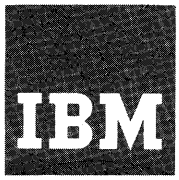
File 1130-36
Form C26-3750-0



Systems Reference Library

IBM 1130 Disk Monitor System Reference Manual

This publication describes the 1130 Disk Monitor, a combined operating and programming system. This system includes a Supervisor program, a Disk Utility program, a symbolic assembler, a FORTRAN compiler, and a subroutine library. The latter four programs operate under control of the Supervisor program to provide continuous operation.

The IBM-supplied subroutine library contains routines for input/output, conversion, and arithmetic functions.

FOREWORD

The IBM 1130 Disk Monitor System, a collective name for five distinct but interdependent programs - Supervisor, Disk Utility, assembler, FORTRAN, and subroutine library - is a powerful, combined operating and programming system.

The programs that make up the Monitor System use advanced programming techniques, including relocatable subroutines, highly compressed formats for data and programs, and flexible input and output command structures which facilitate data conversion operations. A unique feature of the 1130 Monitor System is the "floating" boundary between the user program/data file area and the disk Working Storage area. As information is added to disk storage in the User area, the Working Storage area is decreased in size. Conversely, if a program or data file is deleted from disk storage User area, the remaining programs are packed, and the disk Working Storage area is increased in size.

The following publications may assist the user in utilizing the system:

- IBM 1130 Functional Characteristics, Form A26-5881
- IBM 1130 FORTRAN Language, Form C26-5933
- IBM 1130 Assembler Language, Form C26-5927
- IBM 1130 Subroutine Library, Form C26-5929

Throughout this publication all references to locations in storage are in hexadecimal unless otherwise noted; therefore, the subscript 16 has been omitted.

Machine Requirements

The minimum machine features and units required for operation of the Monitor System are:

- IBM 1131 Central Processing Unit, Model 2, with a minimum of 4096 words of core storage
- IBM 1134 Paper Tape Reader and an IBM 1055 Paper Tape Punch, or an IBM 1442 Card Read Punch.

This publication (C26-3750-0) supersedes IBM 1130 Monitor System Specifications (Form C26-5940-0), which is now obsolete.

Copies of this and other IBM publications can be obtained through IBM Branch Offices. A form has been provided at the back of this publication for reader's comments. If the form has been detached, comments may be directed to: IBM, Programming Publications Dept. 452, San Jose, Calif. 95114

CONTENTS

IBM 1130 DISK MONITOR SYSTEM - INTRODUCTION 1

DISK STORAGE LAYOUT 3
- IBM Systems Area 3
- User Storage Area 4
- Working Storage Area 5
- File Protection 5

SUPERVISOR PROGRAM 7
- Skeleton Supervisor 7
- Monitor Control Record Analyzer 7
- Monitor Control Records 7
- Supervisor Control Records 10
- Stacked Input Arrangement 12
- The Loader 12

DISK UTILITY PROGRAM (DUP) 17
- DUP Control Records 17
- DUP Messages 24
- DUP Operating Notes 25

ASSEMBLER 26
- Assembler Control Records 26
- Origin of Source Program 28
- Assembler Paper Tape Format 28
- Assembler Messages and Error Codes 29
- Assembler Operating Procedures 29

FORTRAN COMPILER 32
- FORTRAN Control Records 32
- FORTRAN Printouts 34

SUBROUTINE LIBRARY 36
- Pre-Operative Errors 36
- Card Subroutine (CARD0 and CARD1) Errors 36
- Console Printer Subroutine (TYPE0 and WRTY0) Errors 38
- Keyboard Subroutine (TYPE0) Functions 38
- Paper Tape Subroutines (PAPT) 39
- Adding and Removing Subroutines 39

SYSTEM GENERATION OPERATING PROCEDURES (CARD SYSTEM) 40
- Disk Pack Initialization Routine (DPIR) 40
- User-Supplied Cards 41
- Procedure for Initializing Disk Monitor System from Cards 43
- Cold Start Operating Procedure 44

PAPER TAPE MONITOR SYSTEM 45
- DPIR Paper Tape Load Operating Procedures 45
- Procedure for Initializing Disk Monitor System from Paper Tape 45
- Cold Start Operating Procedure 46
- Paper Tape Control Records 46

APPENDIX A. ERROR MESSAGES 47

APPENDIX B. DATA FORMATS 57
- Disk System Format (DSF) 57
- Disk Core Image Format (DCI) 59
- Disk Data Format (DDF) 59
- Card System Format (CDS) 59
- Card Data Format (CDD) 61
- Print Data Format (PRD) 61
- Paper Tape System (PTS) and Paper Tape Data (PTD) Formats 61

APPENDIX C. DISK STORAGE UNIT CONVERSION FACTORS 63

APPENDIX D. SUPERVISOR AND DUP INPUT/OUTPUT CHARACTER CODES 64

APPENDIX E. 1130 SUBROUTINE LIBRARY LISTING 65

APPENDIX F. IN-CORE COMMUNICATIONS AREA (COMMA) 70

APPENDIX G. LAYOUT OF LET/FLET ENTRIES 74
- Three-Word Entries 74
- Six-Word Entries 74

APPENDIX H. IBM00 (1130 DISK MONITOR SYSTEM MAINTENANCE PROGRAM) 75
- System Program Maintenance 75
- IBM Subroutine Library Maintenance 76
- Operating Procedures 77
- Error Messages 77

APPENDIX I. UTILITY ROUTINES 78
- Console Printer Core Dump 78
- 1132 Printer Core Dump 78

APPENDIX J. SAMPLE PROGRAM OUTPUT 79

INDEX 84

The 1130 Monitor is a disk-oriented system that allows the user to assemble, compile, and/or execute individual programs or a group of programs with a minimum of operator intervention. Jobs to be performed are stacked and separated by control records that identify the operation to be performed.

The Monitor System consists of five distinct but interdependent programs (see Figure 1):

- Supervisor program
- Disk Utility Program
- Assembler program
- FORTRAN compiler
- Subroutine library

The Supervisor program provides the necessary control for the stacked-job concept. It reads and analyzes the monitor control records, and transfers control to the proper program.

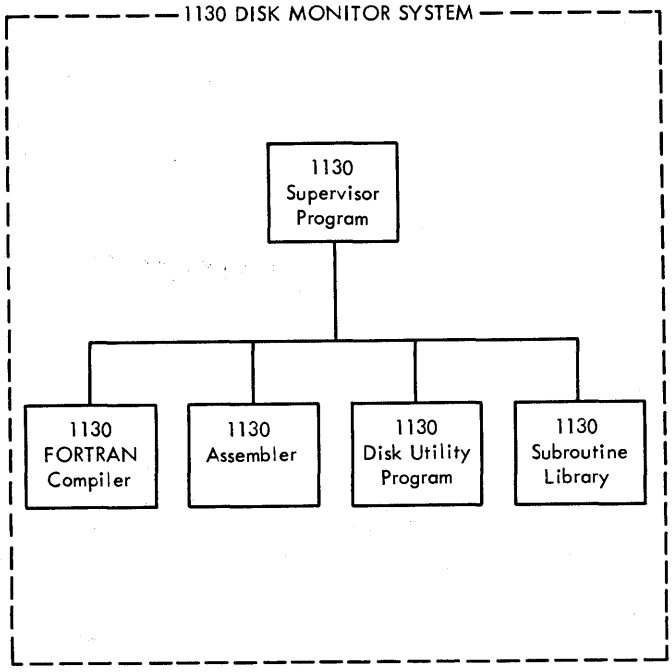


Figure 1. 1130 Disk Monitor System

The Disk Utility Program is a group of routines designed to assist the user in storing information (data and programs) on the disk, and retrieving and using the information stored.

The assembler program converts user-written symbolic-language source programs into machine-language object programs.

The FORTRAN compiler converts user-written FORTRAN-language source programs into machine-language object programs.

The subroutine library contains subroutines for data input/output, data conversion, and arithmetic functions.

The Monitor System coordinates program operations by establishing a communications area in memory which is used by the various programs that make up the Monitor System. It also guides the transfer of control between the various monitor programs and the user's programs. Operation is continuous and setup time is reduced to a minimum, thereby effecting a substantial time saving and allowing greater programming flexibility. The complete Monitor System resides on disk storage. Only those routines or programs required at any one time are transferred to core storage for execution. This feature minimizes the core storage requirements and permits segmenting of long programs.

In addition to providing the user with minimal operating time, the 1130 Disk Monitor System significantly reduces the amount of programming to be done by the user. This is made possible through the sharing of common subroutines by unrelated programs. For example, input/output or conversion operations are required by most user programs, regardless of whether the programs are written in the assembler language or in FORTRAN. IBM provides a library of subroutines as an integral part of the Monitor System.

The assembler and FORTRAN compiler facilitate development of a library of user programs. The object programs can be stored on cards or paper tape, as is customary in installations without disk storage. However, with disk storage, programs can be stored directly on disk without the necessity of designating actual storage locations, remembering or documenting the storage assignments, or updating the storage assignments and documentation as conditions change. The disk-stored programs

and data are referred to by name when called for use. The Monitor System, through the use of a table known as the Location Equivalence Table (LET), can locate any user program, subroutine, or file by a table search for the name. Stored with the name is the amount of disk storage (in disk blocks)* required by the program or data.

Any program that is added to the user's disk-stored programs is usually placed at the end of the other programs. If a program is deleted, the remaining programs are usually packed for effective utilization of disk storage. This packing facility is described later in this publication.

*There are 16 disk blocks per sector; each disk block contains 20 data words (refer to Appendix C).

Disk Storage is divided into three logical areas: IBM Systems area, User Storage area, and Working Storage area. The contents of these three areas, described in detail in subsequent paragraphs, are shown in Figure 2. The sectors and cylinders that these areas occupy are shown in Table 1. Appendix C shows disk storage unit conversion factors.

IBM SYSTEMS AREA

This area contains the integral parts of the 1130 Disk Monitor System: the Supervisor program, Disk Utility Program (DUP), FORTRAN compiler, and the assembler. The FORTRAN compiler and/or the assembler can be deleted at the user's option.

Supervisor Program

This program supervises all monitor operations and performs the control and loading functions for the user programs and monitor programs (FORTRAN, Assembler, and DUP). The Supervisor is directed by monitor control records in the stacked input. Included within the Supervisor is the Skeleton Supervisor, which resides in core during monitor operation and provides the communications link between the monitor programs and the user's programs.

Table 1. Disk Storage Allocation

Use	Total Sectors	Sectors Occupied	Cylinders Occupied	Total Cylinders
Identification	1	0	0	1/8
Cold Start	1	1	0	1/8
DCOM	1	8	0	1/8
Supervisor	53	2-7, 9-55	0-6	6-5/8
DUP	72	56-127	7-15	9
FORTRAN	104	128-231	16-28	13
Assembler	40	232-271	29-33	5
CIB	24	272-295	34-36	3
LET	8	296-303	37	1
Remainder of User Area and Working Storage	1296	304-1599	38-199	162
	1600			200

Disk Utility Program (DUP)

DUP is a group of routines provided by IBM that aid the user in the day-to-day operation of his installation. By means of these routines, certain frequently required operations, such as storing, deleting, and dumping data and/or programs from disk storage, can be performed with minimum programming effort by the user.

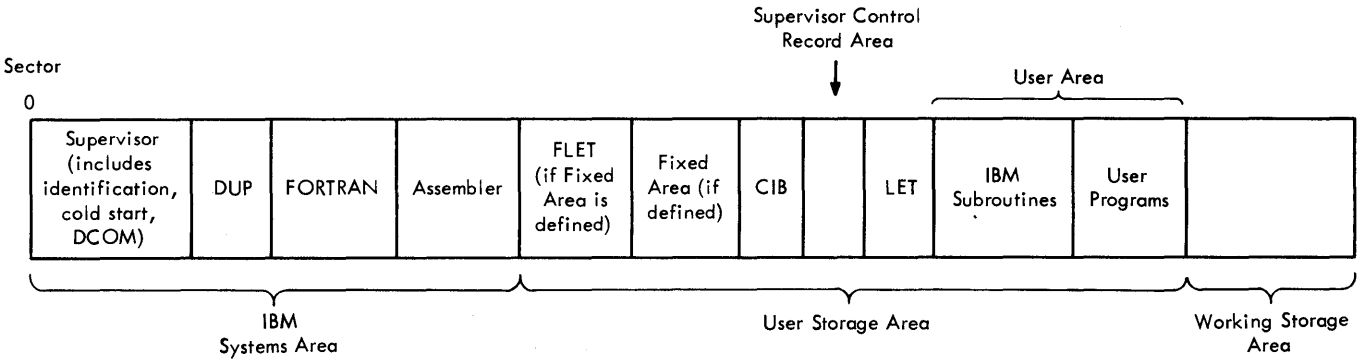


Figure 2. Disk Storage Layout

FORTRAN Compiler

The compiler translates programs written in the FORTRAN language into machine language, and provides for calling the necessary arithmetic, functional, conversion, and input/output subroutines at execution time.

Assembler

The assembler converts source programs written in the assembler language into machine language object programs. The conversion is one-for-one, that is, the assembler normally produces one machine language instruction for each instruction of the source program.

USER STORAGE AREA

This area consists of the following:

- A Fixed area (optional) for storing core image programs and data. If a Fixed area is defined, there will also be a Fixed Location Equivalence Table (FLET).
- The Core Image Buffer (CIB)
- The Supervisor Control Record Area
- The Location Equivalence Table (LET)
- A User area for storing IBM-supplied subroutines, user-written programs, and data files

Core Image Buffer (CIB)

Those parts of a core load (main program and associated subprograms) that fall below core location 4096_{10} are put in the Core Image Buffer (CIB) as they are prepared for execution or for storing in core image format by the Loader (refer to DUP Control Records, *STORECI). When all parts of program have been processed, either the contents of the CIB are read back into core storage by the Loader, which overlays itself in the process, or DUP is recalled from disk to core to complete the *STORECI operation, using the CIB as source of any parts of the core load which are to reside below core location 4096_{10}.

The CIB is also used by the Supervisor to save core locations 256_{10}-409_{10} on every CALL LINK. Before each link is executed, the Loader restores any part of this area which has been included in the COMMON defined by the called link.

Supervisor Control Record Area

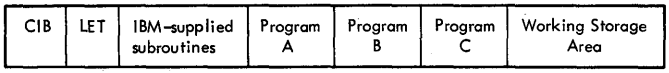
This area is used by the system to store information for use by the Loader (refer to Supervisor Control Records).

Location Equivalence Table (LET)

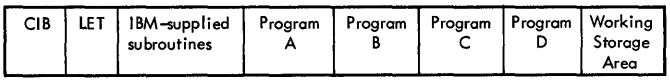
The Location Equivalence Table (LET) serves functionally as a "map" for the IBM subroutines, user's programs, and data files. Each subroutine, user's program, or data file that is stored on disk has at least one entry in the table. The table entry contains the name and disk block length of the subroutine, program, or data file. Each entry point in a subroutine requires a separate entry in LET. The user may print the contents of LET by using the DUP control record DUMPLET (refer to DUP Control Records).

User Area

As each user-written program or data file is added to the User area, the space available for the Working Storage area decreases. Conversely, if a program is deleted, the Working Storage area increases by the amount of space the program formerly occupied in the User area. For example, user-written programs A, B, and C are stored on disk as follows:

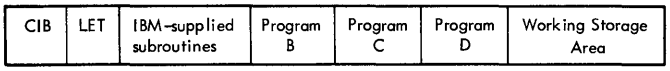


If a program, D, is created, it would be stored on disk causing the Working Storage area to contract:



If Program A is now deleted, Programs B, C, and D would be moved up, maintaining a packed

condition in the User area while expanding the Working Storage area:



NOTE 1: Core Image programs and data files are always put on disk at the beginning of a sector, and remain at the beginning of a sector even after packing. Disk System format programs start at the beginning of a disk block.

NOTE 2: The Working Storage area always starts at the beginning of a sector; therefore, it might not expand or contract by the exact size of the program stored or deleted.

IBM-Supplied Subroutine Library

The IBM-supplied subroutine library contains input/output, data conversion, arithmetic and functional, and selective dump subroutines. These subroutines are generally available for use with both the assembler and the FORTRAN compiler. Operating procedures are described in a subsequent section of this manual. Appendix E contains a complete list of all IBM-supplied subroutines.

Flipper Routine

The subroutine library includes a Flipper routine, which is a part of the core load for those user's programs that use LOCAL (Load-on-Call) routines (refer to Supervisor Control Records). When a LOCAL routine is called, control is passed to the Flipper routine, which reads the LOCAL into core storage if it is not already in core and transfers control to it. All LOCALs in a given core load are executed from the same core storage locations; each LOCAL overlays the previous one. All LOCALs required by a program are relocated and stored by the Loader in Working Storage immediately following the last defined file, if any.

Fixed Area

The Fixed area is an optional area that the user can define to enable him to store programs at fixed disk locations. The user can define the size of the Fixed area to be a whole number of cylinders, with a minimum of two, and he can increment (but not decrease) the size of the Fixed area by a whole number of cylinders at any time. Unlike in the User area, when a program is deleted from the Fixed area no packing occurs. Thus, programs or data files in this area can be referenced by absolute sector addresses, since they will not be moved. The Fixed area, if any has been defined, requires a LET of its own, i.e., a Fixed Location Equivalence Table (FLET). The contents of FLET may also be printed by using the DUP control record DUMPLET. The Fixed area is used only for the storage of core image programs and data, and not for Disk System format programs or for working storage.

WORKING STORAGE AREA

The Working Storage area is used for temporary storage. Most of the area is available to the user during execution of his programs. The Loader stores LOCALs (Load-on-Call routines) and SOCALs (system overlays) in this area, and it is also used extensively by the monitor programs (see Working Storage Indicator Word). For example, the assembler uses this area for temporary storage of a program during the assembly process; at the conclusion of an assembly or compilation, the object program is in the Working Storage area.

The assembler requires 32 sectors of Working Storage for possible symbol table overflow during an assembly, plus whatever additional Working Storage is required for disk output (compressed source statements in Pass 1, object program in Pass 2). Since an assembly requires at least one sector for disk output, the assembler checks for the availability of 33 sectors of Working Storage before beginning to assemble the source program. If at least 33 sectors are not available, an assembler error message is printed (refer to Appendix A), the assembly is terminated, and control is returned to the Supervisor.

During a FORTRAN compilation, FORTRAN requires the amount of Working Storage necessary to contain the compiled program.

FILE PROTECTION

The 1130 Disk Monitor System controls file protection. All Disk I/O subroutines furnished by

IBM check the address of the sector on which they have been instructed to write to ensure that it is greater than the file protection address in COMMA (refer to Appendix F), with the exception of the Write Immediate function (described in IBM 1130 Subroutine Library, Form C26-5929). The file protection address, which is equal to the starting address of Working Storage, is updated by DUP whenever a program is added to the User area.

Only data files which have been created in or moved into Working Storage can be written into by assembly-language programs (unless the Write Immediate function is used). FORTRAN programs may write directly into User and Fixed areas (refer to *FILES under Supervisor Control Records).

The Supervisor program performs the control and loading functions for the Monitor System. Monitor control records, which are used to direct the sequence of jobs without operator intervention, are included in a stacked input arrangement and are processed by the Supervisor program. The Supervisor program decodes the monitor control record and calls the proper monitor program to perform the desired operation. A typical sequence of operations is listed below. The programs in parentheses would be called by the Supervisor to perform the particular operation:

1. Compilation of a FORTRAN program (FORTRAN compiler)
2. Storage of the compiled program on disk (Disk Utility Program)
3. Assembly of a symbolic program (Assembler)
4. Storage of the assembled program on disk (Disk Utility Program)
5. Execution of a disk-stored program (Loader)
6. Punching of a disk-stored program into cards (Disk Utility Program)

The Supervisor itself is a group of several distinct but closely related routines:

- Skeleton Supervisor
- Monitor Control Record Analyzer
- Loader
- Cold Start Routine

SKELETON SUPERVISOR

The Skeleton Supervisor provides the communications link between the monitor programs and the user's programs, i.e., it contains the necessary logic to conduct the transition from one job to another. The Skeleton Supervisor is read into core storage when the operation of the monitor is initially started by means of the Cold Start Routine (refer to Cold Start Operating Procedure), which occupies sector 1. The Disk Communications Area (DCOM), which contains addresses and indicators necessary for the operation of the monitor, is read into core initially with the Skeleton Supervisor. The in-core communications area (referred to as COMMA) is restored from DCOM whenever a Cold Start procedure is initiated or a JOB record is encountered (refer to Monitor Control Records). When COMMA is restored there will be no usable program in Working Storage. Appendix F lists all the core locations and information contained in COMMA.

MONITOR CONTROL RECORD ANALYZER

This routine analyzes the monitor control records, prints out the information contained in the control record, and calls the appropriate program: Disk Utility Program, Assembler, FORTRAN compiler, or Loader.

The following three formats are used by the Monitor System to store information on disk (refer to Appendix B):

- Disk Core Image Format (DCI)
- Disk System Format (DSF)
- Disk Data Format (DDF)

MONITOR CONTROL RECORDS

Input to the Supervisor consists of one or more job decks, each preceded by a JOB monitor control record (see Figure 3). The character codes recognized by the Supervisor are listed in Appendix D. Although the monitor control records are described in terms of cards, these records can be entered in card image form from paper tape or the keyboard/console printer.

The JOB control record defines the starting and ending points of the job; however, the total job can consist of many subjobs. The assembler, FORTRAN compiler, Disk Utility Program, and user's programs can be called for operation by the ASM, FOR, DUP, and XEQ control records,

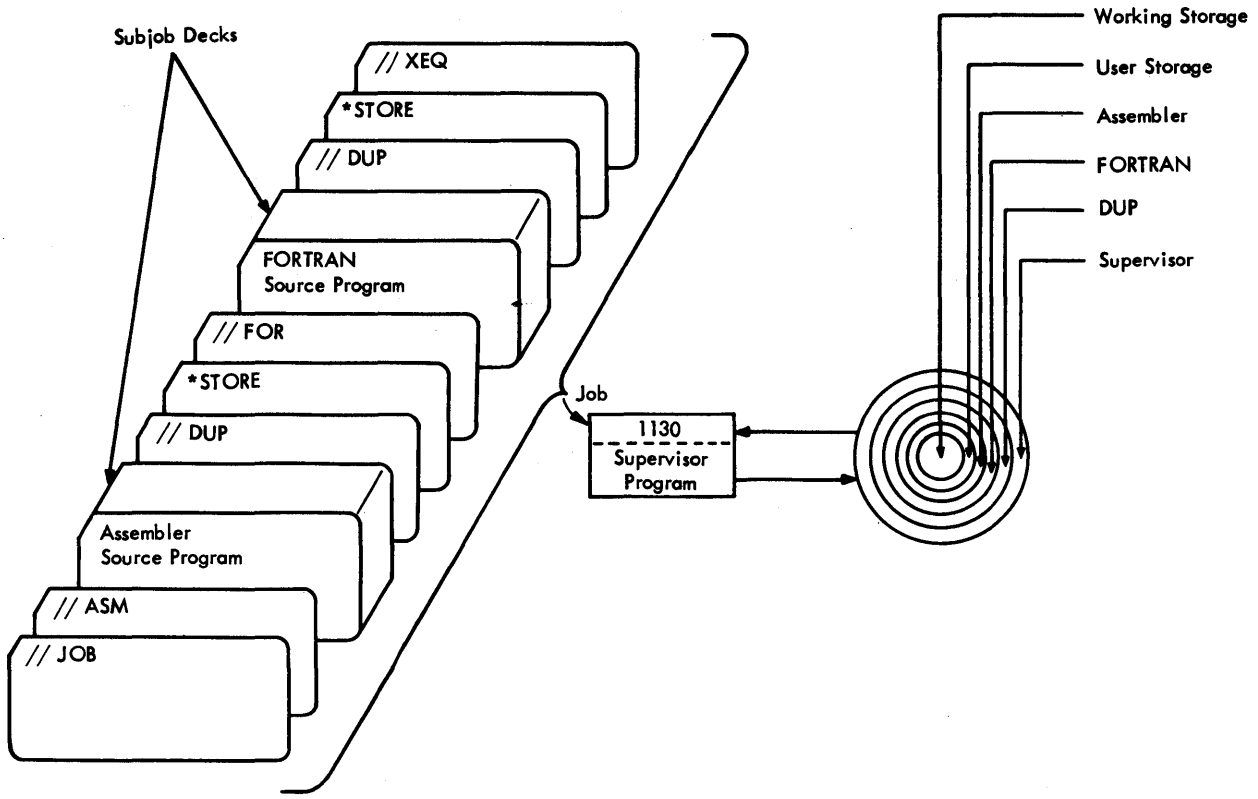


Figure 3. Processing Input Data Under Supervisor Control

respectively. These are each considered individual subjobs. The successful completion of the total job depends on the successful completion of each individual subjob within the job. Some subjobs are not attempted if the preceding subjobs have not been successfully completed.

When a monitor control record is read, the system program required to do the subjob is read into core storage from disk storage. The program then processes input until the end of the subjob deck is reached, a new monitor control record is encountered, or an error occurs. Monitor error messages are described in Appendix A.

Every job is assumed to begin with no programs in Working Storage (see Working Storage Indicator Word).

Control can be returned to the Supervisor by manually branching to core location 0038. The Supervisor then passes records until it encounters a monitor control record.

All monitor control records have the following format:

Columns 1-2: // (slashes, to identify monitor control record)
3: b (blank)
4-7: Pseudo-operation code (left-justified)

The following paragraphs contain a list of the codes and their operations. The monitor control records are summarized in Table 2.

NOTE: Comments are permitted in unspecified columns in all monitor control records. A "b" appearing in a column means that the column must be blank.

Table 2. Summary of Monitor Control Records

CC 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	
/ / b * comments	
/ / b J O B T —— Disk Storage Label ——	Initialize a job sequence
/ / b A S M	Read assembler into core for execution
/ / b F O R	Read FORTRAN into core for execution
/ / b P A U S	Halt until START is pressed
/ / b T Y P	Change control record input from principal input unit to keyboard/console printer for succeeding monitor control records
/ / b T E N D	Change input mode from keyboard/console printer back to the principal unit for succeeding monitor control records
/ / b D U P	Read DUP into core for execution
/ / b X E Q —Program Name — L —Count— Disk 0, 1, N	Read and transfer control to mainline program

JOB

This record causes initialization and termination of a job sequence and restores COMMA from DCOM. The format is

cc	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	/	/	b	J	O	B	b	T			I	D	E	N	T	b

The letter T in column 8 indicates temporary mode. In this mode, programs or data files stored in the User area by DUP are automatically deleted at the end of the current job. DUP operations which are permitted in temporary mode are described in Table 5.

If columns 11-15 contain a disk storage identification, this identification is compared with that which is written on the first sector of the disk cartridge to determine that the desired cartridge is mounted. If the identification is not the same, the Supervisor waits for operator intervention. The identifier must be left-justified in its field.

This record also causes a skip to channel 1 before it is printed on an 1132 Printer.

ASM

This record causes the assembler to be read into core storage for execution. The format is

cc	1	2	3	4	5	6
	/	/	b	A	S	M

The assembler control records and source statements for the program to be assembled must follow the ASM control record.

