus any symbols the user adds by means of the Define System Symbol Table statement. Appendix F contains a listing of the System Symbol table when the system is delivered. Any symbol that is in the System Symbol table may be used in any assembly without defining the symbol within the program being assembled. When used, the Define System Symbol statement first causes all user-defined symbols to be deleted from the table. Then all symbols which follow the Define System Symbol statement are added to the System Symbol table. Symbols to be added are defined in the Symbol Definition record. The format of this record in terms of cards is as follows:

Columns	6-11	Symbol to be defined (left-justified).
	16-75	An operand, symbolic or actual, but not asterisk. (If a symbolic operand is used, it must have been previously defined in the System Symbol table.)

If a symbolic operand, contained in the operand field (columns 16-75) of a Symbol Definition card, cannot be matched with a previously defined symbol in the System Symbol table, the message

UNDEFINED SYMBOL XXXXX

is typed out, where XXXXX is the undefined symbolic operand; no change is made to the System Symbol table. Up to 66 user-defined symbols may be added to the System Symbol table. Any attempt to insert more symbols causes an error message to be typed and control to be returned to the Supervisor program, thus terminating the add-to-symbol-table function. Symbols that have less than six characters will be defined with a blank "heading character" in the System Symbol table. Symbols defined as positive quantities will be treated as positive-absolute quantities in both absolute and relocatable assemblies. Negative quantities will be treated as negative-absolute quantities in an absolute assembly and positive-relocatable quantities in a relocatable assembly.

LIST OP CODE. This statement causes the processor to type a listing of the Op code table. All Op codes are listed in tabular form with their associated 3-digit codes.

ENDLIB. This statement causes control to be returned to the Supervisor program. In the stacked input, it must follow other control statements which utilize the modification program.

This program is used initially to load the Monitor System (DUP, FORTRAN, FEAP, SUPERVISOR) from cards or paper tape into disk storage. Cards contain 75 columns of data followed by a five-position sequence number. Sequence numbers are not present with tape data.

The system to be loaded, in card or paper tape form, is comprised of several blocks of data, each with a unique deck number, a Heading Control record, and a 9's trailer record. With this arrangement it is possible to load each new block of data to a different area of disk storage as specified by the Heading Control records. For card input, the cards within a data block must be consecutively numbered in ascending sequential order.

The combined input data, i.e., all data blocks, must be preceded by the Loader program itself. This program is contained in approximately forty cards. If the sequence of the first four cards is inadvertently altered, the program may not operate correctly. The loader program is deck number 00, columns 30-31. All input cards, with the exception of the first four cards of deck 00, are sequence-checked by the loader.

NOTE: The on-line Input/Output routine is a self-loading program that does not require the use of the loader program. It may be loaded any time before it is required for execution.

CARD FORMATS

Heading Control

Columns	1	Asterisk(*).
	2-7	Code word, LDCNTR.
	9-14	Name of data block (program, table, etc.) to follow.
	16-21	Address of first sector to be loaded.
	23-28	Address of last sector to be loaded.
	30-32	Deck number. This number combined with the two positions 79-80, constitutes the sequence number. Blanks are interpreted as zeros. Therefore, the number 55 and blanks in columns 30-32 are interpreted as sequence number 55000. The first card of the data block must then begin with the sequence number 55001 in columns 76-80.

Data

Columns	1-75	Data to be loaded to disk storage.
	76-80	Sequence number.

Trailer

Columns	1-5	$\bar{9}9999$
	6	∓
	7-8	00

OPERATING PROCEDURES

Switches

The Parity, I/O, and O'FLOW check switches should be in the PROGRAM position for either card or tape loading. For card or tape input, the program will halt after each trailer card if Program Switch 1 is off. If the switch is on, all data blocks are loaded without stopping the computer. Therefore, the user can stack input, if desired.

Paper Tape Loading

1. Ready the paper tape reader with the Loader tape reel.
2. Enter 36 00000 00300 from the typewriter.
3. Depress the Release and Start keys.
4. Ready the tape reader with the Data tape reel.
5. Depress the 1620 Start key.
6. Return to step 4 to load successive Data tapes.

NOTE: When loading with Switch 1 on, the Loader will continue to read more data after each data group has been loaded. Therefore, several such data input groups may be present on one input reel.

Card Loading

1. Ready the card reader with deck number 00, Loader Program. The remaining decks may be stacked behind deck 00 as explained under Switches.
2. Depress the 1622 Load key.

Loader Messages

For Both Paper Tape and Card Loaders:

Message/Cause/Operation Action

AAAAAA LOADED FROM F$\bar{F}$FFFF TO L$\bar{L}$LLLL, where AAAAAA is the name of a data block, F$\bar{F}$FFFF is the address of the first sector loaded, and L$\bar{L}$LLLL is the address of the last sector loaded. This message will type following each successful deck loading. If Program Switch 1 is on, it is an indication to the operator to load the next deck.

DISK RD WR ERROR, START TO RETRY. This message will type if a disk write error occurs that cannot be corrected by one automatic retry. Depressing the Start key will cause the write operation to be retried twice. If the error is not corrected by the retries, the message will again be typed.

RDER. This message will type if a paper tape or card reading error occurs. To correct the error, ready the reader with the corrected record and depress the Start key. (An error card will be located next to the last card in the stacker when a halt occurs for a card reading error.)

CONTROL STATEMENT INVALID, RE-ENTER. This message will type if any of the following conditions are encountered in Heading Control record data.

1. A misspelled code word.
2. A record mark in column 6.
3. First sector to be loaded is greater than last sector to be loaded. The user must supply a corrected control record and depress the 1620 Start key.

For Card Loader Only:

Message/Cause/Operator Action

SEQ. This message will type and the program will halt if any of the cards in the loader program, with the exception of the first four cards, is out of sequence. To resume loading, (1) restore the cards to their correct sequence and place them in the card hopper, (2) depress the Start keys on both the card reader and 1620 console.

$\overline{N}$NNNN CARD SEQ ERROR, CORRECT AND START, where $\overline{N}$NNNN is the sequence number of the first data card out of consecutive ascending sequence. After the message is typed, the program will halt. To restart the computer, (1) restore the sequence of data cards, starting with the card in error, (2) place the resequenced cards in the card read hopper, (3) depress the Start keys on both the card reader and 1620 console.

NO TRAILER REC. CORRECT, RE-LOAD COMPLETE DECK WITH CNTR REC, AND BR TO 7404. This message will type and the program will halt if a 9's trailer record is missing following any data block. To restart the computer, the user should (1) restack the cards in the card reader so as to restart card reading with the Header card of the data block which had the missing trailer record, (2) depress the Reset and Insert keys, (3) enter 49 07404 from the typewriter, (4) depress the Release and Start keys, (5) depress the card reader Start key.

TRAILER CARD SEQ ERROR, CORRECT AND START. This message will be typed if the sequence number on the trailer card is incorrect. The procedure for restarting is the same as for any other card sequence error.

EXECUTIVE ASSEMBLY AND LOADING PROCEDURES

All Executive Control programs are unassembled when received to allow the user to tailor each program to fit his particular application. This is accomplished by assigning addresses or constants to labels that are used throughout the individual Executive Control programs and user's programs. A Define System Symbol Table operation (See FEAP) is used for this purpose.

The procedure is to punch the Define System Symbol Table statements in cards or tape and enter them using the FEAP SPSLIB program. This section describes the purpose of each label that must be defined before assembling the programs. Only those programs required by the user should be assembled. All assemblies must be done using FEAP and the FORTRAN Executive pack to be used while controlling the process. The advantage of this method is that labels placed in the FEAP System Symbol table need not be defined in each program that uses them (see FEAP-System Symbol Table section).

NOTE: Before assembling any programs with this system, the user should be sure that he has defined his system parameters (see DUP DFINE). The mantissa length (ff) are fixed word length (KK), the FORTRAN subroutine set, the starting address for the mainline core loads, and the work area portion of the disk should be set as desired for the particular system. If the size of the Interrupt Save area, the Mainline Save area or the SIOC Message area (in disk) is to be changed, the changes must be made at this time (see the section on Changing or Deleting Disk Area). These parameters, along with all other such parameters, should be defined before assembling the Executive Control programs described in this section.

System Symbols Defined by the User

A complete list of all symbols to be defined by the user is shown below. The cards needed to enter the selected operands are included with the system. After punching the specifications as described in the succeeding sections into these cards, the user must enter all of them using the SPSLIB program described in the FEAP section of this manual. All labels must be entered before any Executive programs are assembled.

CTVT	DCTVT	AOTBAD	NUMDEV	DEV70
SHORTF	MANLIN	AOSLEW	PRPRES	through
AOIND	TIME	AOTRIM	NUMPRT	DEV89
SIIND	OVOPT1	AOPRDK	NUMBUF	
AIIND	OVOPT2	NOSPP	LOFCAR	
DEBUG	OVOPT3	AOTBDK	NUMMES	
PIN01	K	N1		
through		N2		
PIN15				
NPIN				

MASTER INTERRUPT CONTROL

The following labels are used with the Master Interrupt Control (MIC) program.