FOR

This record causes the FORTRAN compiler to be read into core storage for execution. The format is

cc	1	2	3	4	5	6
	/	/	b	F	O	R

The FORTRAN control records and source statements for the program to be compiled must follow the FOR control record.

PAUS

This record causes a wait to allow the operator to make setup changes. The format is

cc	1	2	3	4	5	6	7
	/	/	b	P	A	U	S

The monitor operation proceeds as soon as PROGRAM START is pressed.

TYP

This record changes the control record input from the principal input unit to the keyboard/console

printer for succeeding monitor (only) control records. The format is

```
cc 1 2 3 4 5 6
   / / b T Y P
```

TEND

This record changes the input mode from the keyboard/console printer back to the principal input unit for succeeding monitor control records. The format is

```
cc 1 2 3 4 5 6 7
   / / b T E N D
```

DUP

This record causes the Disk Utility Program to be read into core storage for execution. The format is

```
cc 1 2 3 4 5 6
   / / b D U P
```

Control records for the Disk Utility Program must follow the DUP control record.

XEQ

This record causes the Loader to load a specified mainline program into core storage and to transfer control to it. The format is

```
cc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
   / / b X E Q b X X X X X b L     Y Y     Z b
```

The mainline program XXXXX must be left-justified in columns 8-12. If XXXXX is in Disk System format, the Loader converts it to Core Image format. If columns 8-12 are blank, the mainline program presently stored in Working Storage (by FORTRAN, DUP, or the assembler) is converted and read into core and executed.

A core map is printed during conversion if column 14 contains an L and the program is in Disk System format. This map includes the core loading address of the mainline program, the names and execution addresses of all subroutines and subprograms included in the load, file allocations, if any, giving file number, sector address, and number of sectors in the file. Also, if L is specified, a core map is printed for any DSF program linked to under this execution.

Columns 16-17 must contain the count of LOCAL, NOCAL, and FILES records which follow, if any (refer to Supervisor Control Records). This count is decimal, right-justified.

DISK 0, 1, or N will be loaded with the program if column 19 contains 0, 1, or N, respectively. Any other character (including blanks) causes a special, shorter disk routine (DISKZ) to be loaded. This special version is intended for FORTRAN programs and those assembly-language programs which do not use the disk.

Comments

This record provides comments in the listing. It may not immediately follow an XEQ, DUP, ASM, or FOR record. The format is

```
cc 1 2 3 4 5 - 80
   / / b * comments
```

SUPERVISOR CONTROL RECORDS

LOCAL

LOCAL is an acronym denoting routines specified by the user to be loaded into a LOCAL overlay area as they are called. All subroutines desired by the user to be loaded on call at execution time must be designated by LOCAL records following the XEQ monitor control record. The format is as follows:

```
cc 1
   *LOCALML1,SUB1,SUB2
```

where ML1 = name of a mainline program to be executed, and SUB1 and SUB2 are subroutines in the mainline program.

Each mainline program (in the same XEQ subjob) that calls a subroutine to be loaded on call must have its own LOCAL record. The same mainline program may have more than one LOCAL record.

For example:

```
*LOCALML1,SUB1,SUB2
*LOCALML2,SUB3,SUB4
*LOCALML1,SUB5
```

or

```
*LOCALML1,SUB1,SUB2,SUB5
*LOCALML2,SUB3,SUB4
```

If the record ends with a comma, the next record is treated as a continuation. The mainline name is not repeated in a continuation, e.g.,

```
*LOCALML1,SUB1,SUB2,
*LOCALSUB5
```

If the mainline program is executed from Working Storage, the mainline name must be omitted by putting a comma in column 7, e.g.,

```
*LOCAL,SUB1,SUB2
```

No embedded blanks are allowed in a LOCAL record.

NOCAL

NOCAL is an acronym denoting routines which, although not called anywhere in the core load, are to be included in the load, e.g., debugging aids such as trace or dump routines.

All subroutines which are to be loaded but are not called at execution time must be designated by NOCAL records following the XEQ monitor control record. The format is as follows:

```
cc 1
   *NOCALML1,SUB1,SUB2
```

NOCAL records are governed by the same rules and restrictions as LOCAL records.

NOTE: The user must observe the following rules in LOCAL and NOCAL records:

1. No routine can appear in a LOCAL record if it causes any of the other routines appearing in LOCAL records (for the same mainline program) to be called before the first LOCAL has returned control to the calling routine. Thus, a LOCAL cannot call another LOCAL, nor can it call a routine which causes a second LOCAL to be read into core and executed. For example, if A calls B and B calls C, and A is a LOCAL, then neither B nor C can appear on a LOCAL record for the same mainline program.
2. If a given routine is designated a LOCAL, and the System Overlay scheme is employed, then this routine will be a LOCAL even though it might have been included in one of the System Overlays (SOCALs).
3. No program which uses LOCALs or NOCALs can be stored in Disk Core Image Format (DCI).
4. Only CALL-type subroutines, i.e., type 4 and type 6 subroutines, can appear on NOCAL records (see Appendix B).
5. If a subroutine is designated a LOCAL, it will be loaded as a LOCAL even if it is not referenced anywhere in the core load.
6. The LOCAL information pertaining to any given XEQ record cannot exceed 640 words, counting all LOCAL names on the LOCAL records as two words and mainline program names as three words. The same rule applies to NOCAL information.
7. Only types 3, 4, 5, and 6 subroutines can appear on LOCAL records (see Appendix B).

FILES

File numbers specified in FORTRAN DEFINE FILE statements can be equated to names of data files in the User area or Fixed area at execution time by means of a FILES record entered after an XEQ monitor control record or DUP control record STORECI. The format is as follows:

```
cc 1
   *FILES(FILEN,NAMEN),(FILEM,NAMEM)
```

where FILEN and FILEM are the file numbers specified in FORTRAN DEFINE FILE statements, and NAMEN and NAMEM are names of previously defined disk storage data files.

No embedded blanks are allowed. If the record ends with a comma, the next record is treated as a continuation, e.g.,

```
*FILES(FILEN,NAMEN),
*FILES(FILEM,NAMEM)
```

NOTE: The FILES information for a given XEQ record cannot exceed 640 words, counting the file numbers as one word and the file names as two words.

Any number of LOCAL, NOCAL, and FILES records can follow the XEQ monitor control record, but each type must be grouped together, e.g.,

```
*LOCAL
*LOCAL
*LOCAL
*NOCAL
*NOCAL
*FILES
*FILES
```

The following is not permitted:

```
*LOCAL
*NOCAL
*LOCAL
*FILES
*NOCAL
```

STACKED INPUT ARRANGEMENT

Input to the Monitor System consists of control records, source programs, object programs, and data arranged logically by job.

The following points must be considered when arranging the input for any job.

1. Any number of comments records can be inserted in front of (but not immediately following) DUP, ASM, FOR, or XEQ monitor control records.
2. Any records other than monitor control records which remain after the execution of an ASM, FOR, or XEQ subjob are passed until the next monitor control record is read. After a DUP operation, records are passed until either a monitor control record or another DUP control record is read.
3. If an error is detected in an assembly, FORTRAN compilation, or during loading from Disk System format, the resulting object program or any programs that follow within the job cannot be executed. Also, if an error is detected in an assembly, FORTRAN compilation, or during a loading from Disk System format during a STORECI function, all DUP functions are bypassed until the next valid ASM, FOR, or JOB record is read.
4. If the FORTRAN compiler or the assembler encounters a monitor control record, control will be transferred to the Supervisor, i.e., the monitor control record will be trapped. The Supervisor will correctly analyze the record after the compilation or assembly has been aborted. DUP will not trap a monitor control record during a DUP operation (refer to DUP Control Records).

The stacked input arrangement shown in Figure 4 will compile, store, and execute both Programs A and B, providing there are no source program errors, and there is sufficient room in the Working Storage area (refer to Working Storage Area). A source program error causes the DUP STORE operation (refer to DUP Control Records) to be bypassed for that program, and all following XEQ requests preceding the next JOB record are disregarded. Thus, if the successful execution of one program depends upon the successful completion of the previous program, both programs should be considered as one job and the XEQ control records should not be separated by a JOB record.

Figure 5 shows the stacked input arrangement for three jobs which are not dependent upon each other.

Job A assembles, stores, and executes source program A. This job includes comments cards and a PAUS monitor control card to allow the operator to intervene.

Job B calls in the Disk Utility Program, and stores object program B on disk.

Job C compiles, stores, and executes FORTRAN source program C.

THE LOADER

The Loader has two basic functions:

1. To prepare entire core loads (Disk System format loading).
2. To bring core loads into core storage immediately before execution (Core Image format loading). This includes the restoration of COMMON, if any, between linked core loads.

These two loading processes are described after a discussion of the origin which the Loader gives a particular core load, the object-time transfer vector, and system overlays (SOCALs). Disk System format and Core Image format are described in Appendix B.

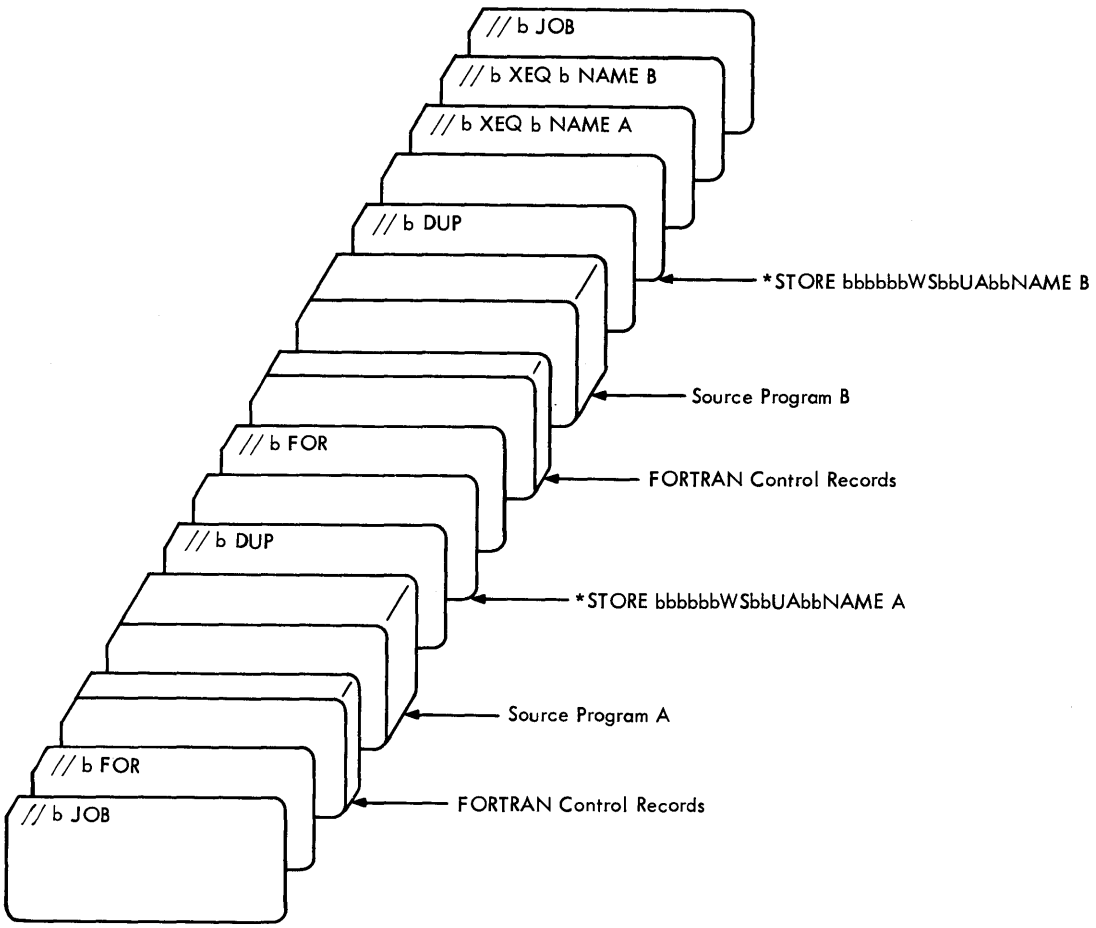


Figure 4. Example of Stacked Input (One Job)

Origins for Core Loads

The Loader origins relocatable mainlines (main programs) after the Disk I/O subroutine requested by the user on the XEQ control record. One of these disk routines is always in lower core, and no disk routine is included in any disk-stored core load. DISKZ is always used unless otherwise specified. The origins used by the Loader are shown below:

Disk I/O Version	Origin (hexadecimal)	(decimal)
DISKZ	1C2	450
DISK0	260	608
DISK1	370	880
DISKN	438	1080

The origins for absolute mainlines are not controlled by the Loader; however, such mainlines must be originated above the end of the Disk I/O version used. All references in a core load to a Disk I/O subroutine must be to the same one.

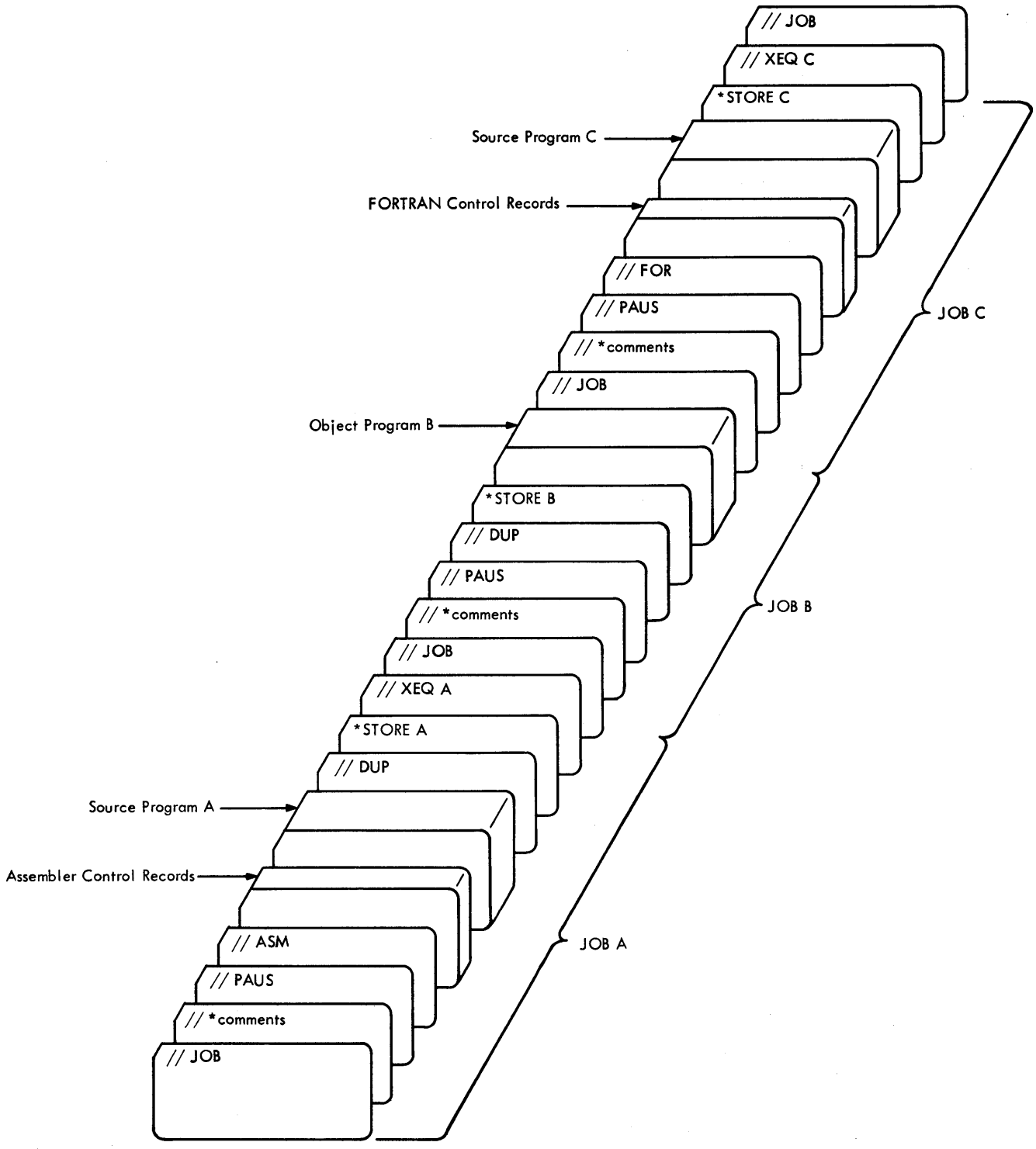


Figure 5. Example of Stacked Input (Three Jobs)

Object-time Transfer Vector

In order to transfer to and from subroutines at execution time, the Loader builds two separate object-time transfer vectors: the CALL TV and the LIBF TV (see Figure 6).

Each CALL TV entry is a single word containing the absolute address of a subprogram entry point; however, in the case of a LOCAL subprogram referenced by a CALL statement, the absolute address is the address of the corresponding Flipper Table entry instead of the subprogram entry point.

Each LIBF TV entry consists of three words. Word one is the link word. Words two and three contain a branch instruction to the subprogram entry point; however, in the case of a LOCAL subprogram referenced by an LIBF statement, words two and three contain a branch instruction to the corresponding Flipper Table entry instead of the subprogram entry point.

The first two LIBF TV entries are special entries, each three words long. The first entry is the Floating Accumulator (FAC). The address of the first word of FAC must be odd; therefore, if necessary, a dummy entry is made in the CALL TV to make FAC begin at an odd address. The second special entry is used by certain subroutines to indicate overflow, underflow, and divide check.

If SOCALs are used, the LIBF TV contains special entries for SOCAL subprograms referenced by LIBF statements. These entries transfer indirectly either to the referenced subprogram if the overlay containing the subprogram is presently loaded, or to the SOCAL Flipper in order to load the required overlay and transfer to the referenced subprogram (refer to System Overlays).

The CALL TV does not contain entries for SOCAL subprograms referenced by CALL statements if SOCALs are used.

System Overlays (SOCALs)

System Overlays (SOCALs) are created for any core load with a FORTRAN mainline program if the core load will not fit into core. The Loader selects certain subroutines used in the core load and writes them into Working Storage in either two or three groups (overlays). An area in core as large as the largest overlay is reserved for these subroutines. Whenever a subroutine in one of these overlays is required during program execution, the corresponding overlay is read from the disk into the overlay area in core (if it is not already in core).

Overlays are constructed from the IBM subroutine library according to type and subtype (described in Appendix B). The user can alter this design by changing the subtypes of the library subroutines, or by specifying a subtype for his own subroutines (refer to DUP Control Records - STORE). The two levels of SOCALs are described in the following paragraphs.

SOCAL Level 1 uses the following two overlays:

1. Type 3, subtype 2 (arithmetics, e.g., FADD), and Type 4, subtype 8 (functionals, e.g., SIN).
2. Type 3, subtype 3 (the non-disk FORTRAN Format subroutine SFIO, and the FORTRAN I/O subroutines, e.g., CARDZ), and Type 3, subtype 4 (the I/O conversion subroutines, e.g., HOLEB).

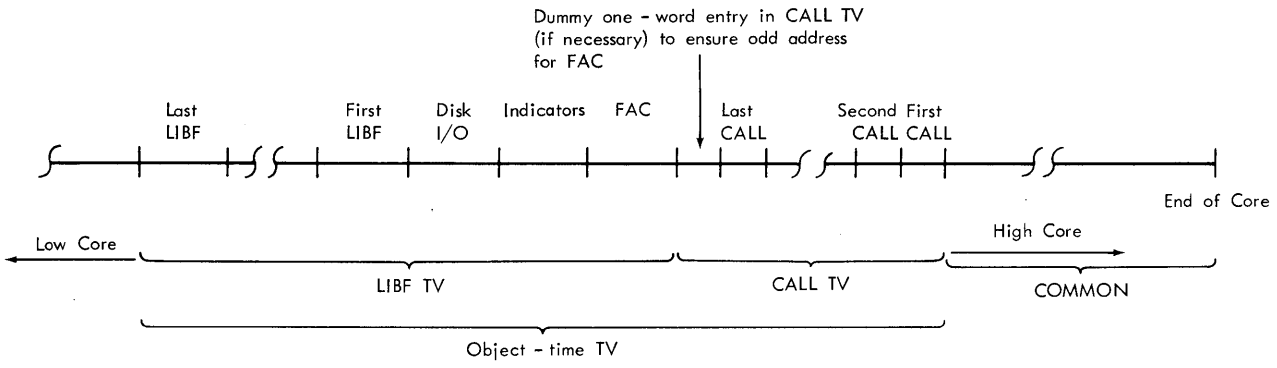


Figure 6. Layout of Object-Time Transfer Vector Area

Level 1 reduces the core requirements by an amount equal to the total size of the smaller of these overlays. Approximately 15 words of extra linkage, however, are required.

If core loads do not fit with Level 1, then Level 2, employing the following three overlays, is used:

1. Same as (1) above.
2. Same as (2) above.
3. Type 3, subtype 1 (disk FORTRAN I/O subroutines SDFND and SDFIO). In addition, this overlay includes a 320-word buffer.

Level 2 reduces the core requirements by an amount equal to the sum of the two smallest overlays, with approximately 15 words of extra linkage added.

The overlays will not contain all available subroutines of the specified types, but only those required by the core load.

Since LOCALs take priority over SOCALs, if a subroutine which would otherwise be in a SOCAL overlay is designated a LOCAL, it will appear as a LOCAL and not as part of a SOCAL.

If a core load does not fit with Level 2 overlays, core requirements may be reduced by additionally designating the following as LOCALs:

1. Subroutines not contained in any overlay.
2. Subroutines contained in the largest overlay. This reduces the SOCAL overlay area required in core.

If the core load does not fit into core even with SOCALs, an error condition is indicated. An error condition is also indicated for core loads which do not fit and which have mainline programs written in assembler language.

Programs requiring system overlays cannot be stored in Core Image format (refer to Appendix B, Disk Core Image Format).

Disk System Format Loading

A core load is built from programs stored in Disk System format in either of two cases:

1. To execute the core load immediately (called as a result of an XEQ control record or a CALL LINK). In this case control must be passed to the Core Image format loading process at the termination of the Disk System format loading process.

2. To store the core load in Disk Core Image format for future execution (called as a result of a DUP *STORECI control record - refer to DUP Control Records). In this case, control is returned to DUP, which initiated the process.

In this type of loading, a mainline program (with its required subroutines) is converted from Disk System format to Core Image format. This includes the construction of the core image header record and the object-time transfer vector. Parts of the core load which are to reside below location 4096_{10} are stored in the CIB; parts of the core load which are to reside above location 4095_{10} (if any) are placed directly into core storage. LOCALs and SOCALs which are a part of the core load are also processed and written out on Working Storage (following the last data file, if any).

Core Image Format Loading

In this loading process, the core load is read into core, except for the first sector. When loading a program immediately following its conversion from Disk System format, only the contents of the CIB are read into core (other parts of the core load are already in core). When loading a program which has previously been stored in Core Image format, both the sections above location 4095_{10}, if any, and below 4096_{10} are read into core. The Skeleton Supervisor is given the information necessary to enable it to read in the first sector of the core load and to move the object-time transfer vector into its location. Control is then passed to the Skeleton Supervisor, and finally to the object program.

The Disk Utility Program (DUP) is a group of routines designed to accomplish the following:

- Allocate disk storage as required by each program or data file to be stored
- Make these programs available in card or paper tape format
- Provide printed status of the User area, Fixed area, and Working Storage area.
- Perform various disk maintenance operations.

The Disk Utility Program is called into operation by a DUP monitor control record. This record may be followed by any number of DUP control records to select the routines desired. The DUP control records are described in subsequent paragraphs. The character codes recognized by DUP are listed in Appendix D.

Working Storage (WS) Indicator Word

A WS Indicator Word in COMMA (0069) contains the disk block count of the program to indicate that a valid program is in Working Storage.

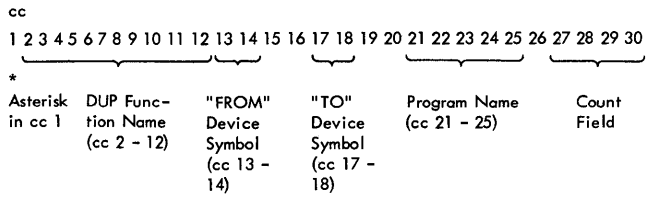
Upon completion of an assembly or compilation, the WS Indicator Word is set to the disk block count of the program left in Working Storage in Disk System format. DUP can then be called upon to store or dump this program.

When a DUP function is used to dump from the User area or the Fixed area, the WS Indicator Word is set to the disk block count of the program being moved. If the DUP function does not destroy any part of the program in Working Storage, the WS Indicator Word is not changed. It is set to zero by a store to the User area, a Cold Start, or a JOB monitor control record.

If a DUP function which involves programs is requested from Working Storage while the WS Indicator Word is zero, a FROM field error message is given and the requested function is bypassed.

DUP CONTROL RECORDS

DUP control records generally have the following format:



All fields except Count Field are left-justified. Each DUP control record contains an asterisk (*) in cc 1. The DUP Function Name is in cc 2 - 12. The "FROM" and "TO" symbols (cc 13 - 14 and 17 - 18, respectively) specify the I/O devices and/or disk areas from and to which data is to be transferred. The following abbreviations must be used in the FROM and TO fields:

Symbol	Meaning
PR	Principal Print Device
CD	Card Reader (if the Disk Monitor System has been loaded from paper tape, CD is equivalent to PT)
PT	Paper Tape
WS	Working Storage, Disk
UA	User area, Disk
FX	Fixed area, Disk

The count field is in decimal, right-justified. For data files, if the source is disk, this field specifies the number of sectors; if the source is cards, this field specifies the number of cards; if the source is paper tape, this field specifies the number of records. Unspecified portions of DUP control records can be used for comments. A "b" appearing in a column indicates that the column must be blank.

In the following paragraphs, each DUP function name is accompanied by a table showing the symbol combinations that may be specified in the FROM and TO fields. The tables also show the various formats that data can be in before the operation, and the corresponding formats to which this data is converted by DUP after the operation. These formats appear in parentheses after the FROM and TO symbols, and have the following meanings:

DSF	Disk System Format
DCI	Disk Core Image Format
DDF	Disk Data Format
CDS	Card System Format
CDD	Card Data Format
PTS	Paper Tape System Format
PTD	Paper Tape Data Format
PRD	Print Data Format

These formats are shown in Appendix B. Table 3 summarizes the DUP functions that move information from one area to another; Table 4 summarizes all DUP control records; and Table 5 gives the restrictions on DUP functions when in temporary mode (JOB T).

DUMP (Dump Program)

The DUMP routine dumps (unloads) information from the User area, Fixed area, or Working Storage area to cards, paper tape, or printer, or from the User or Fixed area to the Working Storage area. The decimal number of disk blocks dumped is specified in the corresponding LET entry or in the WS Indicator Word.

From	To	CD (CDS)	CD (CDD)	PT (PTS)	PT (PTD)	PR (PRD)	WS (DSF)	WS (DDF)
WS (DSF)		X		X		X		
UA (DSF)		X		X		X	X	
UA (DDF)			X		X	X		X
UA (DCI)			X		X	X		X
FX (DCI)			X		X	X		X
FX (DDF)			X		X	X		X

The control record format is as follows:

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
* D U M P b          "FROM"     "TO"      Program Name
                     Symbol     Symbol    (required ex-
                                          cept for WS to
                                          PR)
```

NOTE 1: If the DUMP is from WS, and the WS Indicator is zero, a FROM field error message is given (refer to Appendix A).

NOTE 2: When the DUMP is to cards, each card is checked to see that it is blank before it is punched (refer to Appendix A).

NOTE 3: At the end of DUMP operations, all subsequent blank cards are selected into Stacker 2.

DUMPDATA (Dump Data)

The DUMPDATA routine dumps data from the User area, Fixed area, or Working Storage area to cards, paper tape, or printer, or from the User or Fixed area to the Working Storage area. The number of sectors to dump must be specified by the count field. This number of sectors will be dumped regardless of the length of the specified data file or program.

From	To	CD (CDD)	PT (PTD)	PR (PRD)	WS (DDF)
WS (DSF)		X	X	X	
WS (DDF)		X	X	X	
UA (DSF)		X	X	X	X
UA (DDF)		X	X	X	X
UA (DCI)		X	X	X	X
FX (DCI)		X	X	X	X
FX (DDF)		X	X	X	X

The control record format is as follows:

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
* D U M P D A T A b   "FROM"     "TO"      Program Name     Sector Count
                      Symbol     Symbol    (required ex-    (when dump-
                                           cept for WS to   ing from WS,
                                           PR)              the sector
                                                            count overrides
                                                            the WS Indica-
                                                            tor)
```

NOTE 1: When the dump is to cards, each card is checked to see that it is blank before it is punched (refer to Appendix A).

NOTE 2: At the end of DUMPDATA operations, all subsequent blank cards are selected into Stacker 2.

From / To		UA (DSF)	UA (DDF)	UA (DCI)	FX (DSF)	FX (DDF)	FX (DCI)	WS (DSF)	WS (DDF)	CD (CDD)	CD (CDS)	PT (PTD)	PT (PTS)	PR (PRD)
UA	(DSF)							DUMP**	DUMPDATA**	DUMPDATA**	DUMP**	DUMPDATA**	DUMP**	DUMP** DUMPDATA**
	(DDF)								DUMP** DUMPDATA**	DUMP** DUMPDATA**		DUMP** DUMPDATA**		DUMP** DUMPDATA**
	(DCI)								DUMP** DUMPDATA**	DUMP** DUMPDATA**		DUMP** DUMPDATA**		DUMP** DUMPDATA**
FX	(DCI)								DUMP** DUMPDATA**	DUMP** DUMPDATA**		DUMP** DUMPDATA**		DUMP** DUMPDATA**
	(DDF)								DUMP** DUMPDATA**	DUMP** DUMPDATA**		DUMP** DUMPDATA**		DUMP** DUMPDATA**
WS	(DSF)	STORE* STOREMOD		STORECI*			STORECI			DUMPDATA	DUMP	DUMPDATA	DUMP	DUMP DUMPDATA
	(DDF)		STOREDATA* STOREMOD			STOREDATA STOREMOD				DUMPDATA		DUMPDATA		DUMPDATA
CD	(CDD)		STOREDATA*			STOREDATA**			STOREDATA**					
	(CDS)	STORE*		STORECI			STORECI**	STORE**						
PT	(PTD)		STOREDATA*			STOREDATA**			STOREDATA**					
	(PTS)	STORE*		STORECI*			STORECI**	STORE**						

*Eliminates stored information from Working Storage

**Replaces current contents of Working Storage

Table 3. Movement of Information Using DUP Control Records

Table 4. Summary of DUP Control Records

CC 1 2 3 4 5 6 7 8 9 10 11 12	13 14 15 16	17 18 19 20	21 22 23 24 25	26 27 28 29 30
* D U M P b	—From—	— To —	——— Name ———	
* D U M P D A T A b	—From—	— To —	——— Name ———	—— Sector Count ——
* S T O R E	—From—	— To —	——— Name ———	
* S T O R E C I	– From—	— To —	——— Name ———	—— Count of 'Files'— Records
* S T O R E D A T A	—From—	— To —	——— Name ———	—Sector, Card, or— Record Count
* S T O R E M O D b	— WS —	UA or FX	——— Name ———	
* D U M P L E T	(Print contents of LET on principal printing unit)			
* D W A D R	(Write sector addresses in Working Storage area)			
* D E L E T E			——— Name ———	
* D E F I N E b F I X E D b A R E A				— Cylinder — Count
* D E F I N E b V O I D b A S S E M B L E R				
* D E F I N E b V O I D b F O R T R A N				

Table 5. Restrictions on DUP Functions in Temporary Mode (JOB T)

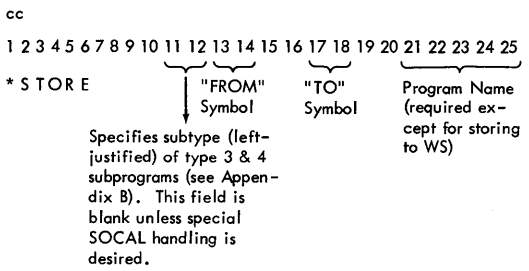
Functions	Restrictions (if any)
DUMP	None
DUMPDATA	None
STORE	None
STORECI	To UA only
STOREDATA	To UA and WS only
STOREMOD	Not allowed
DUMPLET	None
DWADR	Not allowed
DELETE	Not allowed
DEFINE FIXED AREA	Not allowed
DEFINE VOID ASSEMBLER	Not allowed
DEFINE VOID FORTRAN	Not allowed

STORE

The STORE routine stores programs from cards, paper tape, or the Working Storage area to either the User area or Working Storage area on disk.

From / To	UA (DSF)	WS (DSF)
CD (CDS)	X	X
WS (DSF)	X	
PT (PTS)	X	X

The control record format is as follows:



NOTE: If the STORE is from WS, and the WS Indicator is zero, a FROM field error message is given (refer to Appendix A).

STORECI (Store Core Image)

The STORECI routine stores programs from cards, paper tape, or the Working Storage area to either the User area or Fixed area on disk. The programs are converted to Disk Core Image format (see Appendix B), hence they are loaded into core storage faster than programs stored otherwise. The STORECI function uses the Loader to convert the Disk System format program to core image. After control is returned to

DUP, the core image header and that portion of the program (excluding Disk I/O) that resides below core location 4096_{10} are stored from the CIB, and that portion of the program above core location 4095_{10}, if any, is stored from core. No COMMON area is stored, but the transfer vector is included. STORECI always requests a map from the Loader since it will not be available when the program is loaded from Core Image format.

From	To	UA (DCI)	FX (DCI)
CD (CDS)		X	X
WS (DSF)		X	X
PT (PTS)		X	X

The control record format is as follows:

cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
* S T O R E C I — See Note 1 — "FROM" Symbol — "TO" Symbol — Program Name (always required) — See Note 2

NOTE 1: Column 9 is used to specify the Disk I/O routine required by this program.

0 = DISK0
1 = DISK1
N = DISKN
Others = DISKZ
(including blank)

NOTE 2: Count Field (cc 27-30) contains decimal count of *FILES records that are required for program being stored. This number of records will be read before the normal STORECI function is performed.

NOTE 3: Data files named in the *FILES record must be in the Fixed area.

NOTE 4: If the STORECI is from WS, and the WS Indicator is zero, an error message is given (refer to Appendix A).

STOREDATA (Store Data)

The STOREDATA routine stores data from cards, Working Storage area, or paper tape to the User area, Fixed area, or Working Storage area. Each data file starts at the beginning of the next available sector and the length is defined in whole numbers of sectors.

From	To	UA (DDF)	FX (DDF)	WS (DDF)
WS (DDF)		X	X	
CD (CDD)		X	X	X
PT (PTD)		X	X	X

The control record format is as follows:

cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
* S T O R E D A T A — "FROM" Symbol — "TO" Symbol — Program Name (not required for CD to WS or PT to WS) — see NOTE

NOTE: Count Field (cc 27-30) must contain one of the following in decimal:
Sector count if source is WS (overrides the WS Indicator), card count if source is CD, record count if source is PT.

STOREMOD (Store Modify)

The STOREMOD routine moves a program or data from Working Storage to the User area or Fixed area, overlaying an item specified by name in the User area or Fixed area. This permits the user to modify an item in the User or Fixed area without changing its name or relative position. The length of the item in Working Storage (in disk blocks or sectors, as appropriate) cannot be greater than the length of the item it overlays. If the name is not found in LET/FLET, the STOREMOD control record functions as a STORE control record.

From	To	UA	FX
WS		X	X

The control record format is as follows:

cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
* S T O R E M O D b — WS — UA or FX — Program Name

DUMPLET (Dump Location Equivalence Table)

The DUMPLET routine dumps the contents of the Location Equivalence Table (LET) to the principal printing unit (see Figure 7). If a Fixed area has been defined, the Fixed Location Equivalence Table (FLET) is printed as a separate table following LET (see Figure 8).

The control record has the following format:

```
cc
12345678
*DUMP LET
```

DWADR (Disk Write Address)

The DWADR routine writes sector addresses on every sector in the Working Storage area. It restores correct disk sector addresses in the Working Storage area if they have been modified during execution of a user's program. Previous contents of the area are overlaid. Following the address word, the first two words of each sector contain D120 2663 (in hexadecimal). The next 238 words have the format Annn, where nnn is the hexadecimal address of the sector; the last 80

Line 1:	LET						
Line 2: (Entries in comma-base and adjusted addresses – see Appendix E)	XXXX XXXX Work Storage starting sector address	XXXX XXXX Disk block address available for next User area program or data file	XXXX XXXX Number of words used by LET				
Line 3: (LET sector header words)	XXXX Relative sector number (0-7)	XXXX 0 if last sector of LET; otherwise non-zero	XXXX Number of disk blocks referenced by this sector of LET	XXXX Words available in this sector for more entries	XXXX Sector address of next sector (0 if last sector)		
Line 4 . . . Line n	XXXXX Program name	XXXX Program size (disk blocks)	XXXX Starting address (disk blocks)				For DSF programs
Line 4 . . . Line n	XXXXX Program name	XXXX Program size (disk blocks)	XXXX Starting address in User area (disk blocks)	XXXX Execute core address (absolute)	XXXX Program load address in core	XXXX Actual word count of core image program (includes a 60-word header)	For core image programs
Line 4 . . . Line n	XXXXX Data file name	XXXX Data file size (disk blocks)	XXXX Starting address in User area (disk blocks)	0000 0000 Reserved		XXXX Data file size (disk blocks)	For data files

NOTE 1: The header words of the first sector are printed on line 3. Additional header words are printed for each following sector as required.

NOTE 2: For multi-entry subroutines, the Program Size and Starting Address fields for entry points subsequent to the first one will be blank.

NOTE 3: Program size is the disk block count of the program. This corresponds to word 3 of the actual LET entry (see Appendix G).

NOTE 4: Words 4, 5, and 6 of the printout reflect the actual LET entry words 4, 5, and 6.

NOTE 5: All numbers are in hexadecimal.

Figure 7. Output Format from a DUMPLET Operation (LET)

words are zeros. The control record has the following format:

```
cc
1 2 3 4 5 6
* DWADR
```

DELETE (Delete Program or Data)

The DELETE routine deletes a specified program or data file from the User or Fixed area. The LET or FLET entry is deleted and if the program was in the User area the User area is repacked. A 1DUMY entry is created to replace deleted FLET entries. Although no packing of the Fixed area occurs, dummy entries in FLET are packed so that two are not adjacent but are consolidated. The control record has the following format:

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
* D E L E T E                                    Name
```

DEFINE (Define)

The DEFINE routine defines variable parameters required by the Monitor System. The following options are available:

- Define or increase the size of the Fixed area
- Delete the assembler from the system
- Delete FORTRAN from the system

If the user wishes to delete the assembler or FORTRAN, he must do so before he defines a Fixed area.

Within the Fixed area, programs can be stored at fixed disk locations. This area is defined as a whole number of cylinders, with a minimum of two, and it may be incremented (but not decreased) by a whole number of cylinders at any time. Defining or increasing the size of the Fixed area reduces disk storage available for User and Working Storage areas by the same amount.

Line 1:	FLET				
Line 2: (Entries in COMMA)	XXXX XXXX Sector address of CIB	XXXX XXXX Sector address of FLET	XXXX XXXX Number of words used by FLET		
Line 3: (FLET sector header words)	XXXX Relative sector number (first sector is numbered 16)	XXXX 0 if last sector of FLET; otherwise non-zero	XXXX Number of disk blocks referenced by this sector of FLET	XXXX Words available in this sector for more entries	XXXX Sector address of next sector (0 if last sector)
Line 4 ⋮ Line n	FLET entries are the same as for LET except that DSF programs do not appear in the Fixed area; therefore, no three-word entries appear in FLET.				

NOTE 1: All references are in disk blocks unless otherwise indicated.

NOTE 2: The header words of the first sector are printed on line 3. Additional header words are printed for each following sector as required; there is a header for each 52 FLET entries.

NOTE 3: Program size is the disk block count of the program. This corresponds to word 3 of the actual FLET entry (see Appendix G).

NOTE 4: Words 4, 5, and 6 of the printout reflect the actual FLET entry words 4, 5, and 6.

NOTE 5: All numbers are in hexadecimal.

Figure 8. Output Format from a DUMPLET Operation (FLET)

Deleting the assembler and/or FORTRAN packs the remaining information on the disk, thus increasing disk storage available for User and Working Storage areas by the amount occupied by the deleted programs (see Figure 2).

The control record formats are as follows:

To Define the Fixed area -

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
*DEFINEbFIXEDbAREA                                          NNNN
```

where NNNN = positive cylinder count in decimal, right-justified, specifying the initial size of Fixed area (minimum of 2 cylinders) or an increment to the Fixed area.

NOTE: The first cylinder of the first DEFINE FIXED AREA is used for FLET.

To Delete the Assembler -

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
*DEFINEbVOIDbASSEMBLER
```

To Delete FORTRAN -

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
*DEFINEbVOIDbFORTRAN
```

EDIT (to recall system loader)

The *EDIT control record is used only by DUP to recall the System Loader, which initially loads the system into disk storage. The control record has the following format:

```
cc
1 2 3 4 5
*EDIT
```

NOTE: This record must not be used by the user.

DUP MESSAGES

Each DUP control record is printed at the time it is read, thus signaling that DUP has control and will remain in control until the next monitor control record is properly read. When a requested DUP function has been successfully completed, the following two-word exit message is printed in hexadecimal:

Word 1: Disk block address of program or disk area that has been processed.
Word 2: Number of disk blocks involved.

For the DUP functions listed below, these words contain the following information (all addresses and lengths are given in disk blocks):

DUP Function	Information Printed
DUMP, STORE, STORECI, STOREMOD, DELETE	Program address* and program length**
DUMPDATA, STOREDATA	Program address and decimal sectors or records specified converted to disk blocks
DUMPLET - (LET)	User area address and User area length
DUMPLET - (FLET)	Fixed area address and Fixed area length
DWADR	Working Storage address and Working Storage length
DEFINE FIXED AREA	Fixed area address and size
DEFINE VOID ASSEMBLER	Former address and size of assembler
DEFINE VOID FORTRAN	Former address and size of FORTRAN

If the above words are not printed, the DUP function was not successfully completed.

* If storing or dumping from Working Storage, the address of Working Storage is printed.
**Length is the third word of LET/FLET entry (see Appendix G).

The DUP error messages are described in Appendix A.

DUP OPERATING NOTES

DUP functions should not be interrupted before the exit message is printed, since the status of LET/FLET, COMMA, and DCOM may be in an intermediate unusable state (see DUP Waits and Loops in Appendix A).

Some DELETE functions may take several minutes since they may have to pack much of disk storage. These long DELETE and DEFINE functions must be allowed to complete their respective operations.

Control can be returned to the DUP section that reads DUP and monitor control records by manually branching to core location 0276.

ASSEMBLER

The language for the monitor assembler is described in the publication IBM 1130 Assembler Language (Form C26-5927). Therefore, only a general description of the operation and the control records for the monitor assembler are described in this section.

The monitor assembler cannot be operated independent of the Monitor System; however, the assembler can be deleted from the Monitor System if desired.

A monitor control record with the pseudo-op ASM is used to call the assembler into operation. The assembler reads the source program, including control records, from cards or paper tape. After assembly, the object program resides in the disk Working Storage area, and can be called for execution with a monitor XEQ control record, or it can be stored in the User or Fixed area with a DUP STORE or STORECI operation or punched as a binary deck or tape with a DUP DUMP operation.

ASSEMBLER CONTROL RECORDS

Assembler control records are used to specify assembly options and to provide input to the assembly process for certain types of source decks. Assembler control records can be either card or paper tape.

All assembler control records have the following format:

Column 1: *
2-71: Option

Assembler control records can be written in free form, but at least one blank must separate the last character in the operation and the first character of any comments or numeric field.

Assembler control records and their meanings are listed below. A summary is contained in Table 6.

*TWO PASS MODE

The source deck (or tape) must be read twice. TWO PASS MODE must be specified when:

1. The user desires a list deck to be punched (see LIST DECK and LIST DECK E).
2. One pass operation cannot be performed because intermediate output (source records) fill the Working Storage area of disk.

Table 6. Summary of Assembler Control Records

*TWO PASS MODE	Read source deck twice; must be specified when LIST DECK or LIST DECK E is specified, or when intermediate output fills Working Storage
*LIST	Print a listing on the principal printing device
*LIST DECK	Punch a list deck on the principal I/O device (requires TWO PASS MODE)
*LIST DECK E	Punch only error codes (cc 18-19) into source program list deck (requires TWO PASS MODE)
*PRINT SYMBOL TABLE	Print a listing of the symbol table on the principal printing device
*PUNCH SYMBOL TABLE	Punch a list deck of the symbol table on the principal I/O device
*SAVE SYMBOL TABLE	Save symbol table on disk as a System Symbol Table
*SYSTEM SYMBOL TABLE	Use System Symbol Table to initialize symbol table for this assembly
*LEVEL n	n = interrupt level number. Required for ILS subroutines
*FILE n	n = number (decimal) of sectors of Working Storage required at object time by the program being assembled
*COMMON n	n = length of COMMON in words (decimal)

*LIST

A printed listing is provided on the principal printing device (console printer or 1132 Printer). The format of the printed listing corresponds to that of the list deck (see Figure 9).

*LIST DECK

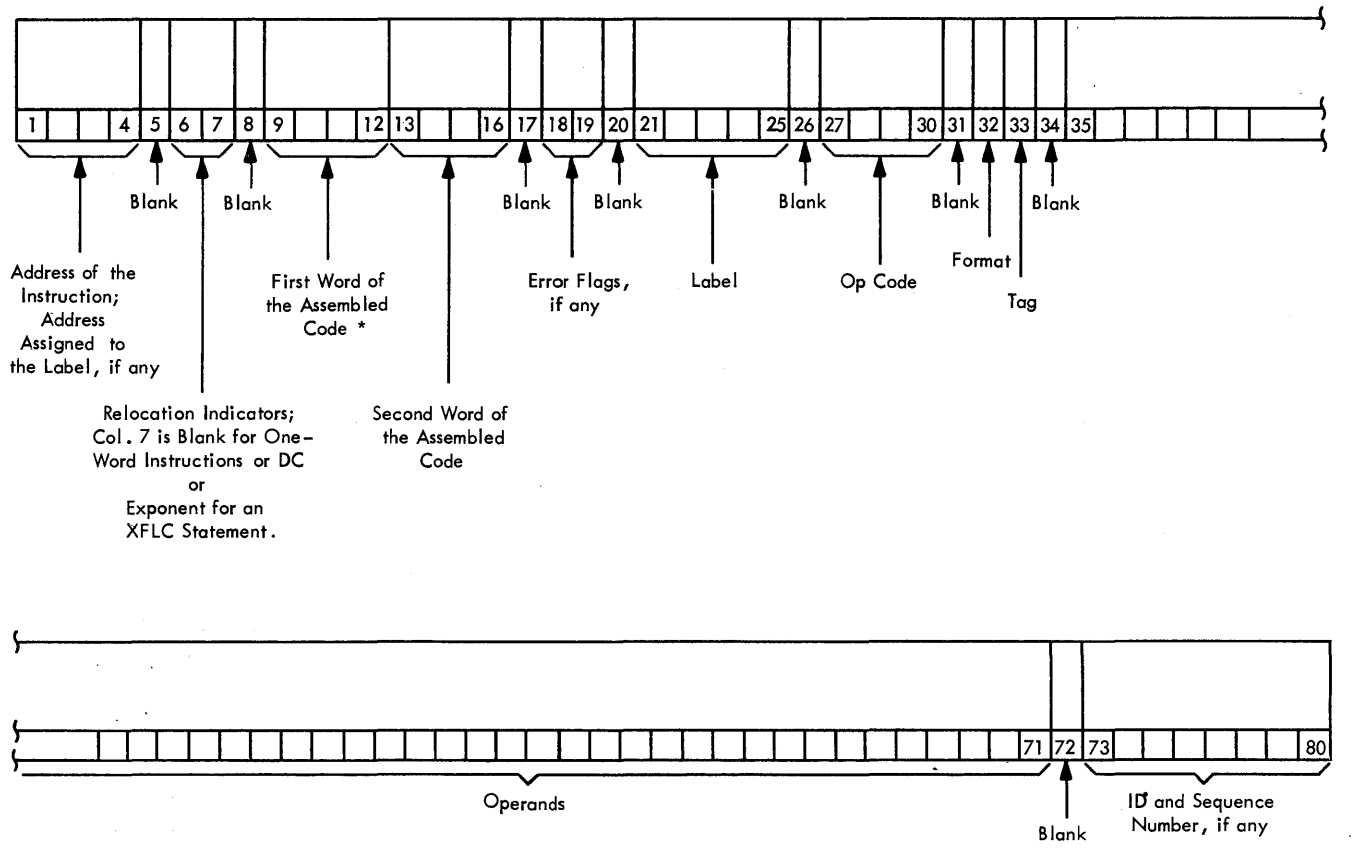
A list deck is punched on the principal I/O device (card or paper tape). This option requires two passes (TWO PASS MODE). The list deck format is shown in Figure 9. In cards, object information is punched into columns 1-19 of the source deck in pass 2 to make the list deck. In paper tape, the list tape punched is similar to the input tape, but with 20 frames added to the beginning of each record corresponding to card columns 1-20.

*LIST DECK E

Same as LIST DECK except no object information other than errors (positions 18-19) are punched.

*PRINT SYMBOL TABLE

A printed listing of the symbol table is provided on the principal printing device (console printer or 1132 Printer). Symbols are grouped five per line. Multiply-defined symbols are preceded by the letter M; symbols with absolute values in a relocatable program are preceded by the letter A. The M and A flags, however, are not counted as assembly errors.



* For EBC Statements, Col. 9-12 Contains the Number of EBC Characters

For BSS and BES Statements, Col. 9-12 Contains the Number of Words Reserved for the Block.

Figure 9. List Deck Format

*PUNCH SYMBOL TABLE

A list deck of the symbol table is punched on the principal I/O device (card or paper tape). The record format is the same as for PRINT SYMBOL TABLE. This option may be advantageous if off-line card-to-printer or paper tape-to-printer facilities are available.

*SAVE SYMBOL TABLE

The symbol table generated in this assembly is saved on the disk as a System Symbol Table. The System Symbol Table is saved until the next SAVE SYMBOL TABLE control record causes a new assembly-generated symbol table to replace it. This control record is also used with the SYSTEM SYMBOL TABLE control record to add symbols to the System Symbol Table. The SAVE SYMBOL TABLE option requires this assembly to be absolute. If any assembler errors are detected, or if the symbol table exceeds the allowable size of the System Symbol Table - 100 symbols - the symbol table will not be saved as a System Symbol Table, and an assembler error message will be printed (refer to Appendix A, Assembler Error Messages).

*SYSTEM SYMBOL TABLE

Before assembly begins, the System Symbol Table (previously built by a SAVE SYMBOL TABLE assembly) is copied into the symbol table for this assembly. This control record is used when it is desired to refer to symbols in the System Symbol Table without definition of those symbols in the source program, or together with the SAVE SYMBOL TABLE control record when it is desired to add symbols to the System Symbol Table. All symbols in the symbol table from the System Symbol Table will have absolute values.

*LEVELbn

This control record is required for the assembly of an ILS routine. n = A decimal interrupt level number (0-5). If the device operates on two levels of interrupts (1442 Card Read Punch), two LEVEL control records are required. At least one blank must separate the word LEVEL and the interrupt level number.

*FILEbn

n = Number of sectors (decimal) of the disk Working Storage area required at object time by the program being assembled. These sectors will be reserved at the beginning of Working Storage before any LOCALs or SOCALs are stored. This control record is used only when assembling a relocatable mainline program. At least one blank must separate the word FILE and the number of sectors.

*COMMONbn

n = Length of COMMON in words (decimal). This allows a COMMON area to be saved in linking from a FORTRAN mainline program to an assembly mainline and linking back to a FORTRAN mainline. At least one blank must separate the word COMMON and the decimal number.

ORIGIN OF SOURCE PROGRAM

The origin of a relocatable source program will always be at relative zero unless otherwise specified in the source program.

The origin of an absolute source program, if not otherwise specified, will be at the end of the disk routine DISKN (location 0438). If the program will use another disk routine, the origin may be set lower to correspond to the proper disk routine. If no disk routine is used, the origin may be set to the end of the disk routine DISKZ (refer to Origins for Core Loads).

ASSEMBLER PAPER TAPE FORMAT

The paper tape input to the assembler is punched on PTTC/8 tape, one frame per character. The format of the tape control records is the same as the card format. The format of the symbolic program tape records is the same as the card format except for the following:

1. The tape does not contain preceding blanks corresponding to card columns 1-20.
2. The tape does not contain blanks or data corresponding to card columns 72-80.
3. Trailing blanks need not be punched. Therefore, up to 51 characters (corresponding to card columns 21-71) can appear in the tape record.

Tape records are separated by NL (new line) characters (code DD). The delete character (code 7F) is ignored whenever it is read, but the reader stop character (RS, code 0D) causes the program reading the tape to wait and start reading again when PROGRAM START is pressed. The case shift characters (codes 0E, 6E), when required, are not considered to occupy a space in the format.

ASSEMBLER MESSAGES AND ERROR CODES

Appendix A contains the assembler error messages printed during operation of the 1130 Monitor. If LIST DECK or LIST DECK E is specified, the error detection codes shown in Table 7 are punched in columns 18 and 19. For the first error detected in each statement the assembler stores and then punches the code in column 18; the code for a second error is stored, overlayed by any subsequent errors, and punched in column 19. Thus, if more than two errors are detected in the same statement, only the first and last are indicated.

At the end of the assembly, a message is printed indicating the number of assembly errors detected in the source program. Since no more than two errors are flagged per statement, the error count may exceed the actual number of flags.

If symbol table overflow exceeds 32 sectors in Working Storage, an assembler error message is printed (refer to Appendix A). The maximum size of the symbol table (including overflow) and, hence, the maximum number of symbols that can be defined in a program is determined by the size of core storage as indicated below:

Type of Assembly	Size of Core Storage (words) 4096	8192
With Listing	3502	4867
One Pass, No Listing	3574	4940
Two Pass, No Listing	3609	4974

ASSEMBLER OPERATING PROCEDURES

Card Input

The source deck (including assembler control cards) can be assembled as part of a job, or it can be assembled as a separate job. In either case, the source deck must be preceded by a ASM monitor control record.