CTVT	XXXXX	Core address of the Executive Transfer Vector area. This address once established, must not be changed without reassembling all Executive Control programs. The choices for XXXXX are 07600 if FORTRAN Subroutine set 1 or 3 is used, 09100 if set 4 is used, or 09700 if set 2 is used.

The next five labels are for the purpose of decreasing the core storage requirements of the MIC program when certain items are not used. A zero in the units position of the operand causes appropriate instructions to be eliminated.

SHORTF	0	FORTRAN Subroutine sets 1 or 3 are not used.
	1	FORTRAN Subroutine sets 1 or 3 are used.
AOIND	00000	The Analog Output Setup interrupt is not used.
	00001	The Analog Output Setup interrupt is used.
SIIND	00000	The SIOC program is not used.
	00001	The SIOC program is used.
AIIND	00000	The ADC program not used.
	00001	The ADC program is used.
DEBUG	00000	The trace feature is not to be incorporated in the object output of this program.
	00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Option under Diagnostic Aids.)

Assigning Process Interrupts and Timed Interrupts

Since the number of Process interrupts and Timed interrupts used is entirely up to the user, the MIC program must be provided with a label-versus-indicator assignment for each of these interrupts that are used. For the purpose of this assignment, the Timed interrupts and Operator Entry interrupt are treated like Process interrupts. The information supplied here is used to establish the user's Interrupt Indicator table.

The labels to be used for this assignment are PIN01 through PIN15. The 15 labels must be defined even if all 15 interrupts are not used. For example, if only six interrupts are used, the labels PIN01 through PIN06 would be assigned to the six interrupts, in any order, while the remaining PINXX labels would be defined with zeros in the address operand.

The label assignment below indicates that eight Process interrupts and the two Timed interrupts are being used.

Label	Indicator Number	
PIN01	48	Process interrupt indicator 48
PIN02	49	Process interrupt indicator 49
PIN03	50	Process interrupt indicator 50
PIN04	51	Process interrupt indicator 51
PIN05	52	Process interrupt indicator 52
PIN06	53	Process interrupt indicator 53
PIN07	54	Process interrupt indicator 54
PIN08	55	Process interrupt indicator 55
PIN09	43	One Minute interrupt indicator 43
PIN10	44	One Hour interrupt indicator 44
PIN11	00	Unused indicator
PIN12	00	Unused indicator
PIN13	00	Unused indicator
PIN14	00	Unused indicator
PIN15	00	The order or priority in which the interrupts above will be interrogated is specified by the number in the last two positions of the label. If the user wishes interrupt 54 to be tested before any other process interrupt, he must assign the label PIN01 to that indicator. The Interrupt Subroutine

NOTE: The label-versus-indicator assignment above does not have to be in any particular order.

Identification Map (ISIM) will be set up in the sequence specified by the numbers in the labels described here. In addition, one ISIM entry for the user's SIOC manual entry and sense switch enter interrupt and one ISIM entry for th CE interrupt will be set up before any process interrupts in this table. The CE interrupt service routine is supplie with the system. The user will write his own SIOC enter interrupt routine. Both of these ISIM entries must be defined even if the user does not utilize the SIOC feature. (NOTE: the order of the user's ISIM entries will be the same as the order of his Interrupt Indicator Table (IIT). The SIOC enter interrupt indicator has been arbitrarily selected as 45 in the IIT).

The following label is used to define the number of combined Process interrupts and Tim interrupts that are to be used. If SIOC is used in the system, an additional interrupt must be included in the count. In the label assignment above, the number would be 10 if SIOC was not on the system or 11 if it was.

NPIN	000XX	
AOPRDK	0	AO program on disk
	1	AO program not on disk

PROGRAM SCHEDULE CONTROL

The following System Symbols are used with the PSC program

CTVT	XXXXX	Core address of the Executive Transfer Vector (ETV).
MANLIN	XXXXX	Core address of the user's mainline program. This can be any even hundreds core address after the last Executive Control Program to be used (i. e. , after the Skeleton Executive Program).
9999Z	XXXXX	This label is not assigned by the user. It is the core addr of the start of the in-core portion of PSC. This is placed in the permanent system symbol table when the PSC portio of the Skeleton Executive is loaded. It is used by the disk portion of PSC, which must be assembled after the Skeleto Executive is loaded since the symbol is not equated to a number until the in-core portion of PSC is loaded. NOTE: it is necessary to assemble the entire PSC program if a change is made to the in-core portion.
DCTVT	XX	The core address of the user's Executive Transfer Vector divided by one hundred. The user enters this symbol. It will be 76, 97 or 91 according to which subroutine set is used. Sets 1 and 3 require 76, set 2 requires 97, and set 4 requires 91.
TIME	XXYY	Desired time interval between logging operations. If no logging is desired. This field must contain zeros. XX is hours and YY is tenths and hundredths of hours.
SIIND	00000	SIOC not used.
	00001	SIOC used.
DEBUG	00000	The trace feature is not to be incorporated with the PSC.
	00001	The trace feature is desired.

ANALOG-DIGITAL CONTROL

The following System Symbol statements are used with Analog-Digital Control (ADC) program.

DEBUG	00000	The trace feature is not to be incorporated in the object output of this program.
	00001	The trace feature is to be incorporated in the object output of this program. (For a description of the trace feature, see Trace Option under Diagnostic Aids.)
OVOPT1	0	Do not use ADC overload option 1
	1	Use ADC overload option 1
OVOPT2	0	Do not use ADC overload option 2
	1	Use ADC overload option 2
OVOPT3	0	Do not use ADC overload option 3
	1	Use ADC overload option 3

These options are described in the Analog-Digital section.

K	XX	Where XX is the length of the fixed-point variable to be used with the ADC program. The value of XX must be from 04 to 10 as defined for the FORTRAN programs.

ANALOG OUTPUT CONTROL

The following System Symbol Table statements are used with Analog Output Control (AOC) program.

CTVT	XXXXX	Core address of the ETV area.
AOTBAD	XXXXX	Core storage location (first digit) of the Analog Output table if the Analog Output table is maintained in core.
AOSLEW	XXX	Three-digit terminal address for selection of a slew operation.
AOTRIM	XXX	Three-digit terminal address for selection of a trim operation.
NOSPP	XX	Number of set point positioner.
AOTBDK	1	AO record stored on disk with AO program.
	0	AO record not stored on disk with AO program.
DEBUG	00000	The trace feature is not to be incorporated in the object output of this program.
	00001	The trace feature is to be incorporated in the object output of this program.

Two versions of the Analog Output program are supplied. Version 1 resides in core as a part of the Skeleton Executive and Version 2 resides entirely on disk. Only the version to be used should be assembled. If the disk stored version is not used, the user may delete the disk reservation for it from the disk using DIM number 0047 in a *DELET control card. This will make sectors 00084 through 00116 available for other use. If the disk version is used, the core version must not be loaded with the Skeleton Executive. To omit the core version, the user must not assemble it (see Loading Procedure). When using the AO-in-core version, it must be loaded to core after all other Executive programs.

Contact Operate, Contact Sense, Clock Read, Manual Entry Read Subprograms, and Exception Core Load

These programs do not include the DEBUG feature. They are assembled by the user in order

to obtain the values for the FORTRAN fixed-point word and floating-point mantissa. These values must be the same as the corresponding values used with the mainline that calls them. The values for F and K in the system communication sector will be used if the user does not specify different values in the SUBR statement.

The user may load the exception core load provided using the procedure described under Loading Core Loads, or he may substitute his own exception core load, if desired.

SERIAL INPUT/OUTPUT CONTROL

The following System Symbol Table labels are used with the Serial Input/Output Control (SIOC) program.

CTVT	XXXXX	Core address of the ETV area
9999X	XXXXX	Core address of the in-core message directory.
9999Y	XXXXX	Core address of the in-core portion of the SIOC program.

The labels 9999X and 9999Y are in the permanent System Symbol Table and are equivalent to core addresses of in-core portions of the SIOC when the Skeleton Executive is loaded.

NOTE: It is necessary to assemble the entire SIOC program if a change is made in the in-core portion.

NUMDEV	000XX	Total number of SIOC units (01-20)
DEV70 through DEV89	WWXXY	Labels DEV70 through DEV89 are used to show which unit response indicators (6070-6089) are associated with the various SIOC units. The numbers (70-89) in the label correspond to indicators 6070-6089. For output printers that have secondary printers, WW is used to specify the numbers (70-89) of the backup printer. Otherwise WW must be 00. XX is the lowest-numbered (first) address of the associated unit and Y is a code that designates the type of unit. 0 = output printer 1 = digital display 2 = sense switch 3 = manual entry
PRPRES	00000	Output printers are not used.
	00001	Output printers are used.
NUMPRT	000XX	Number of output printers used
NUMBUF	000XX	Number of 100-digit core buffers used for outputting printer messages. The number of buffers may range from 1 to 10.
NUMMES	000XX	The total number of messages to be stored on disk for all output printers together. This value can range from 01 to 99 if printers are used. If no printers are used, the value should be zero. If the number of messages specified has already been stored and another output printer call is executed, the SIOC program will continue to print, but in the masked mode until a message printed is complete.
DEBUG	00000	The trace feature is not to be incorporated in the object output of this program.
	00001	The trace feature is to be incorporated in the object output of this program.