Table 7. Assembler Error Detection Codes

Flag	Cause	Assembler Action
A	Address Error Attempt made to specify displacement field, directly or indirectly, outside range of –128 to +127.	Displacement set to zero
C	Condition Code Error Character other than +, –, Z, E, C, or O detected in first operand of short branch or second operand of long BSC, BOSC, or BSI statement.	Displacement set to zero
F	Format Code Error Character other than L, I, X, or blank detected in col. 32, or L or I format specified for instruction valid only in short form.	Instruction processed as if L format were specified, unless that instruction is valid only in short form, in which case it is processed as if the X format were specified
L	Label Error Invalid symbol detected in label field.	Label ignored
M	Multiply Defined Label Error Duplicate symbol encountered in operand.	First occurrence of symbol in label field defines its value; subsequent occurrences of symbol in label field cause a multiply defined indicator to be inserted in symbol table entry (Bit 0 of first word).
R	Relocation Error Expression does not have valid relocation. Non-absolute displacement specified. Absolute origin specified in relocatable program. Non-absolute operand specified in BSS or BES. Non-relocatable operand in END statement of relocatable mainline program. ENT operand non-relocatable.	Expression set to zero Displacement set to zero Origin ignored Operand assumed to be zero Card columns 9–12 left blank; entry assumed to be relative zero Statement ignored
S	Syntax Error Invalid expression (e.g., invalid symbol, adjacent operators, illegal constant) Illegal character in record. Main program entry point not specified in END operand. Incorrect syntax in EBC statement (e.g., no delimiter in card column 35, zero character count). Invalid label in ENT or ISS operand.	Expression set to zero If illegal character appears in expression, label, op code, format, or tag field, additional errors may be caused. Card columns 9–12 left blank; entry assumed to be relative zero. Card columns 9–12 not punched; address counter incremented by 17. Statement ignored
T	Tag Error Card column 33 contains character other than blank, 0, 1, 2, or 3 in instruction statement.	Tag of zero assumed
U	Undefined Symbol Undefined symbol in expression	Expression set to absolute zero
O	Op Code Unrecognized ISS, ILS, ENT, LIBR, SPR, EPR, or ABS incorrectly placed.	Statement ignored and address counter incremented by 2. Statement ignored

In most cases, the source deck is passed through the 1442 Card Read Punch only once. If the assembly is part of a stacked job, the assembly proceeds without operator intervention. If the END card is the last card in the stack, when the reader goes not ready, press reader START to process the last card.

In some cases it may be necessary to assemble in the two-pass mode, that is, pass the source deck through the 1442 Card Read Punch twice. If a copy of the source deck is placed behind the original, the source deck will be read twice, and a stacked job is again possible even when in the two-pass mode.

It is important to note that when a deck is being assembled in two-pass mode, the assembler is ready to read another card as soon as Pass 1 processing of the END card is completed. Therefore, a monitor control record must not follow the END card the first time (or the first END card if the deck has been copied), or the assembler will trap this record, terminating the assembly and returning control to the Supervisor.

If the deck is not copied, the END card should be the last card. Press reader START to process the last card and complete Pass 1. The assembler will then try to begin reading cards for Pass 2, therefore the source deck (with its control cards) should be removed from the stacker and placed in the hopper. Pressing reader START will continue Pass 2 of the assembly. The card reader will go not ready when all cards but the END card have been read. Press reader START to process the END card and complete the assembly. Operation is continuous from Pass 1 to Pass 2 if the source deck is replaced behind the END card from the stacker during Pass 1.

If the *PUNCH SYMBOL TABLE assembler control card is used, sufficient blank cards must be placed after the END card and before the next monitor control record in the stacked job. In estimating the number of blank cards required, allow one card for each five symbols used in the source deck. Unnecessary blank cards will be passed to the next monitor control record.

Paper Tape Input

Most of the procedures for card input are also applicable to paper tape input.

If the assembly is performed in the one-pass mode, operation is continuous, and control is returned to the Supervisor which will then pass any delete codes between the assembler and the next monitor control record. The assembler will also pass any delete codes that may occur between records of the source program.

When it is necessary to assemble in the two-pass mode, one of the following techniques may be used:

1. Have the stacked job tape contain two copies of the source program. The assembler will simply begin reading the copy after the original has been read in Pass 1.

2. Assemble outside of the stacked job tape. The job tape, or a separate strip containing an ASM monitor control record serves to bring the assembler into core. A separate source program tape (including assembler control records) should then be readied on the paper tape reader, and the assembler will read this tape and complete Pass 1. Ready the source tape again and the assembler will complete Pass 2. A stacked job tape can now be readied again on the paper tape reader, and the Supervisor will continue with the stack.

3. The assembly of a program may start in one-pass mode and then be changed to two-pass mode (see Assembler Error Messages, Appendix A). The assembler will wait, and pressing PROGRAM START continues the assembly in Pass 1 of two-pass mode. If this assembly is part of a stacked job, operator intervention is necessary to prevent the assembler from reading the monitor control record which follows the END record (applicable to card input also). When Pass 1 intermediate output may fill Working Storage, it is recommended that sufficient length of all delete codes be punched into the tape after the END statement and before the next monitor control. When the assembler is reading the delete codes following the END record, the operator should press PROGRAM STOP, and manually reposition the tape at the beginning of the source program. When the tape is positioned, press PROGRAM START to continue Pass 2 of the assembly.

When punching a list tape (*LIST DECK or *LIST DECK E), first create a leader in the punched tape by holding down FEED and DELETE on the punch (press DELETE before FEED and release FEED before releasing DELETE). The same procedure should be used to create a trailer following the last record punched by the assembler.

When the paper tape reader or punch is not ready, the assembler will wait at location 0041 with 3000_{16} displayed in the accumulator. Ready the punch or reader, and press PROGRAM START to continue.

FORTRAN COMPILER

The language for the Monitor FORTRAN compiler is described in the publication IBM 1130 FORTRAN Language (Form C26-5933). Therefore, only a general description of the Monitor FORTRAN compiler operation is contained here.

The FORTRAN compiler cannot be operated independent of the Monitor System; but, if desired, the compiler can be deleted from the system.

A monitor control record having the pseudo-op FOR is used to call the FORTRAN compiler into operation. The compiler reads the source program from cards or paper tape. After compilation, the object program resides in the disk Working Storage area, and can be called for execution with a monitor XEQ control record, loaded to the User or Fixed area with a DUP STORE or STORECI operation, or punched as a binary deck or tape with a DUP DUMP operation. All FORTRAN programs are compiled in relocatable format.

For 1130 FORTRAN I/O logical unit definitions, the I/O unit numbers are permanently set as described in Table 8.

FORTRAN CONTROL RECORDS

Before a FORTRAN program is compiled, the user can specify certain options by means of control records which must precede the source program and can be in any order. The IOCS and NAME control records can be used only in mainline programs; the others can be used in both mainline programs and subroutines.

Table 8. I/O Logical Unit Designations

Logical Unit Number	Device	Kind of Transmission	Record Size Allowed
1	Console printer	Output only	120
2	1442 Card Read Punch	Input/output	80
3	1132 Printer	Output only	1 carriage control + 120
4	1134–1055 Paper Tape Reader/Punch	Input/output	80, plus max. of 80 case shifts for PTTC/8 code, plus NL code.
6	Keyboard	Input only	80

All FORTRAN control records have the following format:

Column 1: * (asterisk)
2-72: Option

FORTRAN control records can be written in free form, no comments allowed. Any unrecognizable control records are considered as comments control records.

FORTRAN control records and their meanings are listed below. A summary is contained in Table 9.

*IOCS (CARD, TYPEWRITER, KEYBOARD, 1132 PRINTER, PAPER TAPE, DISK)

This record is required to specify any I/O device that is to be used during execution of the program; however, only the devices required should be included. Because the *IOCS record can appear only in the mainline program, it must include all the I/O devices used by all FORTRAN subprograms that will be called. The device names must be in parentheses with a comma between each name.

FORTRAN subprograms written in assembly language can use any I/O subroutines for any device that is not mentioned in *IOCS and that is not on the same interrupt level as a device in *IOCS. Otherwise, the subprograms must use FORTRAN I/O routines (CARDZ, PAPTZ, PRNTZ, WRTYZ, TYPEZ, DISKZ).

*LIST SOURCE PROGRAM

The source program is listed as it is read in.

*LIST SUBPROGRAM NAMES

The names of all subprograms (including EXTERNAL subprograms) called directly by the compiled program are listed.

*LIST SYMBOL TABLE

The following items are listed:

- Variable names and their relative addresses
- Statement numbers and their relative addresses

Table 9. Summary of FORTRAN Control Records

* NAME XXXXX	(XXXXX = program name to be printed on listing)
* IOCS (CARD, TYPEWRITER, KEYBOARD, 1132 PRINTER, PAPER TAPE, DISK)	Delete any not used
**header information to be printed on each compiler output page	
* ONE WORD INTEGERS	(Store integer variables in one word)
* EXTENDED PRECISION	(Store floating point variables and constants in 3 words instead of 2)
* ARITHMETIC TRACE	(Switch 15 ON to print result of each assignment statement)
* TRANSFER TRACE	(Switch 15 ON to print value of IF or Computed GOTO)
* LIST SOURCE PROGRAM	(List source program as it is read in)
* LIST SUBPROGRAM NAMES	(List subprograms called directly by compiled program)
* LIST SYMBOL TABLE	(List symbols, statement numbers, constants)
* LIST ALL	(List source program, subprogram names, symbol table)

- Statement function names and their relative addresses
- Constants and their relative addresses
- Unreferenced statement numbers

*LIST ALL

The source program, subprogram names, and symbol table are listed. If this control record is used, the other LIST control records are not required.

*EXTENDED PRECISION

Variables and real constants are stored in three words instead of two, and the compiler generates linkage to extended precision routines. When this control record is used, the program does not conform to the ASA Basic FORTRAN standard for data storage, and it may require modification in order to be used with other FORTRAN systems.

*ONE WORD INTEGERS

Integer variables are allocated one word of storage rather than the same allocation used for real variables. Whether this control record is used or not, integer constants are always contained in one word. When this control record is used, the program does not conform the the ASA Basic FORTRAN standard for data storage, and it may require modification in order to be used with other FORTRAN systems.

*NAME XXXXX

The program name represented by XXXXX is printed on the listing. XXXXX is five consecutive characters (including blanks) starting at the first non-blank column. This control record is used only on mainline programs, since subprogram names are automatically taken from the FUNCTION or SUBROUTINE statement.

**Header Information

The information between columns 3-72 is printed at the top of each page of compilation printout when an 1132 Printer is the principal system printer.

*ARITHMETIC TRACE

The compiler generates linkage to trace routines which are executed whenever a value is assigned to a variable on the left of an equal sign. If Console Entry Switch 15 is turned on at execution time and

program logic (see Optional Tracing) does not prevent tracing, the value of the assigned variable is printed as it is calculated.

*TRANSFER TRACE

The compiler generates linkage to trace routines which are executed whenever an IF statement or Computed GOTO statement is encountered. If Console Entry Switch 15 is turned on at execution time and program logic (see Optional Tracing) does not prevent tracing, the value of the IF expression or the value of the Computed GOTO index is printed.

If tracing is requested, an *IOCS control record must also be present to indicate that either typewriter or printer is needed. If both typewriter and printer are indicated in the *IOCS record, the printer is used for tracing.

The traced value for the assignment of a variable on the left of an equal sign of an arithmetic statement is printed with one leading asterisk. For the expression of an IF statement, the traced value is printed with two leading asterisks. The traced value for the index of a Computed GOTO statement is printed with three leading asterisks.

Optional Tracing

The user can elect to trace only selected parts of the program by placing statements in the source program logic flow to start and stop tracing. This is done by executing a CALL to either subroutine:

CALL TSTOP (to stop tracing)
CALL TSTRT (to start tracing)

Thus, tracing occurs only if:

- The trace control records were compiled with the source program.
- Console Entry Switch 15 is on (can be turned off at any time).
- A CALL TSTOP has not been executed, or a CALL TSTRT has been executed since the last CALL TSTOP.

Operating Notes - *LIST Control Cards

A constant in a STOP or PAUSE statement is treated as a hexadecimal number. This hexadecimal number and its decimal equivalent appear in the list of constants.

Variables and constants that require more than one word of storage have the address of the word nearest the zero address of the machine. In the case of arrays, the given address refers to the addressed word of the first element. In the case of a two- or three-word integer, the integer value is contained in the addressed word. The first variable listed might not be addressed at 0000 because room may be required for generated temporary storage locations.

The relative address for variables not in COMMON would be the actual address if the program started at storage location zero. The relative address for variables in COMMON would be the actual address if the machine had 32K storage. The Loader makes any necessary adjustments. Variables in COMMON are adjusted to reside in the high-order core location of the machine being used (e.g., first COMMON variable will be loaded to 8191 on an 8K machine).

Loading begins at core location 01C2 (450 decimal). The DISKZ routine is used regardless of what disk routine is requested on the XEQ control record (refer to Origins for Core Loads).

FORTRAN PRINTOUTS

Compilation Messages

Near the end of the compilation, core usage information and the features supported (control records used) are printed out as follows:

```
FEATURES SUPPORTED
  EXTENDED PRECISION
  ONE WORD INTEGERS
  TRANSFER TRACE
  ARITHMETIC TRACE
  IOCS
CORE REQUIREMENTS FOR XXXXX
COMMON YYYYY VARIABLES YYYYY PROGRAM YYYYY
```

where XXXXX is the name of the program designated in the *NAME control record or in the SUBROUTINE or FUNCTION statement, and YYYYY is the number of words allocated for the specified parts of the program.

Compilation Error Messages

During compilation a check is made to determine if certain errors have occurred. If one or more of these errors have been detected the error indications are printed at the conclusion of compilation, and no object program is stored on the disk. Only one error is detected for each statement. In addition, due to the interaction of error conditions, the occurrence of some errors may prevent the detection of others until those which have been detected are corrected. With the exception of type 00 messages listed below, the error message appears in the following format:

```
C NN ERROR IN STATEMENT NUMBER XXXXX + YYY
```

where NN is the error number, described in Appendix A. XXXXX is the last encountered valid statement number, and YYY is the count of statements from statement XXXXX.

Specification statements (except FORMAT) are not counted unless they contain an error. Statement numbers on specification statements and statement functions are ignored.

Type 00 Error Messages

Code and Message	Meaning
C 00 MON CALL	A Monitor call was executed (via operator intervention) during the compilation; the compilation is terminated and control is returned to the monitor.
C 00 OVER 50 DISK ERRORS AT SECTOR nnnn	The FORTRAN disk I/O routine encountered an unrecoverable disk error during compilation; nnnn is the hexadecimal address of the bad sector.
C 00 WORKING STORAGE EXCEEDED	The working storage area on disk is too small to accommodate the string area and symbol table for the program being compiled; the compilation is terminated.

The following message is printed for a normal end of compilation (with or without errors):

```
END OF COMPILATION
```

SUBROUTINE LIBRARY

The 1130 Subroutine Library consists of a group of subroutines that aid the programmer in making efficient use of the IBM 1130 Computing System. Descriptions of the subroutines and methods for programming them are contained in the publication, IBM 1130 Subroutine Library (Form C26-5929).

The following paragraphs describe the use of the IBM-supplied subroutines and discuss pre-operative errors and I/O error restarts where special handling is required.

PRE-OPERATIVE ERRORS

A pre-operative error is an error condition detected before an I/O operation is started. It denotes either an illegal LIBF parameter, an illegal specification in I/O area, or a device not-ready condition. This error causes a branch to location 0029 and the following conditions:

- The I-counter displays the address 002A.
- The Accumulator displays an error code represented by four hexadecimal digits.

  Digit 1 identifies the ISS called:
  1 - CARD0 or CARD1
  2 - TYPE0 or WRTY0
  3 - PAPT1 or PAPTN
  5 - DISK0, DISK1, or DISKN
  6 - PRINT1
  7 - PLOT1

  Digits 2 and 3 are not used.
  Digit 4 identifies the error:
  0 - Device not ready
  1 - Illegal LIBF parameter or illegal specification in I/O area

- Location 0028 contains the address of the LIBF in question.

The ISS is set up to attempt initiation of the operation a second time if the LIBF is re-executed. Therefore, since the Loader stores a wait instruction in location 0029 and an indirect branch to location 0028 in locations 002A and 002B, the LIBF can be executed again by pressing PROGRAM START.

When a pre-operative error is encountered the operator can:

- Correct the error condition if possible and press PROGRAM START, or
- Note the contents of the Accumulator and location 0028, dump core storage, and proceed with the next job.

CARD SUBROUTINE (CARD0 AND CARD1) ERRORS

Error Parameters

CARD0. There is no error parameter. If an error is detected during processing of an operation-complete interrupt, the subroutine loops internally, with interrupt level 4 on until the 1442 becomes ready, and then retries the operation.

CARD1. There is an error parameter. If an error is detected during processing of an operation-complete interrupt, the user program can elect to terminate (clear "routine busy" and the interrupt level) or to retry. A retry consists of looping internally, with interrupt level 4 on until the 1442 becomes ready, and then reinitiating the function.

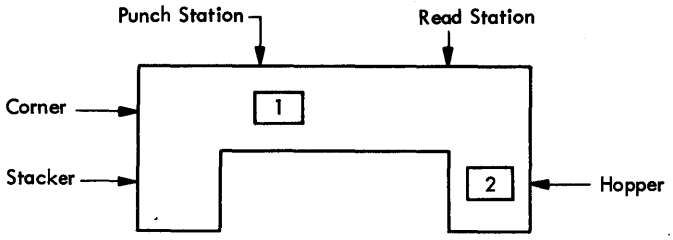
1442 Errors and Operator Procedures

If a 1442 error occurs, the 1442 becomes not ready until the operator has intervened. Unless the stop is caused by a stacker full (no indicator) or Chip Box indication, the 1442 card path must be cleared before proceeding. The 1442 error indicators and the position of the cards in the feed path should be used to determine which cards must be placed back in the hopper.

For the card subroutines, a retry consists of positioning the cards (i.e., skipping the first card in the hopper, if necessary, on a read or feed operation) and reinitiating the function whenever the card reader becomes ready.

Hopper Misfeed. Indicates that card 2 failed to pass properly from the hopper to the read station during the card 1 feed cycle.

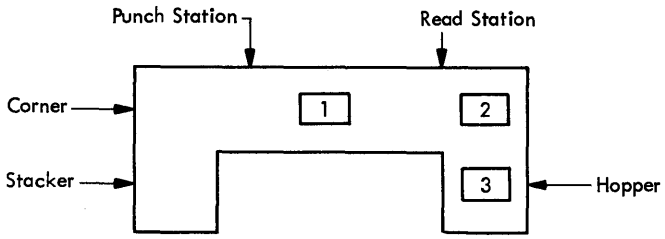
Card positions after error:



Error indicator: HOPR
Operator procedure: When program halts, press NPRO to eject card 1, place card 1 in hopper before card 2, and ready the 1442.

Feed Check (punch station). Indicates that card 1 is improperly positioned in the punch station at the completion of its feed cycle.

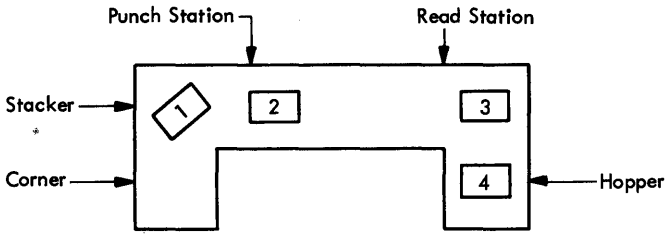
Card positions after error:



Error indicator: PUNCH STA
Operator procedure: When program halts, empty hopper, clear 1442 card path, place cards 1 and 2 in hopper before card 3, and ready the 1442.

Transport. Indicates that card 1 has jammed in the stacker during the feed cycle for card 2.

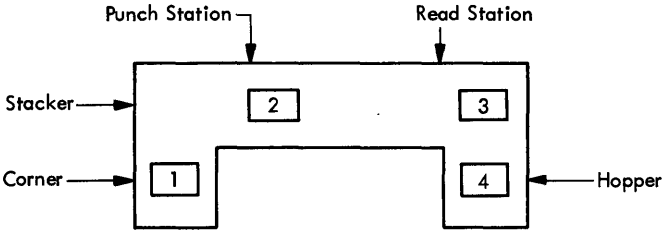
Card positions after error:



Error indicator: TRANS
Operator procedure: When program halts, empty hopper, clear 1442 card path, place cards 2 and 3 in hopper before card 4, and ready the 1442.

Feed Cycle. Indicates that the 1442 took an unrequested feed cycle and, therefore, cards 1, 2, and 3 are each one station farther ahead in the 1442 card path than they should be.

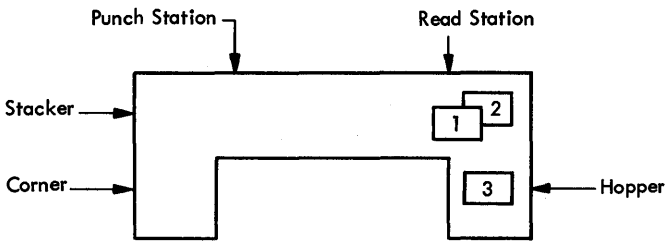
Card positions after error:



Error indicator: FEED CLU
Operator procedure: When program halts, empty hopper, press NPRO to eject cards 2 and 3, place cards 1, 2, and 3 in hopper before card 4, and ready the 1442.

Feed Check (read station). Indicates that card 1 failed to eject from the read station during its feed cycle.

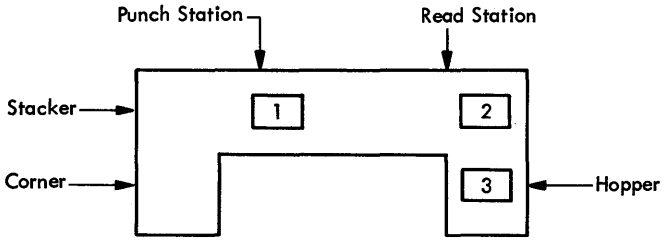
Card positions after error:



Error indicator: READ STA
Operator procedure: When program halts, empty hopper, clear 1442 card path, place cards 1 and 2 in hopper before card 3, and ready the 1442.

Read Registration. Indicates incorrect card registration or a difference between the first and second reading of a column.

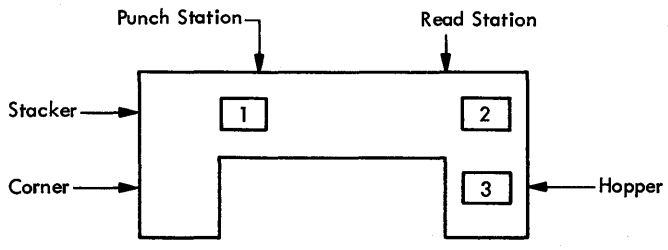
Card positions after error:



Error indicator: READ REG
Operator procedure: See Feed check (punch station). Repeated failures of this type might indicate a machine malfunction.

Punch Check. Indicates an error in output punching.

Card positions after error:



Error indicator: PUNCH

Operator procedure: When program halts, empty hopper, check card position and press NPRO to clear 1442 card path. If necessary, correct card 1 to pre-punched state. Place (corrected) card 1 and card 2 in hopper before card 3 and ready the 1442.

CONSOLE PRINTER SUBROUTINE (TYPE0 AND WRTY0) ERRORS

If the carrier attempts to print beyond the manually positioned margins, a carrier restore (independent of the program) occurs.

Subroutine printing begins wherever the carrier is positioned as a result of the previous print operation. There is no automatic carrier return as a result of an LIBF.

If the console printer indicates a not-ready condition after printing has begun, the subroutines loop internally, with interrupt level 4 on, waiting for the console printer to become ready. Operator procedures are as follows:

1. Press IMM STOP on the console.
2. Ready the console printer.
3. Press PROGRAM START on the console.

KEYBOARD SUBROUTINE (TYPE0) FUNCTIONS

Re-entry

When the Erase Field key is pressed, a character interrupt signals the interrupt response routine that the previously-entered keyboard message is in error and will be re-entered. The routine prints two slashes on the console printer, restores the carrier to a new line, and prepares to replace the old message in the I/O area with the new message. The operator then enters the new message. The old message in the I/O area is not cleared. The new message overlays the previous message, character by character. If the previous message was longer than the new message, characters from the previous message remain (following the NL character which terminated the new message).

Backspace

When the backspace key is pressed, the last graphic character entered is slashed and the address of the next character to be read is decremented by +1. If the backspace key is pressed twice consecutively, the character address is decremented by +2, but only the last graphic character is slashed. For example, assume that ABCDE has been entered and the backspace key pressed three times. The next graphic character replaces the C, but only the E is slashed. If the character F had been used for replacement the paper would show ABCDE̸FFF but ABFFF would be stored in the buffer.

PAPER TAPE SUBROUTINES (PAPT)

If the reader or punch becomes not ready during an I/O operation, the subroutines exit to the user via the error parameter. The user can request the subroutine to terminate (clear device busy and interrupt level) or to loop on not-ready waiting for operator intervention (interrupt level 4 on).

The following procedure should be used to clear a paper tape not-ready condition:

1. Press IMM STOP on the console.
2. Ready the paper tape unit.
3. Press PROGRAM START on the console.

To load the paper tape reader, place the tape so that the delete characters punched in the leader are under the read starwheels. To begin reading at any point in the tape other than the leader, place the tape so that the frame (character position) preceding the character to be read is under the read starwheels. The first start reader control after tape is loaded or repositioned causes the reader to skip the character under the read starwheels and load the next character into the buffer.

ADDING AND REMOVING SUBROUTINES

Subroutines can be added to or removed from the subroutine library as desired by the user. The DUP control record STORE adds a subroutine, and the DUP control record DELETE removes a subroutine. Each subroutine in the IBM-supplied System Deck is preceded by a DUP STORE record.

The user should not remove subroutines that are called by other subroutines left in the library (refer to Appendix E for a list of subroutines called by other subroutines).

SYSTEM GENERATION OPERATING PROCEDURES (CARD SYSTEM)

Before the Disk Monitor System can begin operation, the user must perform the following functions:

1. Load and execute the IBM-supplied Disk Pack Initialization Routine (DPIR) to initialize the disk pack.
2. Prepare a Load Mode Control Card and System Configuration Cards, and insert these cards into the IBM-supplied System Deck.
3. Load the above deck into the disk.
4. Using the IBM-supplied Cold Start Card, load the Supervisor program into core storage from disk storage.

Each of the above procedures is described in detail in subsequent sections of this manual.

DISK PACK INITIALIZATION ROUTINE (DPIR)

The DPIR (Disk Pack Initialization Routine) performs the following functions:

1. Clears the disk and writes disk sector addresses on all cylinders.
2. Determines which, if any, sectors are defective and writes the addresses of the cylinders containing the defective sectors on sector 0000. If sector 0000 is defective, DPIR does not write any defective cylinder table.
3. Puts an ID on the disk pack.

The 1130 Disk Routines operate effectively with up to three cylinders containing defective sectors. An attempt to read or write a defective sector that is not identified in sector 0000 results in a read or write error after the operation has been attempted 10 times.