LOFCAR	00XXX	Length of printed line (output printers). This program will automatically return the carriage after the specified number of characters have been printed (second and succeeding lines only).

SYSTEM ALERT CONTROL

The following System Symbol Table symbols are used with the System Alert Control (SAC) program.

CTVT	XXXXX	Core Address of the ETV area.
DEV70-DEV89	WWXXY	Two portions of the SAC program are provided and must be assembled. These assemblies can be done at the same time. (See SIOC assembly description.) In SAC these labels are used to determine the output printers in the system. These printers will receive end-of-message operations when a restart is required.

Subroutine Identification Map

The Subroutine Identification map is loaded by DUP in a manner similar to that used to make up the Core Load map; however, the sequence of records in the subroutine map must follow this pattern:

Record 1	Manual Entry and Sense Switch Execute Interrupt routine (SIOC)
Record 2	CE Interrupt Subroutine
Record 3	Process Interrupt or Timed Interrupt
.	.
.	.
.	.
Record N	Process Interrupt or Timed Interrupt

NOTE: Record 1 and 2 must be in the order above. All other ISIM entries are ordered by the user when the Interrupt Indicator table is created. (See MIC) There must not be any blanks in the Subroutine Identification map. If any of the interrupt routines have not been loaded to disk, a routine to indicate that situation with a typeout at execute time will be referenced by the ISIM entry.

Loading Procedure

The following System Symbol Table statements are used with the assembly that causes the Skeleton Executive to be created.

N1	The number of variables in one FORTRAN disk record.
N2	The number of FORTRAN disk records that may be used by the system. (See FORTRAN DEFINE DISK Statement.)

The user has been provided with a small deck of cards (or a tape) which consist of the following:

Monitor card	‡ ‡ JOB
Monitor card	‡ ‡ SPS
FEAP control card	*STORE RELOADABLE
FEAP control card	*SYSTEM SYMBOL TABLE
FEAP control card	*NAME SKELTN
FEAP control card	*OBJECT CORE 2

```
(a) FEAP                    MAIN    N 1, N2
                    START   TDM     0, 11100
                            DSC     1, ', * -1
                            TR      19999, START +7, 2
                            BD      END, 0
                            AM      START +18,20000,7
                            B7      START +12
                    END     SM      START +18, 3099, 7
                            TF      CALLR, START +18
                            TFM     LDR, START
                            TFM     IORT, * +19
                            B7      IOCAL
                            DC      1, 2
                            DC      1, 2
                            DC      1, 0
                            DC      4, 0087
                    CALLR   DSA     00000
                            DSC     1, '
                    LDR     DS      ,3339
                    IORT    DS      ,565
                    IOCAL   DS      ,716
                            DEND    START
Monitor card                ++JOB
Monitor card                ++ XEQS SKELTN
                            *CCEND
Control                     *LDCOR MIC
Cards                       *LDCOR PSC
for Skeleton                *LDCOR ADC
Loader                      *LDCOR SAC
                            *LDCOR SIOC
                            *LDCOR AOC
                            *DKSTR 0133
Monitor card                ++JOB
Monitor card                ++ DUP
DUP                         *DELET SKELTN
Monitor card                ++++
Monitor card                ++PAUS SKELETON EXECUTIVE IS ON DISK
```

The user must have loaded the System Symbol Table with the values for N1 and N2 that are referenced in the card identified here with (a). N1 and N2 are explained in the FORTRAN section.

The user must have previously assembled the Executive programs that must be included with the Skeleton Executive as instructed in the previous section. Any program that is not named in the Equivalence table will not be loaded. Since the name will only be placed in the Equivalence table if the program is assembled, the user must not assemble programs that he does not desire with the Skeleton Executive. Only the programs in the LDCOR control cards listed above will be looked for in the Equivalence table when the Skeleton Executive is loaded.

The MIC program must be loaded first and the AOC program last of the Executive Control programs. If SIOC is used, the SIOC program must be loaded just before AOC, if AOC is in core, or last if AOC is not in core. The SAC program must be loaded just before SIOC, if SIOC is in use. The SKELTN program will load even if it reads LDCOR control statements for programs that have not been assembled and loaded to disk. If any such control statement is read by the loader, the message below will be printed and the loading will resume with the next Executive program named in the following control statement:

PROGRAM NOT LOADED

If a control statement is found to contain a record mark in columns 1-13, the message:

RECORD MARK COL. 1-13
INVALID CONTROL CARD

is printed and the loading is abandoned. If the first six columns do not contain an *LDCOR or *DKSTR, the message:

INVALID CONTROL CARD

is printed and the loading is abandoned.

If the control record is a valid LDCOR record, columns 1-13 are printed and a search for the name of these columns is made in the System Equivalence Table.

If the name is found in the Equivalence table, the program is loaded using the system output loader. The first and last core address loaded are printed in the message format given below:

$\bar{\bar{X}}XXXX$ to $\bar{X}XXXX$

If a valid DKSTR control record is recognized, the message:

TURN SWITCH 1 ON TO INITIALLY LOAD ISIM, OFF TO RETAIN PRESENT ISIM

is printed and the loader halts. If Program Switch 1 is turned on to initially load the ISIM, the loader checks to determine if any ISIM exists on disk. If an ISIM is found (at a location that is consistent with the FORTRAN subroutine set that is in use), the Indicator Table for Interrupts and all of the entries in that ISIM are printed before the initialized ISIM is placed on disk over the previous one. If no such ISIM is found, the message

NO ISIM FOUND ON DISK

is printed, the initialized ISIM is placed on disk, and no other data is printed.

After creating the Skeleton Executive, the user must assemble the disk portions of PSC and of SIOC (if SIOC is used).

STARTING PROCEDURE

After all data is loaded, the Executive System can be started by the following procedure.

Depress the Reset key on the computer console and enter the following data from any available input unit:

```
34  00034  00701
36  00034  00702
49  19400  III
11735700619400
```

The last line contains the disk address, sector count, and core address of the Skeleton Executive loader.

The instructions will cause the Skeleton Executive to be loaded to core storage and the Core Load specified by III to be loaded by PSC and executed. If III is flagged in the units position, the core load is entered in the masked mode.

When the starting procedure described above is executed, the following initialization operations will take place:

1. The interval for logging will be initialized to the value chosen at assembly time.
2. If Console Switch 1 is on, all interrupt indicators will be turned off. If Console Switch 1 is off, all interrupt indicators that are on will be left on.
3. Instruction Register 3 will be loaded with the address of MIC.
4. The computer will be returned to operation using Instruction Register 1.

Changing or Deleting Disk Areas

Eight special areas on disk have been reserved for various purposes in connection with the FORTRAN Executive. These areas are listed below.

	Sector Address	Sector Count	DIM Number
1. Interrupt Exchange Area	00694	106	0053
2. Diagnostic Program and Recorded Interrupt Exchange Area	00500	100	0054
3. Core Contents Exchange Area for use during PSC Special Call	00800	200	0055
4. SIOC Output Printer Message Area	01000	100	0058
5. Arithmetic Routines Exchange Area	00617	043	0051
6. Arithmetic Routines Exchange Area for use when AO on Disk is being executed	00117	033	0048
7. Arithmetic Routines Exchange Area for use when SAC is being executed	01500	033	0068
8. Disk Utility Program System Area (used only in non-process operation)	19881	095	0151

Any of the areas that are not needed (i. e. , Number 6 when AO is in core) may be deleted using the Disk Utility program. The DIM number must be specified.

The first four areas in the list are areas that might be changed to meet the particular requirements of the user's system. These changes, if any, should be made before assembling the Executive Programs. In other terms, the changing of these areas is a part of defining the system.

If the SIOC message area on disk is relocated, the user must select a disk address that ends in three zeros (i. e. , 121000, 005000, etc.).

Procedure for Changing the Disk Areas Defined by DIMs

The following procedure should be used when redefining any disk areas for the FORTRAN Executive system.

1. Select the DIM for the area that is to be redefined. (i. e. , DIM 0058 for the SIOC Output Printer Message Area).
2. Punch a DUP DELET control card with the DIM number selected.
3. Follow the DUP DELET procedure.
4. Punch a DUP DLOAD control card to load to the selected DIM with enough sectors (limit 999) from the work cylinders and to the sector address that is the first sector of the desired area. (It is also possible in most cases to permit DUP to determine the location of the new area.)
5. Follow the DUP DLOAD procedure.

Program Areas and Special Areas that may be Deleted if not Required with the 1710 System

	Sector Address	Sector Count	DIM
DKIO for Flip Version Sets 2 and 4	00000	020	0043
Arithmetic Routines for Sets 1 and 3	00020	033	0044
DKIO, FIX, FLOAT for Sets 1 and 3	00053	031	0045
Analog Output Routine and AO Records Area for Sets 1, 2, 3 and 4 (may be deleted if core version of AO or if no version of AO is used).	00084	033	0047
Analog Output Exchange area while performing disk version of AO.	00117	033	0048
DKIO for Flip Version Sets 1 and 3	00150	020	0080
DKIO for Flip Version Sets 1 and 3	00170	030	0081
*FORTRAN Nondisk I/O	00200	135	0088
Routines for A **I, I**J and ATNF for flip version Sets 1 and 3.	00335	030	0046
Routines for A**B, LNF and EXPF for flip version Sets 1 and 3.	00365	030	0050
Routines for SQRTF, COSF and SINF for flip version Sets 1 and 3.	00600	017	0049
Arithmetic Routines for Sets 1 and 3 (duplicate)	00660	033	0052
**Special Interrupt Core Load for use when another ICL is deleted.	00693	001	0086
***Exchange Area for Mainline during PSC special call	00800	200	0055
SIOC Output Printer Message Buffer	01000	100	0058
Routines for DKIO & subscripting used with Sets 2 and 4	01100	033	0056
Routines for A**I, I**J, and ATANF for Sets 2 and 4	01133	033	0057
Routines for A**B, LNF and EXPF for Sets 2 and 4	01166	033	0059
Routines for SQRTF, SINF and COSF for Sets 2 and 4	01200	024	0090
SIOC Formatting Routines	01224	145	0060
SIOC Manual Entry, Sense Switch and Digital Display	01375	025	0061
SIOC Store Message Routine and SIOC Error Routines	01400	060	0062
SIOC End of Message Routine	01460	040	0063
****Skeleton Loader	17370	030	0087
DUP DCOPY	16511	042	0141
DUP DALTR	16553	039	0142
DUP DFINE	16592	039	0070
DUP DFINE	16631	075	0079
DUP DREPL (Phase 2)	16706	074	0075
DUP DLABL	16780	020	0140
DUP DREPL (Phase 1)	18399	041	0072
DUP DWRAD	19342	029	0154
DUP DFLIB	19371	029	0155

* FORTRAN Nondisk input-output must not be used if these routines are deleted.
** May be deleted after all Interrupt Core Loads are permanently loaded.
*** May be deleted if no PSC scecial calls are performed.
**** May be deleted after the Skeleton Executive is permanently loaded.

Any other area not used may be deleted.

SEQUENTIAL PROGRAM LIST EDIT ROUTINE

The SP List Edit Routine will check the DIM entries and the Sequential Program List on the Monitor pack. If more than one DIM entry describes any disk area on the Monitor pack, the higher-numbered DIM entry that defines the area will be removed. An SP List will be created during the DIM edit and compared against the existing SP List at the conclusion of the edit. The user may print the old SP List if it is incorrect. A correct SP List will always result. If any DIM entries were removed, the numbers of those entries will be printed at the end of the execution of the routine. If any DIM entries with incorrect form are found, they will be typed along with the number of the entry, and then removed from the DIM table.

Operation

Place the cards (tape) in the input unit. Precede with a JOB control record.

Results:

1. DIM Table Edit
   If a DIM entry is in error, the message "DIM FORMAT ERROR", the DIM number and the DIM entry are typed. This entry will be deleted from the DIM Table and any names which refer to that DIM number will be replaced, in the Equivalence table, with an entry of all 8's.
2. Sequential Program List Creation
   If the new Sequential Program List, which is built up from the DIM entries, compares equally with the one on the disk, the message, "S. P. LIST IS CORRECT" is typed. If the Lists do not compare equally, the message, "S. P. LIST IS INCORRECT" is typed and the program halts. Turning Console Switch 1 on and pressing Start will dump the incorrect list (the one that was on the disk) onto the typewriter. In either case, the new list will be written on the disk when the Start key is pressed.

   Any DIM numbers which are typed after the SP List message has been typed have been deleted from the DIM Table and replaced by 8's in the Equivalence Table. These 8's will not interfere with the correct handling of the SP List by the DUP routines.

FORTRAN STATEMENTS:

Control Statements

CALL EXIT
CONTINUE
DO n i = m_1, m_2
or
DO n i = m_1, m_2, m_3
END or END (I_1, I_2, I_3, I_4, I_5)
GO TO n
GO TO (n_1, n_2, .., n_m), i
IF (a)n_1, n_2, n_3
IF (SENSE SWITCH i) n_1, n_2
PAUSE or PAUSE n
STOP or STOP n

Input/Output Statements

ACCEPT n, List
ACCEPT TAPE n, List
FETCH (I) List
FIND (I)
PRINT n, List
PUNCH n, List
READ n, List
RECORD (I) List
TYPE n, List

Specification Statements

COMMON A, B ..
DEFINE DISK (N_1, N_2)
DIMENSION v, v, v, ...
EQUIVALENCE (a, b, c, ...), (d, e, f, ...) , ...
FORMAT (S_1,.., S_n)

Subprogram Statements

CALL LINK (Name)
CALL Name (a_1, a_2, ..., a_n)
FUNCTION Name (a_1, a_2,...,a_n)
SUBROUTINE Name (a_1, a_2,...,a_n)
RETURN

APPENDIX B: FEAP MNEMONICS

Summary of FEAP Declarative Operations

NOTE: Except for the constants in DC, DSC, and DAC, all operands may be actual or symbolic. All symbolic length and address operands must be previously defined. All operands may use address adjustment. Remarks may follow operands except in DSA and DVLC statements. "Alpha Record Address" in the table refers to the leftmost position plus one of an alphameric field, whereas "Field Address" refers to the rightmost position of a field. The term "Numerical Record Address" refers to the leftmost position of a field.

DECLARATIVE STATEMENT FORMAT			AMOUNT ADDED TO LOCATION ASSIGNMENT COUNTER IF ADDRESS (A) IS BLANK	VALUE STORED IN SYMBOL TABLE AS EQUIVALENT TO "SYMBOL"	DATA FIELDS WHICH ARE LOADED AS A PART OF THE OBJECT PROGRAM
LABEL	MNEMONIC	OPERANDS			
SYM	DS	L,A	L (length). If L is blank, 0 is added.	A address. If A is blank, the field address from the location assignment counter is stored.	None.
SYM	DSS	L,A	L (length). If L is blank, 0 is added.	A address. If A is blank, the numerical record address from the location assignment counter is stored.	None.
SYM	DAS	L,A	2 x L is added. If L is blank, 0 is added.	A address must be odd. If A is blank, the alpha record address from the location assignment counter is stored.	None.
SYM	DC	L,C,A	L is added.	A address. If A is blank, the field address from the location assignment counter is stored.	C, the (numerical) constant.
SYM	DSC	L,C,A	L is added.	A address. If A is blank the numerical record address from the location assignment counter is stored.	C, the (numerical) constant.
SYM	DVLC	A,L,C, L,C, etc.	L is added.	First C address.	C, C, etc., the (numerical) constants.
SYM	DAC	L,C,A	2 x L is added.	A address must be odd. If A is blank, the alpha record address from the location assignment counter is stored.	C, the (alphameric) constant.
SYM	DSA	D,E,F,G, H, I, J, K, L,M	5 x (number of addresses) is added.	Field address of the first address on list.	A list of the actual addresses that correspond to D,E,F, etc.
SYM	DSB	L,N,A	Length of each element times the number of elements is added.	A address. If A is blank, field address of the first element is stored.	None.
SYM	DNB	L,A	L is added.	A address. If A is blank, the field address from the location assignment counter is stored.	Number of blank characters that equal L.
SYM	DDA	A,D,F,S, M	14, length of a disk control field.	(Same as DSC).	D,F,S,M.
SYM	DGM	A	1	A address or location counter.	≢ (Group Mark).
SYM	DSCI	D,E,F,G, H,I,J,K, L,M	5 times the number of addresses is added.	Addresses of leftmost position of the generated constant.	A list of the channel addresses (5-digits for each address) that correspond to D,E,F,G, etc.
SYM	DFLC	L,C,A	L (mantissa length) + 2 If L is blank, ff (mantissa length) + 2 is added	A address. If A is blank, the field address from the location assignment counter is used.	C, the (floating point) constant.

Summary of FEAP Arithmetic Instructions

NOTE: Indirect Addressing and indexing are allowable with all P address operands listed below. An * to the left of the Q operand indicates these features may be used with it.