At the completion of DPIR, an eight-word table is written on sector 0000. The first word (word 0) of the table contains the sector address 0000. Words one, two, and three contain the first sector address of any defective cylinders found (maximum of three). When there is no defective cylinder, these words contain 0658_{16}. Word 4 is reserved. Words five, six, and seven contain a five character ID name in packed EBCDIC. Words five and six contain two characters per word, and word seven contains an EBCDIC character in the left half of the word and an EBCDIC blank in the right half of the word.

To determine which sectors are defective, the user can dump core upon completion of execution; the defective sector table starts at location 0739 and the actual count of defective sectors is in location 035E.

Table 10 lists the DPIR halt addresses.

DPIR Card Load Operating Procedures

The procedure for loading and executing the Disk Pack Initialization Routine is as follows:

1. Load the disk pack in the console cabinet.
2. Put the DPIR deck in the card hopper.
3. Set the console mode switch on RUN.
4. Press IMM STOP, then RESET on the console.
5. Press START on the card reader.
6. Press PROGRAM LOAD on the console.

The DPIR is read in and the keyboard is selected. (The keyboard Select light comes on.) Wait for the File Ready light to come on and then:

1. Enter the five-character ID to be written on the disk pack. If the ID is less than five characters, left-justify by following the ID with spaces. Only those characters recognized by the Supervisor should be used (see Appendix D). When the fifth character is entered, the program branches to execute. The disk surface is now cleared and the sector addresses are written. The routine waits at 038A.
2. Set all the Console Entry switches off.
3. Press PROGRAM START. The defective sector and file protect address data is written on sector 0000. A scan of the disk is now performed to check for seek failures. If a seek or read failure occurs, the routine waits at 03EA. Other DPIR halt addresses are described in Table 10.

Table 10. DPIR Halt Addresses

Halt Address (hex)	Meaning	Action Required
0346	A Write Select error has occurred.	CE intervention is required; reload the routine or branch to 02EE manually.
03D2	The disk is not ready.	Make the disk ready and restart the program.
0367	Sector 0000 is defective; the sector addresses have been written on the disk, but the table has not been written on sector 0000.	
03BE	The routine has run successfully and no defective sectors were found.	
03BA	The routine has run successfully and three or less defective sectors were found.	
03C8	The routine has run successfully, but more than three defective sectors were found.	
038A	The routine is in WAIT.	Turn the console entry switches off; then press PROGRAM START on the console.
03EA	A seek failure or read failure occurred during a scan of the disk.	

USER-SUPPLIED CARDS

Before loading the Disk Monitor System programs onto the disk, the user must prepare the following cards:

1. Load Mode Control Card
2. System Configuration Deck:
   a. SCON Card
   b. REQ Card(s)
   c. TERM Card

The System Loader will give error messages for missing or invalid user-supplied cards (see Appendix A).

Load Mode Control Card

The Load Mode Control Card is used to specify an initial load or a reload. It also permits the user to specify whether the assembler and/or FORTRAN is to be loaded. The format is shown in Table 11 (only columns 1 through 3 are used). For example,

Table 11. Load Mode Control Card Format

Column	Punch	Meaning
1	12 Punch	Initial load. Monitor System programs, ILS and ISS subroutines required by the system, and functional subroutines are loaded. However, the assembler will not be loaded if column 2 contains a 0 punch, and FORTRAN will not be loaded if column 2 contains a 1 punch.
	No 12 Punch	Reload. Monitor System programs are reloaded, and the contents of the User and Fixed areas (including LET/FLET) are not changed. Only programs presently on the disk can be reloaded; if the assembler or FORTRAN were not loaded during an initial load, or have since been deleted, they cannot be reloaded.
2	0 Punch	Bypass (do not load) assembler.
	No 0 Punch	Load assembler.
	1 Punch	Bypass (do not load) FORTRAN.
	No 1 Punch	Load FORTRAN.
3	9 Punch	Required in all cases to identify Load Mode Control Card.

to initially load the monitor (including the assembler and FORTRAN), the Load Mode Control Card is punched with a 12 punch in column 1 and a 9 punch in column 3.

System Configuration Deck

SCON Card

The SCON Card is the deck header card. The format is as follows:

Columns	Contents
1-4	SCON

REQ Cards

REQ Cards identify devices present in system. The System Loader uses this information for selective generation and loading of ILS subroutines and selective loading of ISS subroutines. The format is shown in Table 12.

TERM Card

The TERM Card is the last card of the System Configuration Deck. The format is as follows:

Columns	Contents
1-4	TERM

Table 12. REQ Card Format

Columns				
Device	1-3	10	15-16	21-22
		(ISS No.)	(Primary Interrupt Branch Address	(Second Interrupt Branch Address
1442 Card Read Punch	REQ	1	08	12
Input keyboard and console printer		2	12	Blank
1134 paper tape reader or 1055 paper tape punch		3	12	Blank
Disk		4	10	Blank
1132 Printer		6	09	Blank
Plotter		7	11	Blank

NOTE: If both the console printer and the 1132 Printer are included, the 1132 Printer will be the principal printing device; if both the 1442 Card Read Punch and the 1134/1055 paper tape units are included, the 1442 Card Read Punch will be the principal I/O device.

PROCEDURE FOR INITIALIZING DISK MONITOR SYSTEM FROM CARDS

1. Execute the following:
   a. Press IMM STOP on console.
   b. Press RESET on console.
   c. Press NPRO on the 1442 card reader.
2. Load the following decks into hopper of the 1442 card reader (see Figure 10).
   a. IBM-supplied System Loader Deck, Part 1. Columns 73-74 contain the ID: E1.
   b. User-supplied Load Mode Card.
   c. IBM-supplied System Loader Deck, Part 2. Columns 73-74 contain the ID: E2.
   d. User-supplied System Configuration Deck (SCON Card, REQ Cards, TERM Card).
   e. Remainder of IBM-supplied System Deck.
3. Execute the following:
   a. Turn the File Switch on, and wait for the File Ready light on the console to go on.
   b. Press START on the 1442 card reader.
   c. Press RESET on console.
   d. Press PROGRAM LOAD on console.

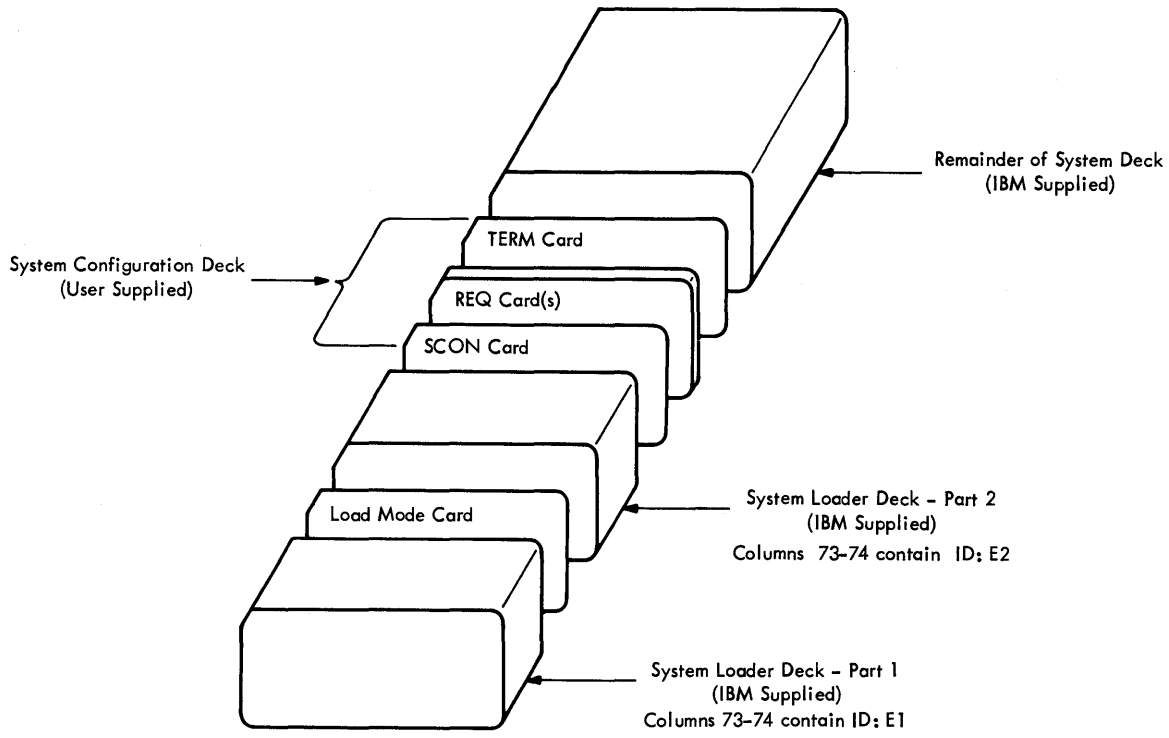


Figure 10. System Loader Card Sequence

COLD START OPERATING PROCEDURE

To begin operation of the Disk Monitor System after it has been loaded to the disk, the user must load the Supervisor Program into core storage from disk storage by using the IBM-supplied Cold Start Card (last card of subroutine deck) or Cold Start paper tape record.

The procedure for executing the Cold Start Card is as follows:

1. Insert the monitor disk pack in the console cabinet.
2. Turn the File Switch on, and wait for the File Ready light on the console to go on.
3. Put the Cold Start Card into the card hopper followed by a monitor control record to be processed.
4. Press IMM STOP, then RESET on the console.
5. Press START on the card reader.
6. Press PROGRAM LOAD on the console.

The Cold Start record reads the Cold Start sector (0001) from disk into core location 0802. The Monitor Supervisor Program is then read into core. The first monitor control record is read under control of the Supervisor Program by the Monitor Control Record Analyzer routine.

Possible stopping locations are given in Table 13.

NOTE: A cold start cannot be used to resume an operation that has been previously terminated.

Table 13. Cold Start Halt Addresses

Halt Address (hex)	Meaning
0024	The disk was not ready and the first XIO was treated as a NOP
002B	When sector 0001 was read, the address was not 0001
0034	There was a Disk Data Error
080F	Sector 000A was not read correctly
0816	Sector 0009 was not read correctly
0823	Sector 0008 was not read correctly
0839	There was a Disk Data Error
0026	Disk not ready
0803	Disk not ready
080A	Disk not ready
0811	Disk not ready
081E	Disk not ready

All of the paper tape records needed to load the Paper Tape Monitor System to disk storage are supplied to the user by IBM. These records have the same functions as the corresponding IBM-supplied and user-written card decks. These functions are described under System Generation Operating Procedures (Card System).

The Load Mode Control record and System Configuration records are supplied by IBM to the user of the Paper Tape System. These tapes are supplied with all the possible configurations, and the user need only select the configuration for his particular use. If these tapes are not read correctly, the System Loader will give error messages (see Appendix A).

The tapes constituting the Paper Tape Monitor System are described below. The procedure for loading these tapes onto disk is described under Procedure for Initializing Disk Monitor System from Paper Tape.

Tape	Description
1	System Loader, Part 1
2	Load Mode Control Record (same function as Load Mode Control Card)
3	System Loader, Part 2
4	System Configuration Records (same function as System Configuration Deck)
5	Supervisor Tape (includes the Loader)
6	Disk Utility Program
7	FORTRAN Compiler
8	Assembler
9	ILS Control Records and Library Subroutines
10	Cold Start Paper Tape Record
11	DPIR Tape (includes core-image loader)

If FORTRAN and/or the assembler are not to be loaded during an initial load, the corresponding tapes (7 and/or 8) need not be read.

During a reload of system programs, tapes 1 through 5 must be read. If DUP, FORTRAN, and/or the assembler are not to be reloaded, the corresponding tapes (6, 7, and/or 8) need not be read. The procedures for reloading DUP, FORTRAN, and the assembler are the same as the card system procedures. Tape 9 need not be read during a reload operation.

DPIR PAPER TAPE LOAD OPERATING PROCEDURES

The procedure for loading and executing the DPIR (Disk Pack Initialization Routine) is as follows:

1. Insert the disk pack in the console cabinet.
2. Put the DPIR tape in the reader; position one of the delete codes that appear after the program name in the leader under the read starwheels.
3. Press IMM STOP, RESET, and PROGRAM LOAD on the console.
4. When the loader reads in and waits, position the DPIR tape.
5. Press PROGRAM START on the console.

From this point on, the operation is identical to the card load.

PROCEDURE FOR INITIALIZING DISK MONITOR SYSTEM FROM PAPER TAPE

To load the paper tape system onto disk, the operator must perform the following steps:

1. Place the System Loader, Part 1, (Tape 1) in the Paper Tape Reader. When loading tapes 1 through 9, position any one of the delete codes following the program name in the tape header under the read starwheels.
2. Press RESET on the console.
3. Press PROGRAM LOAD on the console. Tape 1 is read into core starting at location 0.

4. When a WAIT occurs (at 05BC), place the Load Mode Control tape (Tape 2) in the Paper Tape Reader.
5. Press PROGRAM START on the console.
6. When a WAIT occurs, place the next system tape in the Paper Tape Reader.
7. Press PROGRAM START on the console.
8. Repeat steps 6 and 7 until the last system tape is read.

The System Loader determines if the complete system has been loaded. If the system has not been loaded, the System Loader WAITs for another tape to be readied by the operator until the complete system is loaded.

A WAIT at 0EA6 is a checksum error, indicating faulty tape.

COLD START OPERATING PROCEDURE

The procedure for executing the Cold Start paper tape record is as follows:

1. Insert the monitor disk pack into the console cabinet.
2. Turn the File Switch on, and wait for the File Ready light on the console to go on.
3. Put the Cold Start paper tape record into the reader; position any one of the delete codes following the program name in the tape leader under the read starwheels.
4. Press IMM STOP, then RESET on the console.
5. Press PROGRAM LOAD on the console.

PAPER TAPE CONTROL RECORDS

Paper tape control records must be punched in PTTC/8 (perforated tape transmission code). The formats are the same as the previously-described card formats. Paper tape control records must be separated by one NL (new line) control character. A control record which immediately follows paper tape data not followed by an NL code must be preceded by one NL code. Delete codes may precede or follow this NL code.

Table A-1. System Loader Error Codes

Error Code	Type of Error	Corrective Action
E1	Check-sum error.	Follow Procedure A or reload and restart.
E2	Illegal card type or blank card.	Follow Procedure A or reload and restart.
E3	Card out of sequence.	Follow Procedure A or reload and restart.
E4	ORG backward to an address lower than that established by last sector break card.	Inspect deck for card(s) missing or out of sequence. Correct deck and reload edit program.
E5	Error in Load Mode card.	Make necessary card correction and reload edit program.
E6	Disk error.	Press PROGRAM START on console to retry.
E7	Disk pack not initialized or Sector 0 data damaged.	Use DPIR program to initialize Sector 0. Initial load should follow since DPIR clears the disk.
E8*	Configuration deck missing or one of the following errors detected: a) SCON card not followed by REQ cards. b) less than 2 REQ cards present. c) more than 6 REQ cards present. d) Secondary Interrupt Branch Address (IBA) not included in ISS #1 card. e) Secondary IBA not equal to 12. f) Primary IBA not in range 8 through 12. g) ISS number missing or negative. h) ISS number 5 detected (illegal). j) ISS number greater than 7. k) TERM card missing	Make corrections and reload edit program.
E9**	File protect address (in COMMA) prohibits loading System Loader, Part 2.	No recovery unless file protect address can be lowered by deleting part of material on disk. System Loader requires temporary use of Cylinders 198 and 199.
E10**	During reload, old FORT or ASM address in COMMA is different from new FORT or ASM sector address.	If COMMA has been damaged, an initial load is required; otherwise system programs deck is faulty.
E11**	Fixed area or Core Image Buffer area, as defined by COMMA, is about to be overlayed.	Same as E10.

* Applies to initial load only.

** Applies to reload only.

Procedure A:

1. Lift remaining cards from hopper and depress NPRO on 1442.
2. Place the two ejected cards (after corrections) in card hopper.
3. Replace remaining cards in card hopper.
4. Press START on 1442.
5. Press PROGRAM START on console.

Table A-2. System Loader Wait Locations (Part 1)

Address	Explanation
01C7	Wait after displaying E6 error
053D	Wait after displaying E1 error
06B0	Wait after displaying E3 error
0806	Wait after displaying E2 error
080B	Wait after displaying E8 error
0821	Wait after displaying E5 error
0835	Wait after displaying E9 error
0839	Wait after displaying E8 error
083D	Wait after displaying E4 error
08A4	Wait after displaying E7 error
0962	Wait after displaying E4 error
0EE6	Wait during loading of the System Loader due to incorrect check sum, e.g., a missing card or card out of sequence.

Table A-3. System Loader Wait Locations (Part 2)

Address	Explanation
0220	Wait after displaying E 10 error
022F	Wait after displaying E 10 error
0245	Wait after displaying E 11 error
025F	Wait after displaying E 1 error
027D	Wait after displaying E 3 error
05C5	Wait after displaying E 2 error
0750	Wait after displaying E 5 error
0816	Wait after displaying E 3 error
08B6	Wait after displaying E 4 error
0991	Wait after displaying E 2 error
0AD7	Wait after displaying E 6 error
0FFF	Wait after displaying END reload

Table A-4. Monitor Supervisor Error Messages

Error Message	Cause of Error
M 01 PHASE NONX	Execution is not permitted for this job.
M 02 INVALID	The above listed record is an invalid Supervisor record.
M 03 NON XEQ	The currently called execution is not permitted.
M 04 CHARACTER	A character in the name listed above is not permitted.
M 05 OFLO DISK	The records listed above were too many for the disk storage allocated.
M 06 NO PROGRAM	The mainline program name listed above is not in the LET or FLET table or is not a mainline program.
M 07 NON DUP	DUP is not allowed for the subjob.
M 09 RECORD TRAP	A system program detected a Supervisor record and returned control to the Supervisor.
M 11 NOT IDENT	The cartridge identifier on the cartridge is not identical to the one on the input record. The Supervisor waits to allow the operator to rectify the difference if desired.
M 12 SEQ ERROR	LOCAL, NOCAL, and/or FILES records are intermixed (they must be grouped).
M 13 T ERROR	Column 8 in the JOB record does not contain a blank or a T. An ampersand is printed in place of the illegal character. The Supervisor waits so that the operator can (1) correct the JOB record, reload it in the reader, and press PROGRAM START on the console; or (2) press PROGRAM START on the console. In either case the JOB record is processed completely before any other processing. The job is considered non-temporary if column 8 contains a blank or a character other than a T.

Table A-5. Monitor Supervisor Wait Locations

Address	Explanation	Operator Action
0005	Operator pressed PROGRAM STOP on the console.	Press PROGRAM START to continue.
0029	I/O error or device not-ready condition.	Refer to Subroutine Library - Preoperative Errors.
00DE	Disk error.	Press PROGRAM START to retry.
07E7	1. Pause due to PAUS control record.	1. Press PROGRAM START to continue.
	2. Identifier error in JOB control record.	2. Correct the record, reenter it, and press PROGRAM START; or press PROGRAM START.
0399	Paper tape reader not ready.	Ready paper tape reader and press PROGRAM START.
07D5	Column 8 in the JOB record does not contain a blank or a T.	Correct the record, reenter it, and press PROGRAM START; or press PROGRAM START. In either case, the current job is processed first.

Table A-6. Loader Messages/Error Messages (Part 1)

Code and Message	Explanation and Recovery Procedures
R 01 ORIGIN BELOW 1ST WORD OF MAINLINE	The Loader has been instructed to load a word into an address lower than that of the first word of the mainline program. The ORG statement which caused this situation must be removed, or the mainline program must start at a lower address.
*R 03 LOAD REQUIRES SYSTEM LOCALS, LEVEL 1	No error. The load was too long to fit into core. The Loader has made two overlays, and the program will be executed with these two groups of routines overlaying each other (refer to System Overlays).
*R 04 LOAD REQUIRES SYSTEM LOCALS, LEVEL 2	No error. The load was too long to fit into core. The Loader has made three overlays, and the program will be executed with these three groups of routines overlaying one another (refer to System Overlays).
R 06 FILE(S) TRUNCATED (SEE FILE MAP)	At least one defined file has been truncated either because the previously defined storage area in the User or Fixed area was inadequate or because there is inadequate Working Storage available to store the file. See Message R 12 for a possible remedy.
R 08 CORE LOAD EXCEEDS 32K	The Loader has been instructed to load a word into an address exceeding 32, 767, which is a negative number. The loading process is immediately terminated, because the Loader cannot process negative addresses. This error was probably caused by bad data, i.e., the program being loaded from the disk has been destroyed.
R 10 LIBF TV REQUIRES 82 OR MORE ENTRIES	There are at least 82 different entry points referenced in the load by LIBF statements. A possible remedy would be to subdivide the load into two or more links.
R 11 TOO MANY ENTRIES IN LOAD-TIME TV	There are more than 135 references to different entry points with CALL and/or LIBF statements in the load. A possible remedy would be to subdivide the load into two or more links.
R 12 LOCALS/SOCALS EXCEED WKNG. STORAGE	There is insufficient Working Storage remaining to accommodate the LOCAL and/or SOCAL overlays required in the load. A possible remedy would be to create more Working Storage by deleting subroutines, subprograms, and/or data no longer required by the installation.
R 13 DEFINED FILE(S) EXCEED WKNG. STORAGE	There is insufficient Working Storage remaining to accommodate even one record of the defined file(s). See Message R 12 for a possible remedy.
**R 16 XXXXX IS NOT IN LET OR FLET	The program or data file designated in the message cannot be found in LET or FLET. A possible remedy is to store the program or data file. If the name cannot be explained otherwise, the program being loaded has probably been destroyed.
**R 17 XXXXX CANNOT BE DESIGNATED A LOCAL	The routine named in this message is either a type which cannot appear on a LOCAL record, or this routine, which is a LOCAL, has been referenced, directly or indirectly, by another LOCAL, the name of which cannot be supplied by the Loader.
**R 19 XXXXX IS NOT ON A SECTOR BOUNDARY	The area named in this message does not begin at a sector boundary, which implies that it is not a storage area but a relocatable program, and thus a possible error. Choose another area for the storage of this file.
**R 20 XXXXX COMMON EXCEEDS THAT OF ML	The length of COMMON for the routine named in this message is longer than that of the mainline program. A possible remedy is to define more COMMON for the mainline program.
**R 21 XXXXX PRECISION DIFFERENT FROM ML	The precision for the routine named in this message is incompatible with that of the mainline program. Make the precisions compatible.
**R 22 XXXXX AND ANOTHER VERSION REFERENCED	At least two different versions of the same I/O routine have been referenced, e.g., both CARDZ and CARD0 (FORTRAN utilizes the "Z" version). If a disk routine is named in the message, it is possible that the XEQ record specifies one version, e.g., DISK0, whereas the program references another, e.g., DISK1 (a blank in col. 19 of the XEQ record causes DISK0 to be chosen).
**R 23 XXXXX IS A USER AREA FILE REFERENCE	The area named in this message is in the User area; references in DEFINE FILE and DSA statements for *STORECI functions must be to the Fixed area.

*FORTRAN mainline programs only
**XXXXX = the name of the program or disk file concerned

Table A-6. Loader Messages/Error Messages (Part 2)

Code and Message	Explanation and Recovery Procedures
*R 24 XXXXX IS BOTH A LIBF AND A CALL	The routine named in this message has been either referenced improperly, i.e., CALL instead of LIBF or vice versa, or has been referenced in both CALL and LIBF statements. The only remedy is to reference the routine properly. NOTE: NOCALs must be CALL-type routines, i.e., type 4 or 6 routines (refer to Appendix B).
*R 25 XXXXX HAS MORE THAN 14 ENTRY POINTS	This message usually indicates that the routine has been destroyed since no routine is stored with more than 14 entry points.
*R 26 XXXXX HAS AN INVALID TYPE CODE	The routine named in this message has either been designated on an XEQ record and is not a mainline program, indicating a mistake has probably been made in preparing the XEQ record, or contains a type code other than 3 (subroutine), 4 (functional), 5 (ISS), or 6 (ILS), in which case the routine has probably been destroyed. This error could also be caused by a DSA statement referencing a program which is in Disk System format, or a CALL or LIBF referencing a program in Core Image or Disk Data format.
*R 27 XXXXX LOADING HAS BEEN TERMINATED	The loading of the mainline program named in this message has been terminated as a result of the detection of the error(s) listed in the messages preceding this one.
**R 32 XXXXX CANNOT REF'CE THE LOCAL XXXXX	The routine named first in this message has referenced the routine named second, which is a LOCAL. Either the first named routine is a LOCAL or it is entered (directly or indirectly) from a LOCAL. Neither case can be allowed for it could cause a LOCAL to be overlaid by another LOCAL before the first LOCAL has been completely executed.
R 40 XXXX (HEX) = ADDITIONAL CORE REQUIRED	If the load was executed, $XXXX_{16}$ is the number of words by which it exceeded core storage before the Loader made it fit by creating special overlays (SOCALs); if the load was not executed, the first occurrence of the message is as described and the record indicates the number of words by which it exceeds core storage even after creating the deepest level of special overlays. A possible solution to the latter problem is to create two or more links or LOCALs.
R 41 XXXX (HEX) TOO MANY WDS IN COMMON	The length of COMMON specified in the mainline program plus the length of the core load exceeds core storage by $XXXX_{16}$ words.
R 42 XXXX (HEX) IS THE EXECUTION ADDR	No error. This message follows every successful conversion from Disk System format to Core Image format provided a core map is requested.
R 43 XXXX (HEX) = ARITH/FUNC OVERLAY SIZE	No error. It has been necessary to employ the special overlays (SOCALs), and $XXXX_{16}$ is the length of the arithmetic/functional overlay (refer to System Overlays).
R 44 XXXX (HEX) = FI/O + I/O OVERLAY SIZE	No error. It has been necessary to employ the special overlays (SOCALs), and $XXXX_{16}$ is the length of the FORTRAN I/O, I/O, and conversion routine overlay (refer to System Overlays).
R 45 XXXX (HEX) = DISK FI/O OVERLAY SIZE	No error. It has been necessary to employ the special overlays (SOCALs), and $XXXX_{16}$ is the length of the Disk FORTRAN I/O overlay, including the 320-word buffer.
R 46 XXXX (HEX) = AN ILLEGAL ML LOAD ADDR	$XXXX_{16}$ is the address at which the loader has been requested to start loading the mainline program, but this address is lower than the highest address occupied by the version of Disk I/O requested for this load. Either make the mainline origin higher or request a shorter version of Disk.
R 47 XXXX (HEX) WORDS AVAILABLE	No error. $XXXX_{16}$ is the number of words of core storage not occupied by this core load.