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Add	A	21	Storage address of units position of augend	*Storage address of units position of addend
Add Immediate	AM	11	Same as code 21	Q_{11} of instruction is units position of addend
Subtract	S	22	Storage address of units position of minuend	*Storage address of units position of subtrahend
Subtract Immediate	SM	12	Same as code 22	Q_{11} of instruction is units position of subtrahend
Multiply	M	23	Storage address of units position of multiplicand	*Storage address of units position of multiplier
Multiply Immediate	MM	13	Same as code 23	Q_{11} of instruction is units position of multiplier
Load Dividend	LD	28	Storage address in product area to which units position of field (dividend) is to be transmitted	*Storage address of units position of dividend
Load Dividend	LDM	18	Same as code 28	Q_{11} of instruction is units position of dividend
Divide	D	29	Storage address at which first subtraction of the divisor occurs	*Storage address of units position of divisor
Divide Immediate	DM	19	Same as code 29	Q_{11} of instruction is units position of divisor
Floating Add	FADD	01	Storage address of units position of exponent of augend	*Storage address of units position of exponent of addend
Floating Subtract	FSUB	02	Storage address of units position of exponent of minuend	*Storage address of units position of exponent of subtrahend
Floating Multiply	FMUL	03	Storage address of units position of exponent of multiplicand	*Storage address of units position of exponent of multiplier
Floating Divide	FDIV	09	Storage address of units position of exponent of dividend	*Storage address of units position of exponent of divisor

Summary of FEAP Internal Data Transmission Instructions

NOTE: Indirect Addressing and indexing are allowable with all P address operands listed below. An * to the left of the Q address operand indicates these features may be used with it.

OPERATION	OPERATION CODES		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Transmit Digit	TD	25	Storage address to which single digit is transmitted	*Storage address of single digit to be transmitted
Transmit Digit Immediate	TDM	15	Same as code 25	Q_{11} of instruction is the single digit to be transmitted
Transmit Field	TF	26	Storage address to which units position of field is transmitted	*Storage address of units position of field to be transmitted
Transmit Field Immediate	TFM	16	Same as code 26	Q_{11} of instruction is the units position of the field to be transmitted
Transmit Record	TR	31	Storage address to which high-order position of the record is transmitted	*Storage address of high-order position of the record to be transmitted
Transmit Record No Record Mark	TRNM	30	Same as code 31	*Same as code 31
Transfer Numerical Strip	TNS	72	Storage address of rightmost position of alphameric field to be transmitted	*Storage address of the units position of the numerical field
Transfer Numerical Fill	TNF	73	Storage address of rightmost position of alphameric field	*Storage address of the units position of the numerical field to be transmitted
Floating Shift Right	FSR	08	Storage address to which units (rightmost) digit of mantissa is transmitted	*Storage address (rightmost) digit of mantissa to be transmitted
Floating Shift Left	FSL	05	Storage address to which high-order digit of the mantissa is transmitted	*Storage address of low-order digit of mantissa to be transmitted
Transmit Floating	TFL	06	Storage address to which units position of exponent is transmitted	*Storage address of units position of exponent of field to be transmitted
Move Address	MA	70	Storage address of units position of 5-digit field to which data is transmitted	*Storage address of units position of 5-digit field to be transmitted
OR to Field	ORF	92	Storage address of leftmost position of first field for OR logic input	*Storage address of leftmost position of second field for OR logic input
AND to Field	ANDF	93	Storage address of leftmost position of first field for AND logic	*Storage address of leftmost position of second field for AND logic
Exclusive OR to Field	EORF	95	Storage address of leftmost position of first field for Exclusive OR logic	*Storage address of leftmost position of second field for Exclusive OR logic
Complement Ocial Field	CPLF	94	Storage address of leftmost position of field to which data is transmitted	*Storage address of leftmost position of field to be complemented
Octal to Decimal Conversion	OTD	96	Storage address of the units position of the power-of-eight table	*Storage address of leftmost position of field to be converted
Decimal to Octal Conversion	DTO	97	Storage address of the units position of the highest power-of-eight required	*Storage address of leftmost position of field to be converted

Summary of FEAP Logic (Branch and Compare) Instructions

NOTE: Both the BI (Branch Indicator) and BNI (Branch No Indicator) instructions require one of the switch or indicator codes listed in Table 21 as a Q address. The code indicates the switch or indicator to be interrogated for status. To relieve the programmer of having to code a Q address, unique mnemonics are included in SPS language for both BI- and BNI-type instructions. For a unique mnemonic, the processor generates the actual machine language code 46 (Branch Indicator) or 47 (Branch No Indicator) and the Q address that represents the switch or indicator.

Indirect Addressing and indexing are allowable with all P address operands listed below except Branch Back. An * to the left of the Q address operand indicates these features may be used with it.

OPERATION	OPERATION CODES		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Compare	C	24	Storage address of units position of the field to which another field is compared	*Storage address of units position of the field to be compared with the field at the P address
Compare Immediate	CM	14	Same as code 24	Q_{11} of instruction is units position of the field to be compared with the field at the P address
Branch	B	49	Storage address of the leftmost digit of the next instruction to be executed	Not used
Branch and Adjust Assignment Counter	B7	49	Storage address of the leftmost digit of the next instruction to be executed	Not used. However, these five locations are used by the next instruction in sequence
Branch No Flag	BNF	44	Storage address of the leftmost digit of next instruction to be executed if branch occurs	*Storage address to be interrogated for presence of a flag bit
Branch No Record Mark	BNR	45	Same as code 44	*Storage address to be interrogated for presence of a record mark character
Branch No Group Mark	BNG	55	Same as code 44	*Storage address to be interrogated for presence of a group mark character
Branch on Digit	BD	43	Same as code 44	*Storage address to be interrogated for a digit other than zero
Branch Indicator	BI	46	Storage address of leftmost position of next instruction to be executed if indicator tested is on	Q_8 and Q_9 of instruction specify the program switch or indicator to be interrogated
Unique Branch Indicator Mnemonics:				
Branch High	BH	46	Same as BI	None required
Branch Positive	BP	46	Same as BI	None required
Branch Equal	BE	46	Same as BI	None required
Branch Zero	BZ	46	Same as BI	None required
Branch Overflow	BV	46	Same as BI	None required
Branch Any Data Check	BA	46	Same as BI	None required

Summary of FEAP Logic (Branch and Compare) Instructions (cont'd.)

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Branch Not Low	BNL	46	Same as BI	None required
Branch Not Negative	BNN	46	Same as BI	None required
Branch Band A Selected	BBAS	46	Same as BI	None required
Branch Band B Selected	BBBS	46	Same as BI	None required
Branch Neither Band Selected	BNBS	46	Same as BI	None required
Branch Console Switch 1 On	BC1	46	Same as BI	None required
Branch Console Switch 2 On	BC2	46	Same as BI	None required
Branch Console Switch 3 On	BC3	46	Same as BI	None required
Branch Console Switch 4 On	BC4	46	Same as BI	None required
Branch Last Card	BLC	46	Same as BI	None required
Branch Exponent Check	BXV	46	Same as BI	None required
Branch No Indicator	BNI	47	Storage address of leftmost position of next instruction to be executed if indicator tested is off	Q_8 and Q_9 of instruction specify program switch or indicator to be interrogated (see Table 21)
Unique Branch No Indicator Mnemonics:				
Branch Band A Not Selected	BANS	47	Same as BNI	None required
Branch Band B Not Selected	BBNS	47	Same as BNI	None required
Branch Either Band Selected	BEBS	47	Same as BNI	None required
Branch Not High	BNH	47	Same as BNI	None required
Branch Not Positive	BNP	47	Same as BNI	None required
Branch Not Equal	BNE	47	Same as BNI	None required
Branch Not Zero	BNZ	47	Same as BNI	None required

Summary of FEAP Logic (Branch and Compare) Instructions (cont'd.)

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Branch No Overflow	BNV	47	Same as BNI	None required
Branch Not Any Data Check	BNA	47	Same as BNI	None required
Branch Low	BL	47	Same as BNI	None required
Branch Negative	BN	47	Same as BNI	None required
Branch Console Switch 1 Off	BNC1	47	Same as BNI	None required
Branch Console Switch 2 Off	BNC2	47	Same as BNI	None required
Branch Console Switch 3 Off	BNC3	47	Same as BNI	None required
Branch Console Switch 4 Off	BNC4	47	Same as BNI	None required
Branch Not Last Card	BNLC	47	Same as BNI	None required
Branch Not Exponent Check	BNXV	47	Same as BNI	None required
Branch and Transmit	BT	27	P address minus one is the storage address to which the units position of the Q field is transmitted. P address is leftmost digit of the next instruction to be executed	*Storage address of units position of the field to be transmitted
Branch and Transmit Immediate	BTM	17	Same as code 27	Q_{11} of instruction is units position of field to be transmitted
Branch Back	BB	42	Not used	Not used
Branch Back and Adjust Assignment Counter	BB2	42	Not used. However, these five locations <u>are</u> used by the next instruction in sequence	Not used. However, these five locations <u>are</u> used by the next instruction in sequence
Branch and Transmit Floating	BTFL	07	P address minus one is the storage address to which the units position of the exponent portion of the Q field is transmitted. P is the storage address of the leftmost digit of the next instruction to be executed	*Storage address of units position of exponent of field to be transmitted
Branch and Select	BS	60	Storage address of the leftmost position of the next instruction	Q_{11} specifies condition to be selected
Unique Branch and Select Mnemonics:				
Branch and Select Indirect Addressing	BSIA	60	Same as BS	None required

Summary of FEAP Logic (Branch and Compare) Instructions (cont'd.)

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Branch and Select No I/A	BSNI	60	Same as BS	None required
Branch and Select Band A	BSBA	60	Same as BS	None required
Branch and Select Band B	BSBB	60	Same as BS	None required
Branch and Select No Index Register	BSNX	60	Same as BS	None required
Branch and Modify Index Register	BX	61	Same as BS	**Storage address of units position of field to be added to selected index register
Branch and Modify Index Register Immediate	BXM	62	Same as BS	**Five digits of Q field are added to selected index register
Branch Conditionally, Modify Index Register	BCX	63	Same as BS if (after modification) IX sign has not changed or result is not zero	Same as BX
Branch Conditionally, Modify Index Register Immediate	BCXM	64	Same as BCX	Same as BXM
Branch and Load Index Register	BLX	65	Same as BS	**Storage address of units position of 5-digit field to be loaded to selected index register
Branch and Load Index Register Immediate	BLX	66	Same as BS	**Five digits of Q field are loaded to selected index register
Branch and Store Index Register	BSX	67	Same as BS	**Storage address of units position of field where selected index register data is to be stored
Branch on Bit	BBT	90	Storage address of the leftmost position of next instruction if comparison is successful	*Q_{8-11} specifies storage address of units position of field to be compared with bits of the Q_7 digit
Branch on Mask	BMK	91	Same as code 90	*Q_{8-11} specifies storage address of units position of field to be compared with Q_7 digit

**Specific index register is selected by flags over the Q_{8-10} positions of the instruction.

Summary of FEAP Input and Output Instructions

NOTE: Indirect Addressing and indexing are allowable with all P address operands, where a P operand is required. None of the Q operands shown may be used with Indirect Addressing or Index Registers.

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Read Numerically	RN	36	Storage address at which leftmost (first) numerical character is stored	Q_8 and Q_9 of instruction specify input unit
Unique Read Numerically Mnemonics:				
Read Numerically Typewriter	RNTY	36	Same as RN	None required
Read Numerically Paper Tape	RNPT	36	Same as RN	None required
Read Numerically Card	RNCD	36	Same as RN	None required
Write Numerically	WN	38	Storage address from which leftmost (first) numerical character is written	Q_8 and Q_9 of instruction specify output unit
Unique Write Numerically Mnemonics:				
Write Numerically Typewriter	WNTY	38	Same as WN	None required
Write Numerically Paper Tape	WNPT	38	Same as WN	None required
Write Numerically Card	WNCD	38	Same as WN	None required
Dump Numerically	DN	35	Same as WN	Same as WN
Unique Dump Numerically Mnemonics:				
Dump Numerically Typewriter	DNTY	35	Same as WN	None required
Dump Numerically Paper Tape	DNPT	35	Same as WN	None required
Dump Numerically Card	DNCD	35	Same as WN	None required
Read Alphamerically	RA	37	Storage address at which numerical digit of leftmost (first) character is stored. (Zone digit of first character is at P minus one)	Q_8 and Q_9 of instruction specify input unit

Summary of FEAP Input and Output Instructions (cont'd.)

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Unique Read Alphamerically Mnemonics:				
Read Alphamerically Typewriter	RATY	37	Same as RA	None required
Read Alphamerically Paper Tape	RAPT	37	Same as RA	None required
Read Alphamerically Card	RACD	37	Same as RA	None required
Read Binary Paper Tape	RBPT	37	Same as RA	None required
Write Alphamerically	WA	39	Storage address of numerical digit of leftmost (first) character to be written. (Zone digit of first character is at P minus one)	Q_8 and Q_9 of instruction specify output unit
Unique Write Alphamerically Mnemonics:				
Write Alphamerically Typewriter	WATY	39	Same as WA	None required
Write Alphamerically Paper Tape	WAPT	39	Same as WA	None required
Write Alphamerically Card	WACD	39	Same as WA	None required
Write Binary Paper Tape	WBPT	39	Same as WA	None required
Control	K	34	Not used	Q_8 and Q_9 specify input/output unit. Q_{11} specifies control functions
Unique Control Mnemonics:				
Backspace Typewriter	BKTY	34	Not used	None required
Tabulate Typewriter	TBTY	34	Not used	None required
Index Typewriter	IXTY	34	Not used	None required
Return Carriage Typewriter	RCTY	34	Not used	None required
Space Typewriter	SPTY	34	Not used	None required

Summary of FEAP Input and Output Instructions (cont'd.)

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Seek	SK	34	Storage address of disk control field	X07X1
Read Disk/ WLRC	RDGN	36	Same as SK	X07X0
Write Disk/ WLRC	WDGN	38	Same as SK	X07X0
Check Disk/ WLRC	CDGN	36	Same as SK	X07X1
Read Disk Track/WLRC	RTGN	36	Same as SK	X07X4
Write Disk Track/WLRC	WTGN	38	Same as SK	X07X4
Check Disk Track/WLRC	CTGN	36	Same as SK	X07X5
Read Disk	RDN	36	Same as SK	X07X2
Write Disk	WDN	38	Same as SK	X07X2
Check Disk	CDN	36	Same as SK	X07X3
Read Disk Track	RTN	36	Same as SK	X07X6
Write Disk Track	WTN	38	Same as SK	X07X6
Check Disk Track	CTN	36	Same as SK	X07X7

Summary of FEAP Miscellaneous Instructions

NOTE: Indirect Addressing and indexing are allowable with all P or Q address operands that are marked with an *.

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Set Flag	SF	32	*Storage address at which flag bit is placed	Not used
Clear Flag	CF	33	*Storage address from which flag bit is cleared	Not used
Move Flag	MF	71	*Storage address to which flag bit is moved	*Storage address of flag bit to be moved
Halt	H	48	Not used	Not used
No Operation	NOP	41	Not used	Not used

1710 FEAP Operation Codes

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Select Address and Operate	SAO	84	Not used	Q_7 specifies operation; Q_9 - Q_{11} specify a terminal address
Unique Select Address and Operate Mnemonics:				
Select Address	SA	84	Same as SAO	Q_7 = 1; Q_9 - Q_{11} specify terminal address of analog input point
Select Address and Contact Operate	SACO	84	Same as SAO	Q_7 = 2; Q_9 - Q_{11} specify terminal address of contact point
Select Analog Output Signal	SAOS	84	Same as SAO	Q_7 = 3; Q_9 - Q_{11} specify terminal address of analog output channel
Select Read Numerically	SLRN	86	Depends upon particular operation	Depends upon particular operation
Unique Select Read Numerically Mnemonics:				
Select TAS	SLTA	86	Core location where high-order position of TAS is transferred	Q_7 = 1; Q_8 - Q_{11} are not used
Select ADC Register	SLAR	86	Core location where high-order position of ADC register is transferred	Q_7 = 2; Q_9 - Q_{11} specify analog input address
Select Contact Block	SLCB	86	Core location where status of the first contact scanned is stored	Q_7 = 7; Q_9 - Q_{11} specifies the contact block address where reading begins
Select Real-Time Clock	SLTC	86	Core location where high-order digit of RTC is transferred	Q_7 = 4; Q_8 - Q_{11} are not used
Select ADC and Increment (1711 Model 1)	SLAD	86	Core location where high-order position of ADC is transferred	Q_7 = 6; Q_8 - Q_{11} are not used
Select Manual Entry Switches	SLME	86	Core location where high-order digit of Manual Entry switches is transferred	Q_7 = 8; Q_8 - Q_{11} are not used
Branch Out Of Noninterruptible Mode	BO	47	Address to be placed in IR-3	Q_8 - Q_9 = 00; Q_{11} = 0
Branch Out Of Noninterruptible Mode and Load	BOLD	47	Address to be placed in IR-1	Q_8 - Q_9 = 00; Q_{11} = 1
Mask	MK	46	Not used	Q_8 - Q_9 = 00; Q_{11} = 1
Unmask	UMK	46	Not used	Q_8 - Q_9 = 00; Q_{11} = 0
Select Input Channel	SLIC	86	Not used	Q_{10} - Q_{11} specify the address of an S1OC input unit