*XXXXX = the name of the program or disk file concerned
**XXXXX = the name of the program concerned

Table A-7. Assembler Error Messages

Error Code and Error Message	Cause of Error	Corrective Action
A 01 MINIMUM W. S. NOT AVAILABLE--- ASSEMBLY TERMINATED	Less than 33 sectors of Working Storage are available at the beginning of the assembly.	Perform a DUP DELETE to expand Working Storage to a minimum of 33 sectors before attempting further assemblies.
A 02 SYMBOL TABLE OVERFLOW EXCEEDS 4 CYLINDERS	Symbol table overflow exceeds 3392 symbols (refer to Assembler Messages and Error Codes to compute number of symbols allowed in a program).	1. Reduce number of symbols and reassemble. 2. Divide program into segments and assemble each separately.
A 03 DISK OUTPUT EXCEEDS W.S.	Disk output is greater than Working Storage.	1. If error occured during pass 1, the assembler will wait at 0AD6. When PROGRAM START is pressed, the assembly will continue in the two-pass mode. Therefore, the operator should first insure that the source statements can be read a second time without encountering the next monitor control record. 2. If error occurred during pass 2, object output exceeds Working Storage. Perform a DUP DELETE to enlarge Working Storage.
A 04 SAVE SYMBOL TABLE INHIBITED	With SAVE SYMBOL TABLE option, symbol table exceeds the allowable System Symbol Table size of 100 symbols, or at least one assembly error was detected.	Reduce number of symbols and/or correct the erroneous statements and reassemble.

Table A-8. FORTRAN Error Codes (Part 1)

Error Number	Cause of Error
C 1	Non-numeric character in statement number.
C 2	More than five continuation cards, or continuation card out of sequence.
C 3	Syntax error in CALL LINK or CALL EXIT statement.
C 4	Undeterminable, misspelled, or incorrectly formed statement.
C 5	Statement out of sequence.
C 6	Statement following transfer statement or a STOP statement does not have statement number.
C 7	Name longer than five characters, or name not starting with an alphabetic character.
C 8	Incorrect or missing subscript within dimension information (DIMENSION, COMMON, or type).
C 9	Duplicate statement number.
C 10	Syntax error in COMMON statement.
C 11	Duplicate name in COMMON statement.
C 12	Syntax error in FUNCTION or SUBROUTINE statement.
C 13	Parameter (dummy argument) appears in COMMON statement.
C 14	Name appears twice as a parameter in SUBROUTINE or FUNCTION statement.
C 15	*IOCS control record in a subprogram.
C 16	Syntax error in DIMENSION statement.
C 17	Subprogram name in DIMENSION statement.
C 18	Name dimensioned more than once, or not dimensioned on first appearance of name.
C 19	Syntax error in REAL, INTEGER, or EXTERNAL statement.
C 20	Subprogram name in REAL or INTEGER statement.
C 21	Name in EXTERNAL which is also in a COMMON or DIMENSION statement.
C 22	IFIX or FLOAT in EXTERNAL statement.
C 23	Invalid real constant.
C 24	Invalid integer constant.
C 25	More than 15 dummy arguments, or duplicate dummy argument in statement function argument list.
C 26	Right parenthesis missing from a subscript expression.
C 27	Syntax error in FORMAT statement.
C 28	FORMAT statement without statement number.
C 29	Field width specification >145.

Table A-8. FORTRAN Error Codes (Part 2)

Error Number	Cause of Error
C 30	In a FORMAT statement specifying E or F conversion, w > 127, d > 31, or d > w.
C 31	Subscript error in EQUIVALENCE statement.
C 32	Subscripted variable in a statement function.
C 33	Incorrectly formed subscript expression.
C 34	Undefined variable in subscript expression.
C 35	Number of subscripts in a subscript expression does not agree with the dimension information.
C 36	Invalid arithmetic statement or variable; or, in a FUNCTION subprogram the left side of an arithmetic statement is a dummy argument (or in COMMON).
C 37	Syntax error in IF statement.
C 38	Invalid expression in IF statement.
C 39	Syntax error or invalid simple argument in CALL statement.
C 40	Invalid expression in CALL statement.
C 41	Invalid expression to the left of an equal sign in a statement function.
C 42	Invalid expression to the right of an equal sign in a statement function.
C 43	In an IF, GO TO, or DO statement a statement number is missing, invalid, incorrectly placed, or is the number of a FORMAT statement.
C 44	Syntax error in READ or WRITE statement.
C 45	*IOCS record missing with a READ or WRITE statement (mainline program only).
C 46	FORMAT statement number missing or incorrect in a READ or WRITE statement.
C 47	Syntax error in input/output list; or an invalid list element; or, in a FUNCTION subprogram, the input list element is a dummy argument or in COMMON.
C 48	Syntax error in GO TO statement.
C 49	Index of a computed GO TO is missing, invalid, or not preceded by a comma.
C 50	*TRANSFER TRACE or *ARITHMETIC TRACE control record present, with no *IOCS control record in a mainline program.
C 51	Incorrect nesting of DO statements; or the terminal statement of the associated DO statement is a GO TO, IF, RETURN, FORMAT, STOP, PAUSE, or DO statement.
C 52	More than 25 nested DO statements.
C 53	Syntax error in DO statement.
C 54	Initial value in DO statement is zero.

Table A-8. FORTRAN Error Codes (Part 3)

Error Number	Cause of Error
C 55	In a FUNCTION subprogram the index of DO is a dummy argument or in COMMON.
C 59	Syntax error in STOP statement.
C 60	Syntax error in PAUSE statement.
C 61	Integer constant in STOP or PAUSE statement is > 9999.
C 62	Last executable statement before END statement is not a STOP, GO TO, IF, CALL LINK, CALL EXIT, or RETURN statement.
C 63	Statement contains more than 15 different subscript expressions.
C 64	Statement too long to be scanned, because of compiler expansion of subscript expressions or compiler addition of generated temporary storage locations.
C 65*	All variables are undefined in an EQUIVALENCE list.
C 66*	Variable made equivalent to an element of an array, in such a manner as to cause the array to extend beyond the origin of the COMMON area.
C 67*	Two variables or array elements in COMMON are equated, or the relative locations of two variables or array elements are assigned more than once (directly or indirectly).
C 68	Syntax error in an EQUIVALENCE statement; or an illegal variable name in an EQUIVALENCE list.
C 69	Subprogram does not contain a RETURN statement, or a mainline program contains a RETURN statement.
C 70	No DEFINE FILE in a mainline program which has disk READ, WRITE, or FIND statements.
C 71	Syntax error in DEFINE FILE.
C 72	Duplicate DEFINE FILE, more than 75 DEFINE FILES, or DEFINE FILE in subprogram.
C 73	Syntax error in record number of READ, WRITE, or FIND statement.

*The detection of a code 65, 66, or 67 error prevents any subsequent detection of any of these three errors.

Table A-9. DUP Error Messages (Part 1)

Code and Printed Message	Description
D 01 NOT PRIME ENTRY	The primary name of the program in Working Storage does not match the name on the DUP control record.
D 02 INVALID TYPE	One of the following is detected: non-DSF program, mispositioned header, foreign data, or erroneous subtype.
D 03 INVALID HEADER LENGTH	Word six of the DSF header is outside the range of 3-45. The causes are similar to D 02, except for subtype.
D 05 SECONDARY ENTRY POINT NAME ALREADY IN LET IS....	The specified name is already in LET. The name must be deleted before this subprogram can be stored.
D 13 DCTL, FUNCTION	An invalid DUP function specified in columns 1-12 of the DUP control record.
D 14 DCTL, FROM FLD	Unacceptable characters are in columns 13 and 14 of the DUP control record. If Working Storage is specified in columns 13 and 14, then there is no valid program in Working Storage, i.e., the Working Storage Indicator has been set to zero, thus inhibiting the movement of programs from Working Storage.
D 15 DCTL, TO FIELD	Unacceptable characters are in columns 17 and 18 of the DUP control record.
D 16 DCTL, NAME FLD	If this is a *STORE control record, then the name is already in LET/FLET. If this is a *DUMP control record, then the name is not found in LET or FLET. If this is a *DUMP control record of Working Storage to the principal I/O, then a name is required in columns 21 through 25 of the DUP control record.
D 17 DCTL, COUNT	Columns 27 through 30 are blank or include alphabetic characters. The count field requires a decimal number.
D 18 DCTL, TMP MODE	This function is not allowed during the JOB T mode.
D 41 FIXED AREA PRESENT	The FORTRAN compiler and/or assembler cannot be eliminated if a Fixed area has been previously defined.
D 42 ASSEMBLER NOT IN SYSTEM	The assembler has previously been eliminated from the system.
D 43 FORTRAN NOT IN SYSTEM	The FORTRAN compiler has previously been eliminated from the system.
D 44 INCREASE VALUE IN COUNT FIELD	The count field was read as a value of zero or one. The first DEFINE requires one cylinder for FLET plus one cylinder of Fixed area. Thereafter, as little as one cylinder of additional Fixed area can be defined.
D 45 EXCEEDS WORK STORAGE	The initiation or expansion of the Fixed area is limited to the Working Storage available.
D 61 DUPCO, EXCEEDS WORK STORAGE	This function requires more Working Storage than is available.
D 62 EXCEEDS WORK STORAGE	The Working Storage area is not large enough to contain the program specified.
D 64 EXCEEDS FIXED AREA	There is insufficient room in the Fixed area for the program.
D 71 SEQUENCE OR CKSUM	The cards are out of sequence, or there was an erroneous checksum.
D 72 LOAD BLANK CARDS	More blank cards are required to complete the dump. The operator performs an NPRO and places blank cards between the two cards ejected, removes the first card, places the first card in the output stacker, places the remainder in front of the cards still in the reader hopper, and presses the reader START button.
D 82 NON FILES RECORD	The first six characters of records following *STORECI are not *FILES. The number of *FILES records is determined by the count field of DUP control record STORECI.
D 83 INVALID CHARACTER	The *FILES record following the *STORECI DUP control record has an invalid character.
D 84 EXCEEDS SECTOR ALLOCATION	Too many Files have been defined. More than two sectors are required to contain the information from the *FILES record.

Table A-9. DUP Error Messages (Part 2)

Code and Printed Message	Description
D 92 INVALID CI CONVERT	The Loader has inhibited the continuation of *STORECI. The specific reason has been printed by the Loader.
D 94 LET/FLET OVERFLOW	A ninth sector of LET/FLET is required for the LET/FLET entry. A deletion of a program with a LET/FLET entry of similar size is required before this program can be stored.

NOTE: DCTL means the error was detected in the DUP control record. DUPCO means the error was detected in the DUP common section.

Table A-10. DUP Waits and Loops

Address	Explanation	Operator Action
Loops: 70FF@03A8 (occurs after console printer prints first 320 core positions)	System check. LET/FLET, COMMA, and DCOM do not agree.	Reload entire Monitor System.
Wait at 092C	Paper tape reader not ready.	Read paper tape reader and press PROGRAM START.
Wait at 002E	Operator pressed PROGRAM STOP on console.	To continue, branch to 0030; press PROGRAM START.

Table A-11. FORTRAN I/O Error Codes

The following error codes are displayed in the accumulator upon detection of the errors. Press PROGRAM START to perform a branch to the monitor to proceed to the next subjob.

Error Code	Cause of Error
F001	1. Logical unit defined incorrectly. 2. No *IOCS control record for specified I/O device.
F002	Requested record exceeds allocated buffer size.
F003	Illegal character encountered in input record.
F004	Exponent too large or too small in input field.
F005	More than one E encountered in input field.
F006	More than one sign encountered in input field.
F007	More than one decimal point encountered in input field.
F008	1. Read of output-only device. 2. Write of input-only device.
F100	File not defined by DEFINE FILE statement.
F101	File record too large, equal to zero, or negative.
DISKZ Errors:	
F102	Read error.
F104	Write error.
F106	Read back check error.
F108	Seek error.
F10A	Forced read error (seek or find).

DISK SYSTEM FORMAT (DSF)

Unless otherwise instructed, DUP automatically converts programs in Card System format (CDS) to Disk System format (DSF) when storing programs to disk storage. Likewise, programs in DSF are converted to CDS when dumping from disk storage. Disk System format is shown in Figure 11; Card System format is described elsewhere in this appendix.

Program Header Format

The contents of the program header record (see Figure 11) vary with the type of routine with which it is associated. The first 12 words of the program header record for the seven types of programs are identical except for word 6, which is 9 less than the number of words in the program header record. The format of these 12 words is as follows:

Word	Contents
1	Zero
2	Checksum if source was cards; otherwise zero
3	Type, subtype, precision
4	Effective length of program, i.e., the terminal address in the program
5	Length of COMMON (words)
6	Length of program header record minus 9
7	Zero
8	Number of files defined
9	Length of program, including program header record in disk blocks
10-11	Name of entry point 1 (see Appendix G)
12	Address of entry point 1 (absolute for type 1, relative to zero otherwise)

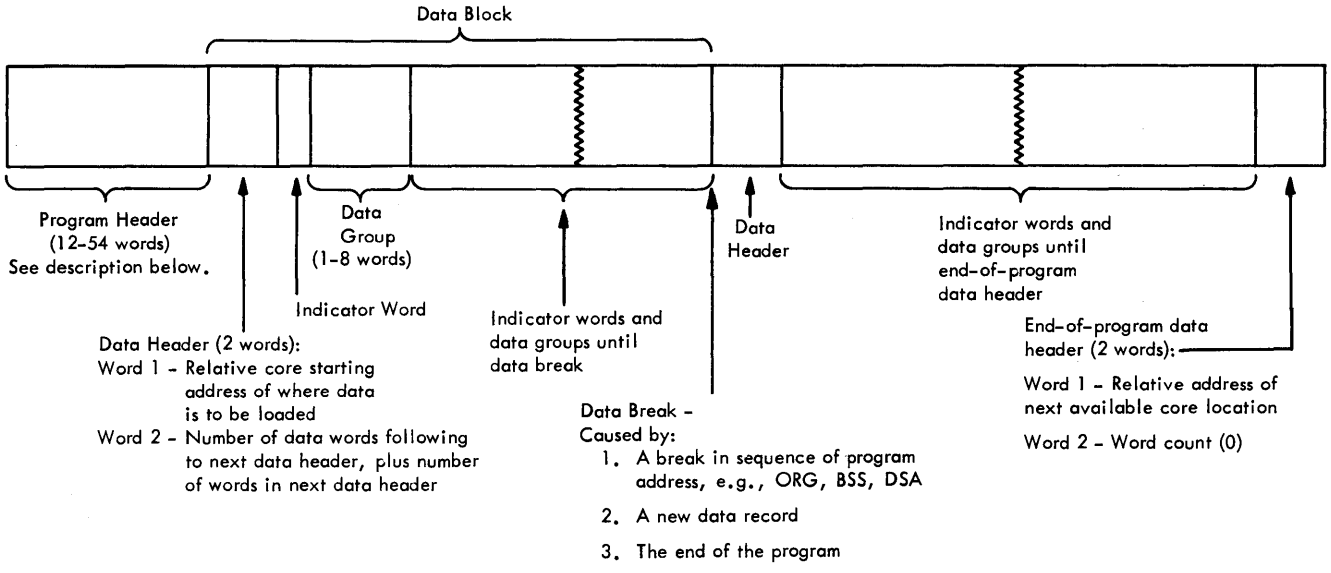


Figure 11. Disk System Format

After the first 12 words, the program header record format depends on the type of program. The header record for types 1 and 2 (absolute and relocatable mainline, respectively) consists of the first 12 words. The program types and their header record formats are shown below.

Program Types

Type Code	Type of Program
1	Mainline (absolute)
2	Mainline (relocatable)
3	Subroutine, not an ISS, referenced by LIBF
4	Subroutine, not an ISS, referenced by CALL
5	Interrupt service subroutine (ISS) referenced by LIBF
6	Interrupt service subroutine (ISS) referenced by CALL
7	Interrupt level subroutine (ILS)

Program Formats

Type 3, 4:

Words	Contents
13-14	Name of entry point 2 (30 bits, right-justified)
15	Address of entry point 2 (relative to zero)
17-18	Name of entry point 3 (30 bits, right-justified)
19	Address of entry point 3 (relative to zero)
20-54	Three words per entry point as above, to a maximum of 14 entry points. The header record ends at the last defined entry point; thus, it is of variable length

Type 5, 6:

Words	Contents
13	50 + ISS number
14	ISS number
15	Number of interrupt levels required
17	Interrupt level number associated with primary interrupt*
18	Interrupt level number associated with secondary interrupt

Type 7:

Words	Contents
13	Interrupt level number

*The 1442 Card Read Punch is the only device requiring more than one interrupt level.

Program Subtypes

Subtypes are defined only for type 3 and 4 subroutines. When undefined, the field contains a zero.

For type 3 subroutines, subtypes are defined as follows:

Subtype	Description
0	In-core subroutines. Of the IBM subroutine library, this group includes the trace, fix, float, dump, subscript, normalize, flipper, initialization, and certain conversion subroutines
1	Disk FORTRAN I/O subroutines, SDFIO and SDFND
2	Arithmetic subroutines, e.g., FADD
3	FORTRAN Format subroutine SFIO, and FORTRAN I/O subroutines, e.g., CARDZ
4	Conversion routines for the FORTRAN I/O subroutines, e.g., HOLEB

For type 4 subroutines, subtypes are defined as follows:

Subtype	Description
0	All type 4 subroutines which are not subtype 8, e.g., DMTD0
8	Functional subroutines, e.g., SIN

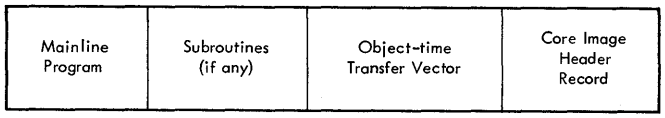
Appendix E lists all IBM-supplied subroutines and their subtypes.

DISK CORE IMAGE FORMAT (DCI)

A program in Disk Core Image format (DCI) is one that the Loader has converted from Disk System format (DSF). A DCI program is an entire core load, i.e., it consists of a mainline program, all subroutines referenced in the core load (except the Disk I/O routine), the object-time transfer vector, and the core image header record. The mainline program and subroutines appear as they will at execution time; however, the Loader must prepare the program for execution before it is read into core storage.

Although programs are loaded faster from DCI than from DSF, DCI programs usually occupy more disk storage because they constitute an entire core load. In addition, unlike DSF programs, the areas reserved by BSS and BES statements are a part of DCI programs unless the first statement in the mainline is a BSS or BES.

A typical DCI program is stored on disk in the User/Fixed area as follows:



The object-time transfer vector is described in the section titled The Loader. Information contained in the 60-word core image header record is used to load the DCI program into core before execution. The format is as follows:

Words	Description
1-6	Interrupt transfer vector (words 8-13 at execution time)
7	Setting for index register 3 at execution time
8	Core address (at execution time) of the subroutine ILS02
9	Number of files defined
10	Length of COMMON (in words)
11	Code for requested version of the Disk I/O subroutine (-1 = DISKZ, 0 = DISK0, 1 = DISK1, 2 = DISKN)
12	Core address (at execution time) of the entry in the LIBF TV which is associated with the Disk I/O subroutine
13	Length of the object-time transfer vector (in words)
14	Core address (at execution time) of the first word of the mainline program, exclusive of initial BSS and/or BES statements
15	Total length of mainline program, subroutines, and object-time transfer vector (in words)
16-60	Reserved

DISK DATA FORMAT (DDF)

Disk Data format (DDF) describes information placed in the User area, Fixed area, or Working Storage area as a result of the DUP control record STOREDATA. Disk Data format consists of 320 binary words per sector; there are no headers, trailers, or indicator words.

CARD SYSTEM FORMAT (CDS)

Card System format is in terms of words on binary cards (see Card Data Format). The card ID and sequence numbers (columns 73-80) are in IBM card code.

Mainline Header Card

A mainline header card specifies the size of the common area and the size of the work area. It is

the first card of the mainline program. The format is as follows:

Word	Contents
1	Reserved
2	Checksum*
3	Type code (first 8 bits): 0000 0001 - absolute 0000 0010 - relocatable Precision code (last 8 bits): 0000 0001 - standard 0000 0010 - extended 0000 0000 - undefined
4	Reserved
5	Length of COMMON storage area (FORTRAN mainline program only)
6	0000 0000 0000 0011
7	Work area required (FORTRAN only)
8-54	Reserved

*The checksum is the two's complement of the logical sum of the record count (position of the record within the deck) and the data word(s). The logical sum is obtained by summing the data word(s) and the record count arithmetically with the addition of a one each time a carry occurs out of the high-order position of the accumulator.

Data Cards

Data cards contain the instructions and data that constitute the assembled program. The format is as follows:

Word	Contents
1	Location (The relative load address of the first data word of the card or record. Succeeding words go into higher numbered core locations. The relocation factor must be added to this address to obtain the actual load address. For an absolute program the relocation factor is zero.)
2	Checksum
3	Type code (first 8 bits): 0000 1010 Data word count (last 8 bits)
4-9	Relocation indicators (2 bits per data word): 00 - nonrelocatable or absolute 01 - relocatable 10 - LIBF (one word call) 11 - CALL (two word call)
10	Data word 1
11-54	Date words 2 through 45

EOP Card

An EOP (end of program) card is the last card of each program and subroutine. The format is as follows:

Word	Contents
1	Starting location of next routine (this number is always even and is assigned by the assembler)
2	Checksum
3	Type code (first 8 bits): 0000 1111 Last 8 bits: 0000 0000
4	XEQ address, if mainline program
5-54	Reserved

Subroutine Header Card

A maximum of 14 entry points can be defined for each subroutine. The format of the subroutine header card is as follows:

Word	Contents
1	Reserved
2	Checksum
3	Type code (first 8 bits); 0000 0011 - to be called by a one-word call only (LIBF) 0000 0100 - to be called by a two-word call only (CALL) Precision code (last 8 bits): 0000 0000 - undefined 0000 0001 - standard 0000 0010 - extended
4-5	Reserved
6	Number of entry points times three
7-9	Reserved
10-11	Name of entry point 1
12	Relative address of entry point 1
13-51	Names and relative addresses of entry points 2 through 14
52-54	Reserved

ISS Header Card

An ISS (interrupt service subroutine) header card for each interrupt service subroutine identifies the entry point defined by an ISS statement. Only one entry point can be defined for each subroutine. The format of the ISS header card is as follows:

Word	Contents
1	Reserved
2	Checksum
3	Type code (first 8 bits): 0000 0101 - to be called by a one-word call only (LIBF) 0000 0110 - to be called by a two-word call only (CALL) Precision code (last 8 bits): 0000 0000 - undefined 0000 0001 - standard 0000 0010 - extended
4-5	Reserved
6	Six plus number of interrupt levels required
7-9	Reserved
10-11	Subroutine name
12	Relative entry address
13	Address of ISTV (interrupt service transfer vector) is equal to 51_{10} plus the ISS number.*
14	ISS number (displacement in ISTV) = $1\text{-}8_{10}$
15	Number of interrupt levels required
16	ID number for the primary interrupt level required (0-5)
17-29	ID numbers for remaining interrupt levels required (0-5)
30	Edit word (contains a 1)
31-54	Reserved

*The ISTV table is initialized by the Loader. This table starts at location 0034. Each TV entry in this table contains the starting addresses for the corresponding ISS routine (maximum of 8 TV entries).

ILS Header Card

An ILS (interrupt level subroutine) header card identifies the ILS routine. The format of the ILS header card is as follows:

Word	Contents
1	Reserved
2	Checksum
3	Type code (first 8 bits): 0000 0111 Reserved (last 8 bits)
4-5	Reserved
6	0000 0000 0000 0100
7-9	Reserved
10-12	Reserved
13	Interrupt level number
14-54	Reserved

CARD DATA FORMAT (CDD)

Card Data format (CDD) is shown in Figure 12. Fifty four words can be placed on a card (1-1/3 columns per word, 4 columns for 3 words). The word numbers appear in every third column across the top of the card.

PRINT DATA FORMAT (PRD)

Print Data format is shown in Figure 13. There are 16 four-character words per line, with a space after each word, and an additional space after each fourth word.

PAPER TAPE SYSTEM (PTS) AND PAPER TAPE DATA (PTD) FORMATS

Paper Tape System format (PTS) is analogous to Card System format (CDS), and Paper Tape Data format (PTD) is analogous to Card Data format (CDD).

In Paper Tape format, two frames contain one binary word, which is equivalent to 16 bits per

binary word in Card Data format. In addition, a one-frame word count precedes a paper tape record. A paper tape data record contains a maximum of 54 binary words, i.e., 108 frames plus a word-count frame.

Information that would appear in columns 73-80 of a card must not appear on paper tape.

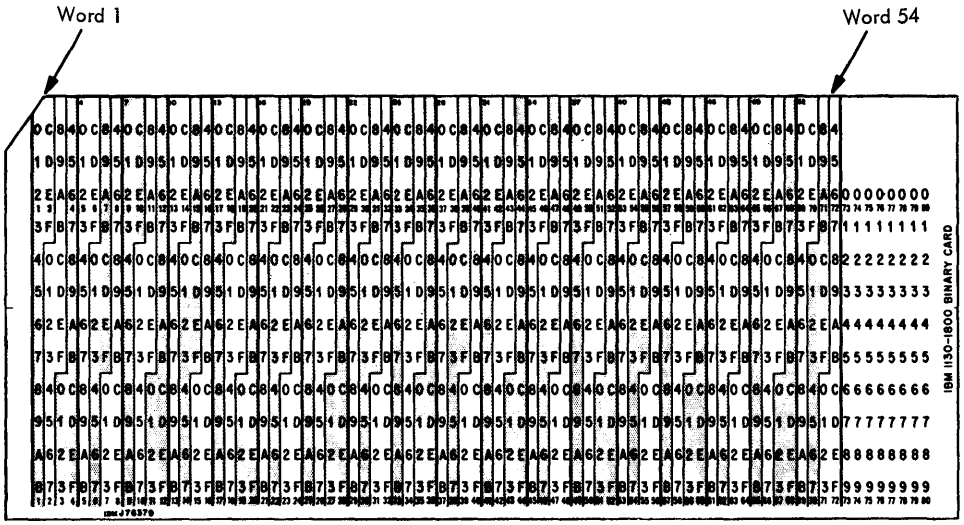


Figure 12. Card Data Format

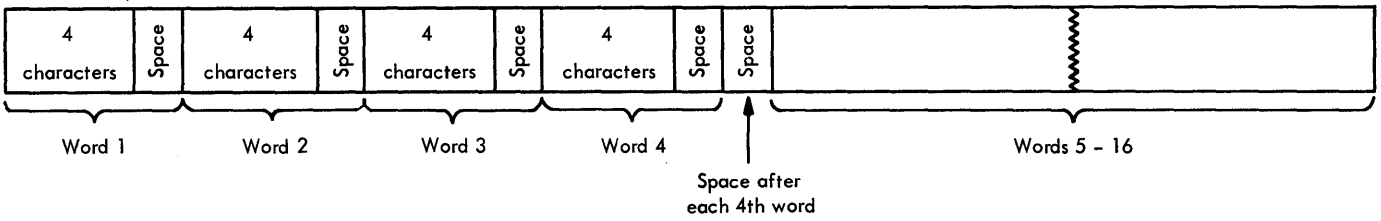


Figure 13. Print Data Format

APPENDIX C. DISK STORAGE UNIT CONVERSION FACTORS

No. Of / Per:	Word	Disk Block	Sector	Track	Cylinder	Disk
Bits	16	320	5,112	20,480	40,960	8,192,000
Data Words		20	320*	1,280	2,560	512,000
Disk Block			16	64	128	25,600
Sectors				4	8	1,600
Tracks					2	400
Cylinders						200

*These follow the first actual word of each sector, which is used for the address.

APPENDIX D. SUPERVISOR AND DUP INPUT/OUTPUT CHARACTER CODES

Keyboard Graphic	1132 Graphic	IBM Card Code	PTTC/8 Hex (U = Upper Case) (L = Lower Case)
	Numeric Characters		
0	0	0	1A (L)
1	1	1	01 (L)
2	2	2	02 (L)
3	3	3	13 (L)
4	4	4	04 (L)
5	5	5	15 (L)
6	6	6	16 (L)
7	7	7	07 (L)
8	8	8	08 (L)
9	9	9	19 (L)
	Alphabetic Characters		
A	A	12-1	61 (U)
B	B	12-2	62 (U)
C	C	12-3	73 (U)
D	D	12-4	64 (U)
E	E	12-5	75 (U)
F	F	12-6	76 (U)
G	G	12-7	67 (U)
H	H	12-8	68 (U)
I	I	12-9	79 (U)
J	J	11-1	51 (U)
K	K	11-2	52 (U)
L	L	11-3	43 (U)
M	M	11-4	54 (U)
N	N	11-5	45 (U)
O	O	11-6	46 (U)
P	P	11-7	57 (U)
Q	Q	11-8	58 (U)
R	R	11-9	49 (U)
S	S	0-2	32 (U)
T	T	0-3	23 (U)
U	U	0-4	34 (U)
V	V	0-5	25 (U)
W	W	0-6	26 (U)
X	X	0-7	37 (U)
Y	Y	0-8	38 (U)
Z	Z	0-9	29 (U)

Keyboard Graphic	1132 Graphic	IBM Card Code	PTTC/8 Hex (U = Upper Case) (L = Lower Case)
	Special Characters		
.	.	12-8-3	6B (L)
<	)	12-8-4	02 (U)
(	(	12-8-5	19 (U)
+	+	12-8-6	70 (U)
&	&	12	70 (L)
$	$	11-8-3	5B (L)*
*	*	11-8-4	08 (U)*
)	)	11-8-5	1A (U)
-	-	11	40 (L)
/	/	0-1	31 (L)*
,	,	0-8-3	3B (L)
%	(	0-8-4	15 (U)
#	Blank	8-3	0B (L)*
@	Blank	8-4	20 (L)*
'	'	8-5	16 (U)
=	=	8-6	01 (U)
Space	Blank	Blank	10 ( )*
ç	Blank	12-2-8	A0 (U)
!	Blank	12-7-8	BB (U)
¦	Blank	11-2-8	DB (U)
;	Blank	11-6-8	9B (U)
¬	Blank	11-7-8	EB (U)
%	Blank	0-5-8	95 (L)
-	Blank	0-5-8	C0 (U)
>	Blank	0-6-8	8F (U)
?	Blank	0-7-8	B1 (U)
:	Blank	2-8	84 (U)
"	Blank	7-8	8B (U)

NOTES:
1. DUP recognizes only those special characters flagged with an asterisk.
2. Any special characters not recognized by SUP and DUP will be corrected to an ampersand (&).

APPENDIX E. 1130 SUBROUTINE LIBRARY LISTING

Subroutines	Names	Sub-type	Other Subroutines Required
Utility Calls			
Selective Dump on Console Printer	DMTD0, DMTX0	0	WRTY0
Selective Dump on 1132 Printer	DMPD1, DMPX1	0	PRNT1
Dump 80 Routine	DMP80	0	None
Common FORTRAN Calls			
Test Data Entry Switches	DATSW	8	None
Divide Check Test	DVCHK	8	None
Functional Error Test	FCTST	8	None
Overflow Test	OVERF	8	None
Sense Light Control and Test	SLITE, SLITT	8	None
FORTRAN Trace Stop	TSTOP	8	TSET
FORTRAN Trace Start	TSTRT	8	TSET
Integer Transfer of Sign	ISIGN	8	None
Extended Arith/Funct Calls			
Extended Precision Hyperbolic Tangent	ETANH, ETNH	8	EEXP, ELD/ESTO, EADD, EDIV, EGETP
Extended Precision A**B Function	EAXB, EAXBX	8	EEXP, ELN, EMPY
Extended Precision Natural Logarithm	ELN, EALOG	8	XMD, EADD, EMPY, EDIV, NORM, EGETP
Extended Precision Exponential	EEXP, EXPN	8	XMD, FARC, EGETP
Extended Precision Square Root	ESQR, ESQRT	8	ELD/ESTO, EADD, EMPY, EDIV, EGETP
Extended Precision Sine-Cosine	ESIN, ESINE, ECOS, ECOSN	8	EADD, EMPY, NORM, XMD, EGETP
Extended Precision Arctangent	EATN, EATAN	8	EADD, EMPY, EDIV, XMD, EGETP, NORM
Extended Precision Absolute Value Function	EABS, EAVL	8	EGETP
FORTRAN Sign Transfer Calls			
Extended Precision Transfer of Sign	ESIGN	8	ESUB, ELD
Standard Precision Transfer of Sign	FSIGN	8	FSUB, FLD
Standard Arith/Funct Calls			
Standard Precision Hyperbolic Tangent	FTANH, FTNH	8	FEXP, FLD/FSTO, FADD, FDIV, FGETP
Standard Precision A**B Function	FAXB, FAXBX	8	FEXP, FLN, FMPY
Standard Precision Natural Logarithm	FLN, FALOG	8	FSTO, XMDS, FADD, FMPY, FDIV, NORM, FGETP
Standard Precision Exponential	FEXP, FXPN	8	XMDS, FARC, FGETP
Standard Precision Square Root	FSQR, FSQRT	8	FLD/FSTO, FADD, FMPY, FDIV, FGETP
Standard Precision Sine-Cosine	FSIN, FSINE, FCOS, FCOSN	8	FADD, FMPY, NORM, XMDS, FSTO, FGETP

Subroutines	Names	Sub-type	Other Subroutines Required
Standard Precision Arctangent	FATN, FATAN	8	FADD, EMPY, FDIV, XMDS, FSTO, FGETP
Standard Precision Absolute Value Function	FABS, FAVL	8	FGETP
Common Arith/Funct Calls			
Fixed Point (Fractional) Square Root	XSQR	8	None
Integer Absolute Function	IABS	8	None
Floating Binary/EBC Decimal Conversions	FBTD (BIN. TO DEC.) FDTB (DEC. TO BIN.)	0	None
Overlay Routines for LOCAL Subprograms			
Long Form	FLIP0	0	DISKZ or DISK0
Short Form	FLIP1	0	DISK1 or DISKN
FORTRAN Trace Routines			
Extended Floating Variable Trace	SEAR, SEARX	0	ESTO, TTEST, SWRT, SIOF, SCOMP
Fixed Variable Trace	SIAR, SIARX	0	TTEST, SWRT, SIOI, SCOMP
Standard Floating IF Trace	SFIF	0	FSTO, TTEST, SWRT, SIOF, SCOMP
Extended Floating IF Trace	SEIF	0	FSTO, TTEST, SWRT, SIOF, SCOMP
Fixed IF Trace	SIIF	0	TTEST, SWRT, SIOI, SCOMP
Standard Floating Variable Trace	SFAR, SFARX	0	FSTO, TTEST, SWRT, SIOF, SCOMP
GOTO Trace	SGOTO	0	TTEST, SWRT, SIOI, SCOMP
Non-Disk FORTRAN Format I/O			
FORTRAN Format Routine	SFIO, SIOI, SIOAI, SIOF, SIOAF, SIOFX, SCOMP, SWRT, SRED, SIOIX	3	FLOAT, ELD/ESTO or FLD/FSTO, IFIX
FORTRAN Find Routine	SDFND	1	DISKZ
Disk FORTRAN I/O	SDFIO, SDRED, SDWRT, SDCOM, SDAF, SDF, SDI, SDIX, SDFX, SDAI	1	DISKZ
FORTRAN Common LIBFs			
FORTRAN Pause	PAUSE	2	None
FORTRAN Stop	STOP	2	None
FORTRAN Subscript Displacement Calculation	SUBSC	0	None
FORTRAN Subroutine Initialization	SUBIN	0	None
FORTRAN Trace Test and Set	TTEST, TSET	0	None
FORTRAN I/O and Conversion Routines			
FORTRAN Card Routine	CARDZ	3	HOLEZ
Disk I/O Routine	DISKZ	0	None
FORTRAN Paper Tape Routine	PAPTZ	3	None

Subroutines	Names	Sub-type	Other Subroutines Required
FORTRAN 1132 Printer Routine	PRNTZ	3	None
FORTRAN Keyboard-Typewriter Routine	TYPEZ	3	GETAD, EBCTB, HOLEZ
FORTRAN Typewriter Routine	WRTYZ	3	GETAD, EBCTB
FORTRAN Hollerith to EBCDIC Conversion	HOLEZ	3	GETAD, EBCTB, HOLTB
FORTRAN Get Address Routine	GETAD	3	None
FORTRAN EBCDIC Table	EBCTB	3	None
FORTRAN Hollerith Table	HOLTB	3	None
Extended Arith/Funct LIBFs			
Extended Precision Get Parameter Subroutine	EGETP	2	ELD
Extended Precision A**I Function	EAXI, EAXIX	2	ELD/ESTO, EMPY, EDVR
Extended Precision Divide Reverse	EDVR, EDVRX	2	ELD/ESTO, EDIV
Extended Precision Float Divide	EDIV, EDIVX	2	XDD, FARC
Extended Precision Float Multiply	EMPY, EMPYX	2	XMD, FARC
Extended Precision Subtract Reverse	ESBR, ESBRX	2	EADD
Extended Add-Subtract	EADD, ESUB, EADDX, ESUBX	2	FARC, NORM
Extended Load-Store	ELD, ELDX, ESTO, ESTOX	0	None
Standard Arith/Funct LIBFs			
Standard Precision Get Parameter Subroutines	FGETP	2	FLD
Standard Precision A**I Function	FAXI, FAXIX	2	FLD/FSTO, FMPY, FDVR
Standard Precision Divide Reverse	FDVR, FDVRX	2	FLD/FSTO, FDIV
Standard Precision Float Divide	FDIV, FDIVX	2	FARC
Standard Precision Float Multiply	FMPY, FMPYX	2	XMDS, FARC
Standard Precision Subtract Reverse	FSBR, FSBRX	2	FADD
Standard Add-Subtract	FADD, FSUB, FADDX, FSUBX	2	NORM, FARC
Standard Load-Store	FLD, FLDX, FSTO, FSTOX	0	None
Standard Precision Fractional Multiply	XMDS	2	None
Common Arith/Funct LIBFs			
Fixed Point (Fractional) Double Divide	XDD	2	XMD
Fixed Point (Fractional) Double Multiply	XMD	2	None
Sign Reversal Function	SNR	2	None
Integer to Floating Point Function	FLOAT	0	NORM
Floating Point to Integer Function	IFIX	0	None
I**J Integer Function	FIXI, FIXIX	2	None

Subroutines	Names	Sub-type	Other Subroutines Required
Normalize Subroutine	NORM	0	None
Floating Accumulator Range Check Subroutine	FARC	2	None
Interrupt Service Subroutines			
Card Input/Output (No Error Parameter)	CARD0	0	ILS00, ILS04
Card Input/Output (Error Parameter)	CARD1	0	ILS00, ILS04
One Sector Disk Input/Output	DISK0	0	ILS02
Multiple Sector Disk Input/Output	DISK1	0	ILS02
High-Speed Multiple Sector Disk Input/Output	DISKN	0	ILS02
Paper-Tape Input/Output	PAPT1	0	ILS04
Simultaneous Paper Tape Input/Output	PAPTN	0	ILS04
Plotter Output Routine	PLOT1	0	ILS03
1132 Printer Output Routine	PRNT1	0	ILS01
Keyboard/Console Printer Input/Output	TYPE0	0	HOL, PRTY, ILS04
Console Printer Output Routine	WRTY0	0	ILS04
Conversion Routines			
Binary Word to 6 Decimal Characters (Card Code)	BINDC	0	None
Binary Word to 4 Hexadecimal Characters (Card Code)	BINHX	0	None
6 Decimal Characters (Card Code) to Binary Word	DCBIN	0	None
EBCDIC to Console Printer Output Code	EBPRT	4	EBPA, PRTY
Card Code to EBCDIC-EBCDIC to Card Code	HOLEB	4	EBPA, HOLL
Card Code to Console Printer Output Code	HOLPR	0	HOLL, PRTY
4 Hexadecimal Characters (Card Code) to Binary Word	HXBIN	0	None
PTTC/8 to EBCDIC-EBCDIC to PTTC/8	PAPEB	4	EBPA
PTTC/8 to Card Code-Card Code to PTTC/8	PAPHL	0	EBPA,HOLL
PTTC/8 to Console Printer Output Code	PAPPR	0	EBPA, PRTY
Card Code to EBCDIC-EBCDIC to Card Code	SPEED	0	None
EBCDIC and PTTC/8 Table	EBPA	0	None
Card Code Table	HOLL	0	None
Console Printer Output Code Table	PRTY	0	None
Interrupt Level Subroutines			
Interrupt Level Zero Routine	ILS00		None

Subroutines	Names	Sub-type	Other Subroutines Required
Interrupt Level One Routine	ILS01		None
Interrupt Level Two Routine	ILS02		None
Interrupt Level Three Routine	ILS03		None
Interrupt Level Four Routine	ILS04		None

APPENDIX F. IN-CORE COMMUNICATIONS AREA (COMMA)

The Disk Communications Area (DCOM), sector 8 on the disk, is read into core starting in address 0028 (decimal 40). The In-Core Communications Area (COMMA), therefore, is an image of DCOM, but offset by 40 words. The first 10 words (0028-0031) include the IOCS error and interrupt error traps. The IOCS error trap entry word (0028) is initialized to contain the version and modification level of the Disk Monitor System, and will be overlayed by a return address if an IOCS error occurs.

The core locations and contents of COMMA are shown below.

Core Location		Comments
Dec.	Hex.	
1. 50	32	IOCS counter, incremented by 1 upon entry to every IOCS subroutine (provided an XIO is to be executed), decremented by 1 after an operation complete interrupt.
2. 51	33	Reserved.
3. 52	34	Core address (in user program) of Interrupt Level 2 subroutine (ILS02).
4. 53	35	Number of files defined.
5. 54-69	36-45	CALL LINK/CALL EXIT linkage to Skeleton Supervisor.
6. 70	46	Length of COMMON (in words).
7. 71	47	Type of Disk I/O required, e.g., -1 = DISKZ (special disk routine) 0 = DISK0 1 = DISKI 2 = DISKN
8. 72	48	Reserved.
		System Addresses
9. 73	49	Reserved.
10. 74	4A	Sector address of FORTRAN, zero if FORTRAN deleted.
11. 75	4B	Sector address of Assembly Program, zero if deleted.
12. 76	4C	320_{10} (length of 1 sector).
13. 77	4D	Sector address of first (numerically lowest) sector of Core Image Buffer (CIB). Word 1 of FLET header printed by DUMPLET.
14. 78	4E	000A (address in lower core, i.e., in the interrupt transfer vector, to which all disk interrupts branch indirectly).
15. 79	4F	Sector address of first (numerically lowest) sector of Fixed Location Equivalence Table (FLET). Word 2 of FLET header printed by DUMPLET.
16. 80	50	Sector address of first (numerically lowest) sector of Location Equivalence Table (LET).
17. 81	51	Sector address of first (numerically lowest) sector of User area.
18. 82	52	File protect sector address, otherwise used as sector address of Working Storage (base). Word 1 of LET header printed by DUMPLET. See Note 1.
19. 83	53	Same as above (adjusted). Word 2 of LET header printed by DUMPLET. See Note 1.
		LET/FLET Entries
20. 84	54	Total number of words used on disk by FLET. Word 3 of FLET header printed by DUMPLET.
21. 85	55	Reserved.
22. 86	56	Total number of words used on disk by LET (base). Word 5 of LET header printed by DUMPLET. See Note 1.
23. 87	57	Same as above (adjusted). Word 6 of LET header printed by DUMPLET. See Note 1.
24. 88	58	Next available disk block address in User area (base). Word 3 of LET header printed by DUMPLET. See Note 1.
25. 89	59	Same as above (adjusted). Word 4 of LET header printed by DUMPLET. See Note 1.
26. 90-91	5A-5B	Name of program (LET/FLET entry words 1 and 2).
27. 92	5C	Disk blocks used by program (LET entry word 3). Second word printed out at end of DUP function.

Core Location		Comments
Dec.	Hex.	LET/FLET Entries (Continued)
28. 93	5D	Core execution address of program (relative for relocatable programs, absolute otherwise) for word 12 of the program header record. Not used as word 4 of LET/FLET entry.
29. 94	5E	Core loading address of program, also used as first word on mainline program, or first word of define file table (LET/FLET entry word 5 for core image program).
30. 95	5F	Word count/disk block count of program, i.e., word count for core image programs, disk block count for data files, and address of next available core location following the program in Disk System format that is in Working Storage on the disk (LET/FLET entry word 6 for core image program data file).
		Switches
31. 96	60	Principal print device (odd = console printer/keyboard, even = 1132). See Note 2.
32. 97	61	Principal I/O device (odd = 1442, even = 1134/1055). See Note 2.
33. 98	62	Temporary mode if non-zero, normal if zero.
34. 99	63	Non-XEQ (disable XEQ until next // JOB record if non-zero, enable XEQ if zero). See Note 3.
35. 100	64	Non-DUP (disable DUP functions until next // JOB record is detected if non-zero, enable DUP functions if zero).
36. 101	65	System Overlays and/or LOCALs are used in program if non-zero.
37. 102	66	Disable Supervisor reading of monitor control record if non-zero, enable if zero (positive indicates monitor control record has been read under invalid conditions, negative under valid conditions).
38. 103	67	Loader return to Supervisor if zero, to address in switch itself if non-zero (after restoring DUP).
39. 104	68	Core map requested if non-zero, no map if zero.
40. 105	69	WS Indicator Word (disk block count of program in Working Storage).
		Parameters for Disk IOCS
41. 106	6A	Disk Arm Position
107-108	6B-6C	Reserved
42. 109	6D	Disk File Protect Address
110-111	6E-6F	Reserved
43. 112-114	70-72	Table of Defective Cylinders
115-120	73-78	Reserved
		Miscellaneous
44. 121	79	Disk block address of program. First word printed out at end of DUP function.
45. 122-125	7A-7D	Reserved
46. 126	7E	Size of core (1000_{16}, 2000_{16})
47. 127	7F	Absolute execution address of core load (for word 6 of LET/FLET entry).
48. 128-137	80-89	Reserved
49. 138	8A	Contents of index register 3 (points to middle of a 255-word transfer vector).
50. 139-144	8B-90	System Work Area

NOTE 1: When requested following a //JOB T control record, DUP will store information to disk and update LET on a temporary basis (only the adjusted value is altered). When the next //JOB or //JOB T control record is encountered, the adjusted value will be replaced by the base value. Thus, all information which has been stored in the User area since the first //JOB T record will be deleted. The //JOB T function requires that both base and adjusted values be available in COMMA. The base and adjusted values will be equal except during //JOB T operation.

NOTE 2: The interrupt levels associated with the I/O devices will be specified in COMMA. Since four bits are sufficient to specify any ILS number, three I/O levels may be included in each of the two words in COMMA which identify the principal I/O device and the principal print device (items 31 and 32, respectively). The rightmost four bits in each of these words identify the devices themselves. The layouts of these two words are as follows:

	0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15
96_{10}	Interrupt level of principal print device	Interrupt level of secondary print device	Reserved for future use	Principal print device indicator (0=1132, 1=console printer/ keyboard)
97_{10}	Interrupt level of principal I/O device	Interrupt level of column interrupt for 1442 if principal I/O device is 1134/1055; interrupt level of end-of-card interrupt for 1442 otherwise.	Interrupt level of end-of-card interrupt for 1442 if principal I/O device is 1134/ 1055; interrupt level of 1134/1055 otherwise.	Principle I/O device indicator (0=1134/1055, 1=1442, 2=console printer/keyboard) Supervisor only

Example

Word 96: 1132 + Console Printer 1400_{16}
Console Printer 4001_{16}

Word 97: 1442 0401_{16}
1442 + 1134/1055 0441_{16}
1134/1055 4000_{16}

NOTE 3: Set to something other than zero or one by any part of the system that finds a non-XEQ type error, reset to one by Supervisor after printing out a message, reset to zero by Supervisor upon sensing a //JOB record.

APPENDIX G. LAYOUT OF LET/FLET ENTRIES

THREE-WORD ENTRIES

Words	Description
1-2	Name of the program, consisting of five 6-bit characters, right-justified in the 32 bits of words 1 and 2. Names of less than five characters are padded with terminal blanks. A 6-bit character is formed by truncating the leftmost two bits of the EBCDIC representation of that character. Bits 0 and 1 are zeros.
3	Disk block count of the program.

SIX-WORD ENTRIES

Words	Description
1-2	Same as for three-word entries, except that for core-image programs, bit 0 is zero and bit 1 is one; and for data files, both bits 0 and 1 are ones.
3	Disk block count of the program, including padding. Padding is the number of disk blocks between the end of the last program or file stored and the beginning of this program, which is a sector boundary.
4	Execution address of the program, i.e., the core location to which control is passed for execution of the program (zero for data files).
5	Loading address of the program, i.e., the core location to which control is passed for the execution of the program (zero for data files).
6	Word count of the program, i.e., the number of words to be read from the disk when reading the information from disk to core storage.

NOTE 1: Eight sectors each are allocated for LET and FLET.

NOTE 2: The order of the entries in LET is the order in which the named items are stored in the User area.

APPENDIX H. IBM00 (1130 DISK MONITOR SYSTEM MAINTENANCE PROGRAM)

IBM00, the 1130 Disk Monitor System maintenance program, is the means by which a user updates his disk as modifications are released. The program automatically updates the monitor programs (Supervisor, Disk Utility, FORTRAN, and Assembler), provides a method of changing the IBM subroutine library, and also updates the version and modification level in the first word in DCOM. The leftmost 4 bits represent the version, and the rightmost 12 bits represent the modification level.

A card deck or paper tape containing corrections to maintain the monitor will be supplied by IBM. This includes all necessary control records. Every modification must be run to update the version and modification level even though the program affected is not on the disk.

IBM00 is stored on the disk as part of the IBM subroutine library. It is called from disk by the following control record:

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
// XEQ  I B M  0  0                    0
```

A zero must appear in position 19 of the XEQ record to specify DISK0.

Input to the program can be stacked with other jobs. However, when stacking modifications to the Monitor System, each patch that increases the modification level must begin with the above control record (see Figure 14).

Input to the program can be cards or paper tape. IBM00 determines the input device automatically by interrogating COMMA for the principal I/O device.

SYSTEM PROGRAM MAINTENANCE

Typical input for a system program update is as follows:

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
// XEQ  I B M  0  0                    0
Patch header record
Patch data record
   .
   .                    } One to eight data records
   .
Patch data record
Patch header record
Patch data record
   .
   .                    } One to eight data records
   .
Patch data record
Patch header record
   .
   .
   .
```

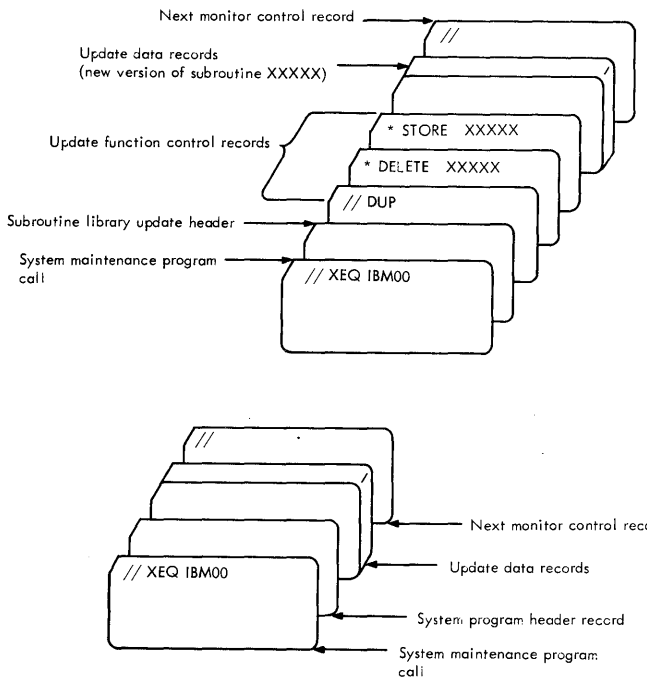


Figure 14. Control Records and Data Organization (in Card Form) for Monitor Program and Subroutine Library Maintenance

Patch Header Record

Each sector to be changed requires a patch header record. Thus, if a patch crosses a sector, two header records are required. If FORTRAN or the assembler is being modified, a check is made to determine if that system program has been voided from the disk. If so, the modification is not made.

The format of a patch header record (in terms of card input) is as follows:

Columns	Contents
1-3	Program ID (FOR, ASM, DUP, SUP)
10-11	Monitor System Version (01-15). The patch is not made if the version number does not agree with the version number in DCOM.
15-17	Modification Level (000-999). This must be the same in every header within a patch deck. The patch is made only if the modification level number is equal to or one greater than the modification level in DCOM. Changes to the modification level must be in ascending order, increasing by one level at a time. The user may rerun the modifications to his system by starting with modification level 001. If this is lower than the modification level in DCOM, a message is typed, followed by a wait. This notifies the user that he is processing his modifications from the beginning; upon continuing, the patch is made and the modification level is changed to 001. The modification level in DCOM is updated after the last record of the entire change is processed.
21-23	Sector Address (absolute, decimal, sector to be modified, except for the assembler, in which case it is relative to the sector address of the first sector of the assembler).
27-29	Relative word number of first patch word (000-319).
33-35	Word count of patch (001-320).
40-41	Total records in modification (02-99). This should appear only on the first patch header record. The count is the total record count of the modification, including data records and patch header records.
78-80	Sequence number (always 001).

Data Record

A data record (in terms of card input) is a binary data card (see Appendix B, Card System Format). One to eight data records can follow each patch header record, depending on the size of the patch. These must be numbered from 002 to 009, and must contain the proper checksum for this sequence.

The data record format is as follows:

Words	Contents
1	Location
2	Checksum
3	Type code (first 8 bits): 00001010; Word count (last 8 bits)
4-9	Relocation indicators
10-54	Data words 1 through 45
55-60	ID and sequence number

IBM SUBROUTINE LIBRARY MAINTENANCE

Changes to the subroutine library require reloading the new subroutine. IBM00 updates the version and modification level word; the actual reload is performed by a DUP DELETE function, followed by a DUP STORE function.

Typical input for a subroutine update is as follows:

```
cc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
// XEQ IBM 0 0                    0
(Subroutine header record)
// DUP
*DELETE                                          name
*STORE              C D      U A                 name
                      or
                    P T
(New version of the subroutine)
```

Subroutine Header Record

The subroutine header record must go through IBM00 even if the subroutine being modified is not on the user's disk. This is necessary to update the version and modification level word in DCOM so that the next sequential modification level can be made.

The format of a subroutine header record (in terms of card input) is as follows:

Columns	Contents
1-3	SUB (to signify a subroutine update)
10-11	Monitor System Version (01-15)
15-17	Modification Level (000-999)

OPERATING PROCEDURES

The card deck or paper tape supplied by IBM is to be run as a monitor job. When the control record //XEQ IBM00 is read, the version and modification level of the monitor is typed (see Figure 15, lines 1, 2, and 3). When the correct input is read, lines 1 through 6 of Figure 15 are typed.