1710 FEAP Operation Codes (cont'd.)

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Read Numerical Input Channel	RNIC	$\bar{86}$	Core storage location where data is to be read	$Q_7 = 5$; $Q_8 - Q_{11}$ not used
Read Alphameric Input Channel	RAIC	$\bar{87}$	Same as RNIC	Same as RNIC
Write Numerical Output Channel	WNOC	88	Core storage location from which data is to be written	$Q_{10} - Q_{11}$ specify an SIOC output unit
Write Alphameric Output Channel	WAOC	89	Same as WNOC	Same as WNOC
Unique SIOC Branch Indicator Mnemonics:				
Branch Output Record Mark	BOR	46	Core storage address of leftmost position of next instruction to be executed if indicator tested is on	None required
Branch End of Message	BRE	46	Same as BOR	Same as BOR
Branch Mode Shift	BMC	46	Same as BOR	Same as BOR
Branch Data Ready	BIR	46	Same as BOR	Same as BOR
Branch SIOC Not Busy	BCNB	46	Same as BOR	Same as BOR
Unique SIOC Branch No Indicator Mnemonics:				
Branch No Output Record Mark	BNOR	47	Core storage address of leftmost position of next instruction to be executed if indicator tested is off	None required
Branch No End of Message	BNRE	47	Same as BNOR	Same as BNOR
Branch No Mode Shift	BNMC	47	Same as BNOR	Same as BNOR
Branch No Data Ready	BNIR	47	Same as BNOR	Same as BNOR
Branch No SIOC Not Busy	BCB	47	Same as BNOR	Same as BNOR

1710 FEAP Operation Codes (cont'd)

OPERATION	OPERATION CODE		OPERANDS	
	MNEMONIC	ACTUAL	P ADDRESS	Q ADDRESS
Reset Timer	RT	84	Not usea.	$Q_7 = 4$, $Q_8 - Q_{11}$ not used.
Read Pulse Counter	RPC	86	Core storage location where contents of Pulse Counter Register are placed.	$Q_7 = 3$, $Q_9 - Q_{11}$ specify next pulse counter address.
Read Pulse Counter, Reset	RPCR	86	Same as RPC	$Q_7 = 9$, $Q_9 - Q_{11}$ specify next pulse counter address.
Scan Synchronized	SCNS	86	Core storage location where digital value of analog signal is placed.	$Q_7 = 6$, $Q_9 = 1$
Scan Free	SCNF	86	Same as SCNS	$Q_7 = 6$, $Q_9 = 0$
Select Latching Contact Operate	SLCO	88	Core storage location where the status of first 2 contacts is placed.	$Q_7 = 7$, Q_{11} specifies matrix row number.
Set Real Time Clock	SRTC	88	Core storage location of leftmost position of data that is set into the RTC.	$Q_7 = 4$
Select Digital Input	SLDI	86	Core storage location of leftmost position of field where data is to be stored.	$Q_7 = 0$, $Q_{10} - Q_{11}$ specify terminal address where reading is to begin.

1620/1710 Indicator Codes for BI-BNI Instructions

NOTE: This table lists only those indicators that do not have unique mnemonics.

INDICATOR	Q ADDRESS				
	Q_7	Q_8	Q_9	Q_{10}	Q_{11}
Read Check		0	6		
Write Check		0	7		
MAR Check		0	8		
MBR-E Check		1	6		
MBR-O Check		1	7		
Operator Entry		1	8		
Terminal Address Selector (TAS) Check		2	1		
Function Register Check		2	2		
Analog Output (AO) Check		2	3		
Mask		2	6		
Customer Engineer (CE) Interrupt		2	7		
Analog Output Setup		2	8		
Multiplexer Busy		2	9		
Multiplex Complete		4	0		
Analog Output Setup Interrupt		4	1		
One Minute Interrupt		4	3		
One Hour Interrupt		4	4		
Any SIOC Interrupt		4	5		
Process Interrupt 1		4	8		
Process Interrupt 2		4	9		
Process Interrupt 3		5	0		
Process Interrupt 4		5	1		
Process Interrupt 5		5	2		
Process Interrupt 6		5	3		
Process Interrupt 7		5	4		
Process Interrupt 8		5	5		
Process Interrupt 9		5	6		
Process Interrupt 10		5	7		
Process Interrupt 11		5	8		
Process Interrupt 12		5	9		
Process Branch Indicator 1		7	0		
Process Branch Indicator 2		7	1		
Process Branch Indicator 3		7	2		
Process Branch Indicator 4		7	3		
Process Branch Indicator 5		7	4		
Process Branch Indicator 6		7	5		
Process Branch Indicator 7		7	6		

INDICATOR	Q ADDRESS				
	Q_7	Q_8	Q_9	Q_{10}	Q_{11}
Process Branch Indicator 8		7	7		
Process Branch Indicator 9		7	8		
Process Branch Indicator 10		7	9		
Process Branch Indicator 11		8	0		
Process Branch Indicator 12		8	1		
Process Branch Indicator 13		8	2		
Process Branch Indicator 14		8	3		
Process Branch Indicator 15		8	4		
Process Branch Indicator 16		8	5		
Process Branch Indicator 17		8	6		
Process Branch Indicator 18		8	7		
Process Branch Indicator 19		8	8		
Process Branch Indicator 20		8	9		
SIOC Output Error		6	0	4	3
Alert		6	0	4	5
SIOC Unit 1 Response		6	0	7	0
SIOC Unit 2 Response		6	0	7	1
SIOC Unit 3 Response		6	0	7	2
SIOC Unit 4 Response		6	0	7	3
SIOC Unit 5 Response		6	0	7	4
SIOC Unit 6 Response		6	0	7	5
SIOC Unit 7 Response		6	0	7	6
SIOC Unit 8 Response		6	0	7	7
SIOC Unit 9 Response		6	0	7	8
SIOC Unit 10 Response		6	0	7	9
SIOC Unit 11 Response		6	0	8	0
SIOC Unit 12 Response		6	0	8	1
SIOC Unit 13 Response		6	0	8	2
SIOC Unit 14 Response		6	0	8	3
SIOC Unit 15 Response		6	0	8	4
SIOC Unit 16 Response		6	0	8	5
SIOC Unit 17 Response		6	0	8	6
SIOC Unit 18 Response		6	0	8	7
SIOC Unit 19 Response		6	0	8	8
SIOC Unit 20 Response		6	0	8	9
Disk Address Check		3	6		
WLR/RBC		3	7		
Cylinder Overflow		3	8		
Any Disk Check		3	9		
Seek Complete		4	2		

APPENDIX C

EXECUTIVE SUMMARY AND LOADING SEQUENCE

The following summary is intended to assist the user in preparing to use the FORTRAN Executive System. This section presumes that the reader is familiar with the other sections of the manual. It serves to tie the various segments together by showing a chronological sequence of events with control card examples where appropriate.

1. Load the system using the System Loader provided (see Monitor Loader).

NOTE: On-line IORT is a self-loading program.

2. Define the following system parameters using the DUP *DFINE card (see DUP DFINE
   a. F and k, keeping in mind that f and k must be the same in all process-control programs.
   b. Core size of object machine.
   c. The subroutine set to be used.
   d. Mainline origin (MANLIN). The first time the system is assembled this must be estimated by use of the following table.

Base Address Short Form (Sets 1 or 3)	7600
Base Address Long Form (Set 2)	9700
Base Address Long Form (Set 4)	9100
Analog Input (ADC)	950
Analog Output in Core (AOC)	1400
SIOC	2800
PSC	600
MIC	2680
SAC	500

Add the base address to the sum of the other routines used, (PSC, SAC, MIC are always used) and adjust to the next higher even hundred address.

Example

```
##JOB
##DUP
*DFINE          08  05      5    1  15000
```

3. Punch the System Symbol Table cards as explained in the Assembly and Loading Procedure Section

Example

```
##JOB
##XEQ SPSLIB
*DEFINE SYSTEM SYMBOL TABLE
(Columns 6-11 and 16-75)
     CTVT        7600
     DEBUG       1
     SHORTF      1
     AOIND       1
     SIIND       1
     AIIND       1
     PIN01       48
```

```
PIN02     49
PIN03     50
PIN04     51
PIN05     52
PIN06     53
PIN07     43
PIN08     44
PIN09     18
PIN10     54
PIN11     47
PIN12     0
PIN13     0
PIN14     0
PIN15     0
NPIN      12
AOPRDK    1
MANLIN    15000
DCTVT     76
TIME      0
OVOPT1    0
OVOPT2    0
OVOPT3    1
K         4
AOTBAD    0
AOSLEW    50
AOTRIM    51
NOSPP     4
AOTBDK    1
NUMDEV    00005
DEV70     70000
DEV71     00501
DEV72     00242
DEV73     00333
DEV74     74100
DEV75     00000
DEV76     00000
DEV77     00000
DEV78     00000
DEV79     00000
DEV80     00000
DEV81     00000
DEV82     00000
DEV83     00000
DEV84     00000
DEV85     00000
DEV86     00000
DEV87     00000
DEV88     00000
DEV89     00000
PRPRES    00001
NUMPRT    00002
NUMBUF    00002
NUMMES    14
LOFCAR    00087
    N1    19
    N2    1598
*ENDLIB
```

4. Assemble the following Executive Control programs. (See table below for required control cards.)
   a. Skeleton Loader
   b. MIC
   c. PSC Core
   d. SAC Core
   e. ADC (if used)
   f. SAC Disk
   g. SIOC Core (if SIOC is used)
   h. AOC Core or Disk (if used; only the program to be used should be assembled)

NOTE: PSC Disk and SIOC Disk cannot be assembled until after Step 5 is performed.