ERROR MESSAGES

IBM00 error messages are listed in Table H-1.

Table H-1. IBM00 Monitor System Maintenance Error Messages

Code and Message	Meaning
U01 INVALID HEADER	Program cannot recognize header record.
U02 CHECKSUM ERROR	Data record checksum error. A record might be out of sequence or there might be an invalid data record.
U03 MCR BEFORE EOJ	Monitor control record was encountered before the updating process was completed.
U04 VERSION ERROR	The version on disk does not agree with the version in header record.
U05 MOD. LEVEL ERROR	The modification level in the header record is not equal to or one greater than the modification level on disk.

NOTE 1: All of the above errors require a retry of the execution after corrective action has been taken. Following an error type out, the program waits; pressing PROGRAM START causes an exit to the Supervisor.

NOTE 2: A user can start at modification No. 1 and add all modifications to date. A message is typed when this condition is encountered, followed by a wait. PROGRAM START must be pushed to continue. The following message will be typed: 'THE MONITOR SYSTEM IS BEING UPDATED WITH MOD. LEVEL NO. 1. PUSH PROGRAM START TO CONTINUE.'

```
(Line 1)   IBM00 MONITOR SYSTEM MAINTENANCE
(Line 2)     VERSION NO. IS   XX
(Line 3)     PRESENT MODIFICATION LEVEL IS  XXX

(Line 4)   MONITOR SYSTEM UPDATE COMPLETED
(Line 5)     VERSION NO. IS  XX
(Line 6)     NEW MODIFICATION LEVEL IS  XXX
```

Figure 15. Typeouts for 1130 Monitor System Maintenance Program

APPENDIX I. UTILITY ROUTINES

In addition to the IBM subroutine library, the following three utility programs, each complete with subroutines and loaders, are supplied to the user to enable him to perform operations external to the 1130 Disk Monitor System:

- Disk Pack Initialization Routine (DPIR). This routine is described under System Generation Operating Procedures - Card System.
- Console Printer Core Dump (described below).
- 1132 Printer Core Dump (described below).

CONSOLE PRINTER CORE DUMP

This routine aids the user in the debugging of programs. The programmer can dump portions of core by loading a single-card console routine which occupies the first 80 words of core. The output device is the console printer.

Format

This routine dumps core in hexadecimal form, starting with the word specified in the Console Entry switches. Dumping continues until PROGRAM STOP is pressed.

Words are dumped in four-digit hexadecimal form, with a space between each word. The first word typed is the starting address of the dump. The number of characters per line depends upon the margin settings of the console printer.

Operating Procedures

1. Set the Console Entry switches to the address at which dumping is to start.
2. Place the program card in the reader and set the Mode switch to RUN.
3. Press IMM STOP and RESET on the console.
4. Press START on the 1442.
5. Press PROGRAM LOAD on the console.

Dumping proceeds until IMM STOP is pressed. Press PROGRAM START to resume the dump.

1132 PRINTER CORE DUMP

This is a self-loading, four-card routine that dumps the contents of core storage in hexadecimal format on the 1132 Printer (the fourth card is blank). The routine is also available in paper tape.

Dumping begins at hexadecimal address 00A0 and continues to the end of core. Sixteen words per line are printed, preceded by a four-digit hexadecimal address of the first word of each line.

Card Operating Procedure

1. Ready the 1132 Printer.
2. Place the dump routine deck in the 1442 Card Read Punch hopper.
3. Press IMM STOP and RESET on the console.
4. Press START on the 1442.
5. Press PROGRAM LOAD on the console.

Dumping continues until the last 16 words of core are addressed and printed.

The program does not skip to the top of a new page to start, nor is page numbering or page overflow provided.

Paper Tape Operating Procedure

1. Ready the 1132 Printer.
2. Place the dump from 00A0 tape in the paper tape reader so that one of the delete codes beyond the program ID in the leader is beneath the starwheels.
3. Press IMM STOP, RESET, and PROGRAM LOAD on the console.

The output format is the same as described for the card routine.

```
// JOB                                                                    DKSAM001
// FOR                                                                    DKSAM002



                                                                          PAGE 01
**    IBM 1130 DISK MONITOR FORTRAN SAMPLE PROGRAM                        DKSAM003
*ONE WORD INTEGERS                                                        DKSAM004
*LIST ALL                                                                 DKSAM005
*IOCS (CARD, 1132 PRINTER, DISK)                                          DKSAM006
*NAME SAMPL                                                              DKSAM007



    IBM 1130 DISK MONITOR FORTRAN SAMPLE PROGRAM                          PAGE 02
C      IBM 1130 DISK MONITOR FORTRAN SAMPLE PROGRAM                       DKSAM008
C      SIMULTANEOUS EQUATION ROUTINE                                      DKSAM009
       DEFINE FILE 1 (10,320,U,INXT)                                      DKSAM010
       DIMENSION A(10,10),X(10),B(10),Y(10)                               DKSAM011
  301  FORMAT (1H1,20X15HINCOMPATIBILITY)                                 DKSAM012
  302  FORMAT (1H 20X41HMORE EQUATIONS THAN UNKNOWNS-NO SOLUTIONS)        DKSAM013
  303  FORMAT (1H 20X46HMORE UNKNOWNS THAN EQUATIONS-SEVERAL SOLUTIONS)   DKSAM014
  304  FORMAT (1H 20X15HSOLUTION MATRIX)                                  DKSAM015
  305  FORMAT(1H 20X8HMATRIX A)                                           DKSAM016
  306  FORMAT(1H 20X8HMATRIX B)                                           DKSAM017
 307   FORMAT (1H 20X10H A-INVERSE)                                       DKSAM018
  308  FORMAT(1H 20X24HDIAGONAL ELEMENT IS ZERO)                          DKSAM019
  309  FORMAT (1H 20X'A-INVERSE TIMES A')                                 DKSAM020
         M=2                                                             DKSAM021
        L=3                                                              DKSAM022
        READ  (M,10)                                                     DKSAM023
   10 FORMAT(72H           SPACE FOR TITLE                               DKSAM024
     1                )                                                 DKSAM025
        WRITE (L,10)                                                     DKSAM026
   12 FORMAT (6I10)                                                      DKSAM027
        READ  (M,12)M1,M2,L1,L2,N1,N2                                    DKSAM028
C      M1 = NO. OF ROWS OF A                                             DKSAM029
C      M2 = NO. OF COLS OF A                                             DKSAM030
C      L1 = NO. OF ROWS OF X                                             DKSAM031
C      L2 = NO. OF COLS OF X                                             DKSAM032
C      N1 = NO. OF ROWS OF B                                             DKSAM033
C      N2 = NO. OF COLS OF B                                             DKSAM034
   13 FORMAT (7F10.4)                                                    DKSAM035
   17 FORMAT (10F10.4)                                                   DKSAM036
      IF (N2-1)63,64,63                                                  DKSAM037
   64 IF (L2-1)63,65,63                                                  DKSAM038
   65 IF (L1-M2)63,66,63                                                 DKSAM039
   66 IF (M1-N1)63,11,63                                                 DKSAM040
   63  WRITE (L,301)                                                     DKSAM041
      GO TO 2                                                            DKSAM042
   11 N=M1                                                               DKSAM043
      N=M2                                                               DKSAM044
      IF (M1-M2) 91,14,93                                                DKSAM045
   91  WRITE (L,302)                                                     DKSAM046
      GO TO 2                                                            DKSAM047
   93  WRITE (L,303)                                                     DKSAM048
      GO TO 2                                                            DKSAM049
   14  WRITE (L,305)                                                     DKSAM050
      DO 70 I=1,N                                                        DKSAM051
       READ  (M,13)(A(I,J),J=1,N)                                        DKSAM052
       WRITE (L,17)(A(I,J),J=1,N)                                        DKSAM053
   70 CONTINUE                                                           DKSAM054
   89 FORMAT (F10.4)                                                     DKSAM055
       WRITE (L,306)                                                     DKSAM056
       READ  (M,89)(B(I),I=1,N)                                          DKSAM057
       WRITE (L,89)(B(I),I=1,N)                                          DKSAM058
C      PRESERVE THE ORIGINAL MATRIX ON DISK                              DKSAM059
       DO 19 I=1,N                                                       DKSAM060
   19 WRITE (1'I) (A(J,I), J=1,N)                                        DKSAM061
C      INVERSION OF A                                                    DKSAM062
   20 DO 120 K=1,N                                                       DKSAM063
      D=A(K,K)                                                           DKSAM064
      IF(D)40,200,40                                                     DKSAM065
   40 A(K,K)=1.0                                                         DKSAM066
```

```
    IBM 1130 DISK MONITOR FORTRAN SAMPLE PROGRAM                    PAGE 03
   50 DO 60 J=1,N                                                   DKSAM067
   60 A(K,J)=A(K,J)/D                                               DKSAM068
      IF(K-N)80,130,130                                             DKSAM069
   80 IK=K+1                                                        DKSAM070
      DO 120 I=IK,N                                                 DKSAM071
      D=A(I,K)                                                      DKSAM072
      A(I,K)=0.0                                                    DKSAM073
      DO 120 J=1,N                                                  DKSAM074
  120 A(I,J)=A(I,J)-(D*A(K,J))                                      DKSAM075
C     BACK SOLUTION                                                 DKSAM076
  130 IK=N-1                                                        DKSAM077
      DO 180 K=1,IK                                                 DKSAM078
  140 I1=K+1                                                        DKSAM079
      DO 180 I=I1,N                                                 DKSAM080
      D=A(K,I)                                                      DKSAM081
      A(K,I)=0.0                                                    DKSAM082
  170 DO 180 J=1,N                                                  DKSAM083
  180 A(K,J)=A(K,J)-(D*A(I,J))                                      DKSAM084
      GO TO 202                                                     DKSAM085
  200  WRITE (L,308)                                                DKSAM086
      GO TO 2                                                       DKSAM087
C     PRINT INVERSE                                                 DKSAM088
  202  WRITE (L,307)                                                DKSAM089
      DO 201 I=1,N                                                  DKSAM090
       WRITE (L,17)(A(I,J),J=1,N)                                   DKSAM091
  201 CONTINUE                                                      DKSAM092
      WRITE (L,309)                                                 DKSAM093
C     COMPUTE AND PRINT A-INVERSE TIMES A                           DKSAM094
      DO 123 J=1,N                                                  DKSAM095
C     RETRIEVE ORIGINAL BY COLUMNS                                  DKSAM096
      READ (1'J) (X(M), M=1,N)                                      DKSAM097
      DO 122 I=1,N                                                  DKSAM098
      Y(I) = 0.0                                                    DKSAM099
      DO 122 K = 1,N                                                DKSAM100
  122 Y(I) = Y(I)+A(I,K)*X(K)                                       DKSAM101
  123 WRITE (L,17) (Y(I), I=1,N)                                    DKSAM102
      DO 21 I=1,N                                                   DKSAM103
      X(I)=0.0                                                      DKSAM104
      DO 21 K=1,N                                                   DKSAM105
   21 X(I)=X(I)+A(I,K)*B(K)                                         DKSAM106
       WRITE (L,304)                                                DKSAM107
       WRITE (L,89)(X(I),I=1,N)                                     DKSAM108
    2 CALL EXIT                                                     DKSAM109
      END                                                           DKSAM110
```

```
    IBM 1130 DISK MONITOR FORTRAN SAMPLE PROGRAM                    PAGE 04

VARIABLE ALLOCATIONS
 A     =00CE  X     =00E2  B     =00F6  Y     =010A  D     =010C  INXT =010E  M     =010F  L     =0110  M1    =0111  M2    =0112
 L1    =0113  L2    =0114  N1    =0115  N2    =0116  N     =0117  I     =0118  J     =0119  K     =011A  IK    =011B  I1    =011C

UNREFERENCED STATEMENTS
 20       50       140      170

STATEMENT ALLOCATIONS
 301   =0127  302   =0134  303   =014E  304   =016A  305   =0177  306   =0180  307   =0189  308   =0193  309   =01A4  10    =01B2
 12    =01D8  13    =01DB  17    =01DE  89    =01E1  64    =021C  65    =0222  66    =0228  63    =022E  11    =0234  91    =0244
 93    =024A  14    =0250  70    =0289  19    =02C6  20    =02E7  40    =02FB  50    =0306  60    =030A  80    =0325  120   =0344
 130   =0374  140   =037E  170   =0399  180   =039D  200   =03CF  202   =03D5  201   =03F6  122   =042D  123   =045C  21    =048C
 2     =04D6

FEATURES SUPPORTED
 ONE WORD INTEGERS
 IOCS

CALLED SUBPROGRAMS
 FADDX   FMPYX   FDIV    FLD     FLDX    FSTO    FSTOX   FSBRX   SRED    SWRT    SCOMP   SFIO    SIOFX   SIOI    SUBSC
 CARDZ   PRNTZ   SDFIO   SDRED   SDWRT   SDCOM   SDFX

REAL CONSTANTS
  .100000E 01=0120     .000000E 00=0122

INTEGER CONSTANTS
     2=0124       3=0125       1=0126

CORE REQUIREMENTS FOR SAMPL
 COMMON       0  VARIABLES    288  PROGRAM    952

END OF COMPILATION
```

```
// XEQ       L                                                    DKSAM111
FILES ALLOCATION

    1 016C  000A

STORAGE ALLOCATION

R 47  0F71 (HEX) WORDS AVAILABLE

          LIBF TRANSFER-VECTOR

          EBCTB   1015
          HOLTB   0FDF
          GETAD   0F9E
          NORM    0F74
          XMDS    0F58
          FARC    0F36
          HOLEZ   0F00
          IFIX    0ED8
          FLOAT   0ECE
          FADDX   0E79
          SDRED   06C0
          FSBRX   0E50
          FMPYX   0E1C
          FDIV    0DCA
          FSTOX   0D72
          FLDX    0D8E
          SDCOM   06E4
          SDFX    06B2
          SDWRT   070E
          SIOFX   0915
          SUBSC   0DA8
          SIOI    0919
          SCOMP   0901
          SWRT    08F8
          SRED    0928
          FSTO    0D76
          FLD     0D92
          PRNTZ   0CC0
          CARDZ   0C7B
          SFIO    09CD
          SDFIO   0713
          DISKZ   00F4

          SYSTEM ROUTINES

          ILS02   1019

     03A5 (HEX) IS THE EXECUTION ADDR.



    IBM 1130 DISK MONITOR FORTRAN SAMPLE PROGRAM
                    MATRIX A
  4.2150   -1.2120    1.1050
 -2.1200    3.5050   -1.6320
  1.1220   -1.3130    3.9860
                    MATRIX B
  3.2160
  1.2470
  2.3456

                     A-INVERSE
  0.2915    0.0833   -0.0467
  0.1631    0.3836    0.1118
 -0.0283    0.1029    0.3008
                    A-INVERSE TIMES A
  0.9999   -0.0000    0.0000
  0.0000    0.9999   -0.0000
 -0.0000    0.0000    1.0000
                    SOLUTION MATRIX
  0.9321
  1.2654
  0.7429
```

```
// JOB                                                                  SMASM001
// ASM                                                                  SMASM002
*LIST                                                                   SMASM003
*PRINT SYMBOL TABLE                                                     SMASM004



                    COMPUTE THE SQUARE ROOT OF 64                       PAGE   1
0000 0  C030        BEGIN LD      D64                                   SMASM006
0001 20 064D6063          LIBF    FLOAT     INTEGER TO FLOATING PT.     SMASM007
0002 30 06898640          CALL    FSQR      FLOATING PT. SQRT.          SMASM008
0004 20 091899C0          LIBF    IFIX      FLOATING PT. TO INTEGER     SMASM009
0005 0  1008              SLA     8                                     SMASM010
                    *               MASK TO BUILD EBCDIC INTEGER        SMASM011
                    *               RESULT AND EBCDIC BLANK IN WORD1.   SMASM012
0006 0  E829              OR      MASK                                  SMASM013
0007 0  D01B              STO     WORD1                                 SMASM014
                    *               CONVERT MESSAGE FROM EBCDIC         SMASM015
                    *               TO ROTATE/TILT CODE.                SMASM016
0008 20 05097663          LIBF    EBPRT                                 SMASM017
0009 0  0000              DC      0                                     SMASM018
000A 1  0023              DC      WORD1                                 SMASM019
000B 1  0015              DC      TYPE&1                                SMASM020
000C 0  001A              DC      26                                    SMASM021
000D 20 23A17170          LIBF    TYPEO     TYPE MESSAGE                SMASM022
000E 0  2000              DC      /2000                                 SMASM023
000F 1  0014              DC      TYPE                                  SMASM024
0010 20 23A17170          LIBF    TYPEO     WAIT FOR TYPING COMPLETE    SMASM025
0011 0  0000              DC                                            SMASM026
0012 0  70FD              MDX     *-3                                   SMASM027
0013 0  6038              EXIT              RETURN TO MONITOR CONTROL   SMASM028
0014 0  000E        TYPE  DC      14                                    SMASM029
0015    000D              BSS     13                                    SMASM030
0022 0  8181              DC      /8181                                 SMASM031
0023 0  0000        WORD1 DC      *-*                                   SMASM032
0024    0018              EBC     .IS THE SQUARE ROOT OF 64.            SMASM033
0030 0  F040        MASK  DC      /F040                                 SMASM034
0031 0  0040        D64   DC      64                                    SMASM035
0032    0000              END     BEGIN                                 SMASM036



                          SYMBOL TABLE

   BEGIN 0000    D64   0031    MASK  0030    TYPE  0014    WORD1 0023

   NO ERRORS IN ABOVE ASSEMBLY.
```

```
// XEQ       L                                                SMASM037
R 47  1907 (HEX) WORDS AVAILABLE

          CALL TRANSFER VECTOR

          FSQR     020C

          LIBF TRANSFER-VECTOR

          FARC     066C
          XMDS     0650
          HOLL     0600
          PRTY     05B0
          EBPA     0560
          FADD     04AF
          FDIV     050E
          FLD      045A
          FADDX    04B5
          FMPYX    0470
          FSTO     043E
          FGETP    0424
          NORM     03FA
          TYPE0    02D2
          EBPRT    026C
          IFIX     0244
          FLOAT    01F4
          DISKZ    00F4

          SYSTEM ROUTINES

          ILS04    0691
          ILS02    06AD

      01C2 (HEX) IS THE EXECUTION ADDR.
```

Program Output on Console Printer

```
8 IS THE SQUARE ROOT OF 64
```

Adding and removing subroutines 39
ARITHMETIC TRACE, FORTRAN control record 33
ASM, monitor control record 9
Assembler 26
  control records 26
  error detection codes 29
  messages and error codes 29
  operating procedures (card) 29
  operating procedures (paper tape) 30
  origin of source program 28
  paper tape format 28
Assembler control records 26
  COMMON 28
  FILE 28
  LEVEL 28
  LIST 27
  LIST DECK 27
  LIST DECK E 27
  PRINT SYMBOL TABLE 27
  PUNCH SYMBOL TABLE 28
  SAVE SYMBOL TABLE 28
  SYSTEM SYMBOL TABLE 28
  TWO PASS MODE 26
Assembler error messages (Table A-7) 52

Card Data format (CDD) 61
Card subroutine errors (CARD0 and CARD1) 36
Card System format (CDS) 59
Character codes, Supervisor and DUP I/O (Appendix D) 64
Cold start
  halt addresses (Table 13) 44
  operating procedure, card 44
  operating procedure, paper tape 46
Cold start operating procedures (cards) 44
Cold start operating procedures (paper tape) 46
COMMA (In-Core Communications Area) 7
  core locations (Appendix F) 70
Comments, Monitor control record 10
COMMON, assembler control record 28
Compilation error messages 35
Compilation messages 34
Console Printer Core Dump 78
Console printer subroutine errors (TYPE0 and WRTY0) 38
Control records
  Assembler 26
  DUP 17
  FORTRAN 32
  Monitor 7
  Supervisor 10
Conversion factors, disk storage unit 63
Core image buffer (CIB) 4
Core Image format
  format (Appendix B) 59
  loading 16

Data cards (Card System format) 60
Data formats (Appendix B) 57
  Card Data (CDD) 61
  Card System (CDS) 59
  Disk Core Image (DCI) 59
  Disk Data (DDF) 59
  Disk System (DSF) 57
  Paper Tape Data (PTD) 61
  Paper Tape System (PTS) 61
  Print Data (PRD) 61
DCOM (Disk Communications Area) 7
DEFINE, DUP control record 23
DELETE, DUP control record 23
Disk Communications Area (DCOM) 7
Disk Core Image format (DCI) 59
Disk Data format (DDF) 59
Disk Pack Initialization Routine (DPIR) 40
Disk storage allocation (Table 1) 3
Disk storage layout 3
Disk storage unit conversion factors 63
Disk system format (DSF)
  format (Appendix B) 57
  loading 16
Disk Utility Program (DUP) 17
  control records 17
  error messages (Table A-9) 55
  messages 24
  operating notes 25
DPIR card load operating procedures 40
DPIR paper tape load operating procedures 45
DUMP, DUP control record 18
DUMPDATA, DUP control record 18
DUMPLET, DUP control record 22
DUP, monitor control record 10
DUP control records 17
  DEFINE 23
  DELETE 23
  DUMP 18
  DUMPDATA 18
  DUMPLET 22
  DWADR 22
  EDIT 24
  STORE 20
  STORECI 20
  STOREDATA 21
  STOREMOD 20
DUP error messages (Table A-9) 55
DUP waits and loops (Table A-10) 56
DWADR, DUP control record 22

EOP card (Card System format) 60
Error messages (Appendix A) 47
  Assembler (Table A-7) 52
  DUP (Table A-9) 55
  DUP Waits and Loops (Table A-10) 56
  FORTRAN (Table A-8) 53
  FORTRAN I/O (Table A-11) 56
  Loader (Table A-6) 50
  Monitor Supervisor (Table A-4) 48
  Monitor Supervisor Wait Locations (Table A-5) 49
  System Loader (Table A-1) 47

System Loader Wait Locations, Part 1 (Table A-2) 48
System Loader Wait Locations, Part 2 (Table A-3) 48
EXTENDED PRECISION, FORTRAN control record 33

FILE, assembler control record 28
File protection 5
FILES, supervisor control record 11
Fixed area 5
Fixed Location Equivalence Table (FLET) 5
FLET (Fixed Location Equivalence Table) 5
layout of LET/FLET entries (Appendix G) 74
output format (Figure 8) 23
Flipper routine 5
FOR, monitor control record 9
Formats (Appendix B) 57
FORTRAN compiler 32
compilation error messages 35
compilation messages 34
control records 32
I/O logical unit designations (Table 8) 32
printouts 34
FORTRAN control records 32
ARITHMETIC TRACE 33
EXTENDED PRECISION 33
IOCS 32
LIST ALL 33
LIST SOURCE PROGRAM 32
LIST SUBPROGRAM NAMES 32
LIST SYMBOL TABLE 32
NAME 33
ONE WORD INTEGERS 33
TRANSFER TRACE 34
FORTRAN error codes (Table A-8) 53
FORTRAN I/O error codes (Table A-11) 56

IBM systems area 3
IBM00 (1130 Disk Monitor System Maintenance Program) 75
ILS header card (Card System format) 61
In-Core Communications Area (COMMA) 7
core locations (Appendix F) 70
Initializing Disk Monitor System from cards 43
Initializing Disk Monitor System from paper tape 45
IOCS, FORTRAN control record 32
ISS header card (Card System format) 61

JOB, monitor control record 9

Keyboard subroutine functions (TYPE0) 38

LET (Location Equivalence Table) 4
layout of LET/FLET entries (Appendix G) 74
output format (Figure 7) 22
LEVEL, assembler control record 28
LIST, assembler control record 27
LIST ALL, FORTRAN control record 33
LIST DECK, assembler control record 27
LIST DECK E, assembler control record 27
LIST SOURCE PROGRAM, FORTRAN control record 32
LIST SUBPROGRAM NAMES, FORTRAN control record 32
LIST SYMBOL TABLE, FORTRAN control record 32
Load Mode Control Card 41
Loader 12
Loader messages/error messages (Table A-6) 50
LOCAL, supervisor control record 10
Location Equivalence Table (LET) 4

Machine requirements ii
Mainline header card (Card System format) 59
Monitor control record analyzer 7
Monitor control records 7
ASM 9
Comments 10
DUP 10
FOR 9
JOB 9
PAUS 9
TEND 10
TYP 9
XEQ 10
Monitor supervisor error messages (Table A-4) 48
Monitor supervisor wait locations (Table A-5) 49

NAME, FORTRAN control record 33
NOCAL, supervisor control record 11

Object-time transfer vector 15
ONE WORD INTEGERS, FORTRAN control record 33
Operating procedures
cold start (cards) 44
cold start (paper tape) 46
DPIR card load 40
DPIR paper tape load 45
initializing Disk Monitor System from cards 43
initializing Disk Monitor System from paper tape 45
system generation (card system) 40
Optional tracing (FORTRAN) 34
Origins for core loads 13

Paper Tape Data format (PTD) 61
Paper tape monitor system 45
Paper tape subroutines (PAPT) 39
Paper Tape System format (PTS) 61
Patch header record (IBM00) 76
PAUS, monitor control record 9
Pre-operative errors (subroutine library) 36
Print Data format (PRD) 61
PRINT SYMBOL TABLE, assembler control record 27
Program header record (Disk System format) 57
Program subtypes 58
Program types 58
PUNCH SYMBOL TABLE, assembler control record 28

REQ Cards 42

SAVE SYMBOL TABLE, assembler control record 28
SCON Card 42
Skeleton Supervisor 7
SOCALs (system overlays) 15
Stacked input arrangement 12
STORE, DUP control record 20
STORECI, DUP control record 20
STOREDATA, DUP control record 21

STOREMOD, DUP control record 20
Subroutine header card (Card System format) 60
Subroutine header record (IBM00) 77
Subroutine library 5, 36
  listing (Appendix E) 65
Supervisor and DUP I/O character codes (Appendix D) 64
Supervisor control record area 4
Supervisor control records 10
  FILES 11
  LOCAL 10
  NOCAL 11
Supervisor program 3, 7
System generation (card system) 40
System loader error codes (Table A-1) 47
System loader wait locations, Part 1 (Table A-2) 48
System loader wait locations, Part 2 (Table A-3) 48
System overlays (SOCALs) 15
SYSTEM SYMBOL TABLE, assembler control record 28

TEND, monitor control record 10
TERM Card 42
TRANSFER TRACE, FORTRAN control record 34
TWO PASS MODE, assembler control record 26
TYP, monitor control record 9

User area 4
User storage area 4
User-supplied cards 41
Utility routines (Appendix I) 78
  Console Printer Core Dump 78
  Disk Pack Initialization Routine (DPIR) 40
  1132 Printer Core Dump 78

Working Storage area 5
Working storage indicator word 17

XEQ, monitor control record 10

1130 Disk Monitor System Maintenance Program (IBM00) 75
1132 Printer Core Dump 78

C26-3750-0

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM 1130 DISK MONITOR SYSTEM
REFERENCE MANUAL

Form C26-3750-0

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

	Yes	No
• Does this publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

- What is your occupation? _____
- How do you use this publication?

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Other _____

- Please give specific page and line references with your comments when appropriate. If you wish a reply, be sure to include your name and address.

COMMENTS

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114

Attention: Programming Publications, Dept. 452

fold fold

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]