Deck/ Tape Number	Name	Control Cards Required
14 15 16 17 18 19	MIC (core) PSC (core) ADC (core) SAC (core) SIOC (core) AOC (core)	++JOB ++SPS *ASSEMBLE RELOCATABLE *STORE RELOADABLE *SYSTEM SYMBOL TABLE *NAME aaaaaa *BEGIN PAPER TAPE INPUT (tape only)
21 22 23 24 25 26 27	SIOC Store Message Routine SIOC End of Message Routine SIOC Error Routine SIOC ME, SS, DD AOC PSC SAC	++JOB ++SPSX *SYSTEM SYMBOL TABLE *BEGIN PAPER TAPE INPUT (tape only)
20 28 29 30 31 32	Mainline for loading Skeleton Executive Exception Core Load Manual Entry Subprogram Real-Time Clock Subprogram Contact Sense Subprogram Contact Operate Subprogram	++JOB ++SPS *STORE RELOADABLE *OBJECT CORE *SYSTEM SYMBOL TABLE *BEGIN PAPER TAPE INPUT (tape only)
33	Control records to form Skeleton Executive	++XEQS SKELTN *CCEND
34	FEAP Sample Program	++JOB ++SPSX
35	FORTRAN II-D Sample Program	++JOB ++FORX *CCEND (tape only)

5. Load the Skeleton Executive by using the deck of cards provided. Cards for those programs that do not apply for your system may be left in the deck. These programs must not be assembled, however.

Example

```
##JOB
##XEQSSKELTN
*CCEND
*LDCOR MIC
*LDCOR PSC
*LDCOR ADC
*LDCOR SAC
*LDCOR SIOC
*LDCOR AOC
*DKSTR 0133
```

NOTE: SAC must preceed SIOC and AOC. AO Core, if used, must be loaded last. At this point, the loading typeout should be examined to insure that the Skeleton Executive will not overlay the start of the mainline programs as defined in the *DFINE card and the MANLIN card in the System Symbol Table.

Example

```
SKELTN   07600   00048 LOADED
*LDCOR MIC             07600 to 10279
*LDCOR PSC             10280 to 10879
*LDCOR ADC             10880 to 11829
*LDCOR SAC             11830 to 12229
*LDCOR SIOC            12330 to 15129
*LDCOR AOC
PROGRAM NOT LOADED
*DKSTR 0133
END OF JOB
```

6. Compile and load logging routine (if used).
7. Assemble PSC and SIOC DISK. This must be done after the Skeleton Executive is loaded.
8. Compile all FORTRAN-written mainline programs using 1710 FORTRAN II-D. The mainlines may be loaded to disk using an *LDISK Control Record. (Include the Exception Core Load provided).

Example

```
##JOB
##FOR
*LDISKMAINI
        .
        .
        END
```

9. Compile all FORTRAN-written subprograms.

Example

```
##JOB
##FOR
*LDISK
        SUBROUTINE NAME (A,B)
        .
        .
        END
```

10. Compile all FORTRAN-written Interrupt Core Loads.

Example

```
++JOB
++FOR
*INTER
*LDISKPRINI
        .
        .
        CALL RTMIEC
        .
        .
        END
```

11. Compile all Interrupt In-Core routines.

Example

```
++JOB
++ FOR
*LDISKINCOR
*INTER
        SUBROUTINE INCOR
        .
        .
        RETURN
        .
        .
        END
```

NOTE: ALL LOCALS that service interrupts are considered to be interrupt in-core routines. If any recorded interrupts are used in the system, all in-core interrupt routines must also be compiled as interrupt core loads.

12. Prepare the Mainline Core Loads.

Example

```
++ JOB
++ DUP
*DLOADSIOCML          0302            M S 3035013010
*FORLDMAIN2           1
*LIBCRLOGF, EXPF, DKIO, SINCOS, ATANF, SQRTF
*INTCRPI122/50, PI123/51, P5HRI/43
*LOCAL MAIN2, LOCS1, LOCS2
*CCEND
```

13. Prepare the Interrupt Core Loads.

Example

```
++DUP
++JOB
*DLOADHRINT           0406            I S44
*FORLDHOUR            1
*CCEND
```

14. If any of the Executive Programs (MIC, PSC, ADC, SAC, SIOC or AOC) are reassemblec the user must reload the Skeleton Executive as in Step 5 and 6.
15. Prepare a "cold start" card as indicated below. XXX is the dim number of the first mainline program to be executed. (If the units position of XXX is flagged, the mainline specified will be entered in the masked mode).

Example

```
3400034007013600034007024919400XXX11735700619400
```

16. Each Mainline Core Load must execute a PSC call to enter the next mainline core load.

OFF-LINE STARTING PROCEDURE

To begin operation of the off-line Monitor System, the Supervisor program must be loaded into core storage from disk storage. This can be accomplished by entering from the typewriter or a card, the following instructions:

Core Storage Address of Instruction	Instruction
00000	34 00032 00701
00012	36 00032 00702
00024	49 02402
00031	X
00032	Y1963611300102

X identifies the form of input for the Monitor Control records; 1 for typewriter, 3 for paper tape, and 5 for cards. Y specifies the disk drive code (1, 3, 5, 7) identifying the disk module where the system is loaded.

When the Release and Start key or Load key is depressed, the Supervisor program is read into core storage and the first Monitor Control card is read in under control of the Supervisor program by the Monitor Control Record Analyzer routine.

The Parity, I/O and Overflow Program switches should be on Program to operate the off-line Monitor.

ON-LINE STARTING PROCEDURE

After all data is loaded, the Executive System can be started by the following procedure.

Depress the Reset key on the computer console and enter the following data from any available input unit:

```
34 00034 00701
36 00034 00702
49 19400 III
11735700619400
```

The last line contains the disk address, sector count, and core address of the Skeleton Executive loader.

These instructions will cause the Skeleton Exective to be loaded to core storage and the Core Load specified by III to be loaded by PSC and executed. If III is flagged in the units position, the core load is entered in the masked mode.

If console switch 1 is off, all interrupt indicators will be turned off. If console switch 1 is on, all interrupt indicators that are on will be left on.

APPENDIX E

Error Message Listing

Message

NOTE: X's represent variable characters

APPENDIX F

Permanent System Symbol Table Listing

Symbol	Address	Symbol	Address	Symbol	Address
9CCYL0	02111	. FIXI	04124	. RSGN	03496
9CCYL1	02113	. FLOAT	03532	. SAVE	03712
9CCYL2	02115	. FMFAC	03210	. SIX	02337
9CCYL3	02117	. FMP	03448	. SLASH	06588
9RCYL0	00513	. FSB	03412	. SWC	06078
9RCYL1	00515	. FSBR	03436	. TAN6	02346
9RCYL2	00517	. FTYPE	04628	. TEN34	02356
9RCYL3	00519	. FX1	03913	. TOFAC	03198
ATANF	02996	. FX9	02380	. TRACE	03222
COSF	02986	. FXA	03400	. TWOPI	02301
DKARAY	02976	. FXD	03376	. TWOZ	02291
DKEND	02981	. FXDR	03388	. WACD	04412
DKLIST	02971	. FXM	03364	. WAPT	04388
ENTSC1	02946	. FXS	03340	. WATY	04364
ENTSC2	02951	. FXSR	03352	. XTYPE	04224
ENTSC3	02956	. FXZ	03903	. ZERO	03893
EXPF	02941	. HTYPE	04676	. ZRFAC	02638
FETCH	02966	. INTER	02921		
LOGF	02936	. ITYPE	04508		
RECORD	02961	. K	03085		
SINF	02991	. LN2	02239		
SQRTF	03001	. LN4	02250		
. 9SCPF	02388	. LN8	02261		
. 9SPF	02226	. LN10	02272		
. ABSF	03484	. LOGE	02282		
. ATYPE	04604	. MATRX	06430		
. BETA	03750	. ONEZ	02364		
. CMPLT	06808	. OVLAI	02469		
. ETYPE	04652	. PAR	03290		
. F	03083	. PI	02311		
. FAC	03639	. PIOV2	02321		
. FAD	03424	. PIOV4	02329		
. FAXB	04244	. RACD	04484		
. FAXI	04148	. RAPT	04460		
. FDV	03460	. RATY	04436		
. FDVR	03472	. REDO	06828		
. FINDI	02898	. REP	04076		
. FIX	03520	. REP3	06928		

INDEX

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601