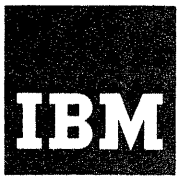
File 1130-36
Form C26-3709-0



Systems Reference Library

IBM 1130 Disk Monitor System, Version 2 System Introduction

This publication describes the 1130 Disk Monitor System, Version 2. It also provides the functional specifications for programming the IBM 1130 Computing System using this monitor system.

The 1130 Disk Monitor System, Version 2, is a combined programming and operating system that provides for continuous operation of the 1130 in a stacked job environment. This monitor system supports the expanded hardware features and the high-speed input/output devices available on the 1130.

First Edition

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

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

Address comments concerning the contents of the publication to IBM Corporation, Programming Publications Dept. 232, San Jose, California, 95114

The IBM 1130 Disk Monitor System, Version 2, supports a wide range of I/O devices and machine configurations, including:

IBM 1131 Central Processing Unit, Model 2, with 4096, 8192, 16,384, or 32,768 words of core storage

IBM 1131 Central Processing Unit, Model 3, with 8192, 16,384, or 32,768 words of core storage

IBM 2310 Disk Storage, Model B

IBM 1442 Card Punch, Model 5 - may not be used with 1442-6 or 1442-7

IBM 1442 Card Read Punch, Models 6 and 7 - may not be used with 1442-5

IBM 2501 Card Reader, Models A1 and A2 - may not be used with 1231

IBM 1231 Optical Mark Page Reader - may not be used with 2501

IBM 1134 Paper Tape Reader and IBM 1055 Paper Tape Punch

IBM 1132 Printer

IBM 1403 Printer, Models 6 and 7

IBM 1627 Plotter

Synchronous Communications Adapter

The minimum machine configuration required by the 1130 Disk Monitor System, Version 2, is:

IBM 1131 Central Processing Unit, Model 2, with 4096 words of core storage

and one of the following input/output devices:

IBM 1134 Paper Tape Reader in combination with the IBM 1055 Paper Tape Punch

IBM 1442 Card Read Punch, Model 6

IBM 2501 Card Reader in combination with the IBM 1442 Card Punch, Model 5

The 1130 Disk Monitor System, Version 2, provides for the continuous operation of the 1130 Computing System, with minimal set-up time and operator intervention, in a stacked job environment. The monitor system consists of seven distinct but interdependent elements - Supervisor, Disk Utility Program, Assembler, FORTRAN Compiler, Core Load Builder, Core Image Loader, and System Library.

The Supervisor performs control functions for the monitor system and provides the linkage between user programs and monitor programs.

The Disk Utility Program is a group of IBM-supplied programs which perform certain operations involving the disk such as storing, moving, deleting, and dumping data and/or programs.

The Assembler converts source programs written in Assembler language into machine language object programs.

The FORTRAN Compiler translates source programs written in 1130 Basic FORTRAN IV language into machine language object programs.

The Core Load Builder constructs core image programs from mainline object programs. The mainline programs are converted into disk core image format from disk system format and the resultant program is readied for immediate execution or for storing for future execution.

The Core Image Loader serves as both a loader for core loads and as an interface for some parts of the monitor system.

The System Library is a group of disk-resident programs which perform I/O, data conversion, arithmetic, disk initialization, and maintenance functions.

For an understanding of the 1130 Disk Monitor System, Version 2, the reader should be familiar with the following publications:

IBM 1130 Functional Characteristics (Form A26-5881-3)

IBM 1130 Computing System Input/Output Units (Form A26-5890-3)

IBM 1130 Assembler Language (Form C26-5927-2)

IBM 1130 Basic FORTRAN IV Language (Form C26-5933-3) with Technical Newsletters N26-0510 and N26-0527

IBM 1130 Subroutine Library (Form C26-5929-2) with Technical Newsletter N26-0557

In addition to the publications listed above the reader should familiarize himself with the terms in the Glossary contained in this manual. It is important that these terms be understood as they are defined herein, in the context of the 1130 Disk Monitor System, Version 2.

CONTENTS

ILLUSTRATIONS

Figures

Tables

STACKED JOB ENVIRONMENT

The 1130 Disk Monitor System, Version 2, enables the user to assemble, compile, and/or execute programs written in the 1130 Assembler Language or 1130 Basic FORTRAN IV Language. This system also provides for the continuous operation of the 1130 Computing System, with minimal set-up time and minimal operator intervention. This is accomplished through stacked job input.

INPUT STREAM

The input stream (that is, the stacked job input) to the monitor system consists of control records, source programs, object programs, and data, as shown in Figure 1.

The control record formats and input sequences are shown in this manual in card form for purposes of illustration. There is, however, no difference between the control record formats and input sequences of card input and those of paper tape input.

JOB AND SUBJOB

In the input stream to the monitor system, a job is defined as:

- The processing that takes place from the detection of a JOB monitor control record until the detection of another JOB monitor control record.
- A JOB monitor control record and all the following control records, source programs, object programs, and data, up to, but not including, the next JOB monitor control record.

A subjob is defined as:

- The processing that takes place from the detection of a monitor control record until the detection of another monitor control record.
- A monitor control record and all the following control records, source programs, object programs, and data, up to, but not including, the next monitor control record.

A job is an independent unit of processing; a subjob is a unit of processing which is dependent on the subjobs preceding it and upon which the following subjobs are dependent. The successful completion of the job depends on the successful completion of each subjob within it. In some cases, a subjob is not attempted if the preceding subjobs have not been successfully completed.

SYSTEM OPERATION

The Supervisor is initially brought into control by means of the Cold Start procedure. The Supervisor then begins analyzing monitor control records from the input stream. If a monitor control record indicates a Supervisor operation only (JOB, PAUS, TYP, TEND, or Comments), the appropriate operation is performed and the Supervisor reads and processes the next monitor control record from the input stream.

When a monitor program is requested on a monitor control record (DUP, FOR, or ASM), the appropriate program is fetched by the Supervisor and control is transferred to it. Control is returned to the Supervisor by the monitor program at the normal completion of its operation(s) or when it detects a monitor control record in the input stream.

If a core load execution is requested by a monitor control record (XEQ), the Supervisor fetches the Core Image Loader and transfers control to it. If the program to be executed is in disk core image format (that is, it is a core image program), it is fetched and control is passed to it. If the program is in disk system format, (that is, it is a DSF program), the Core Image Loader calls the Core Load Builder to construct a core image program. When the building of the core image program is complete, it is loaded into core by the Core Image Loader and control is passed to it.

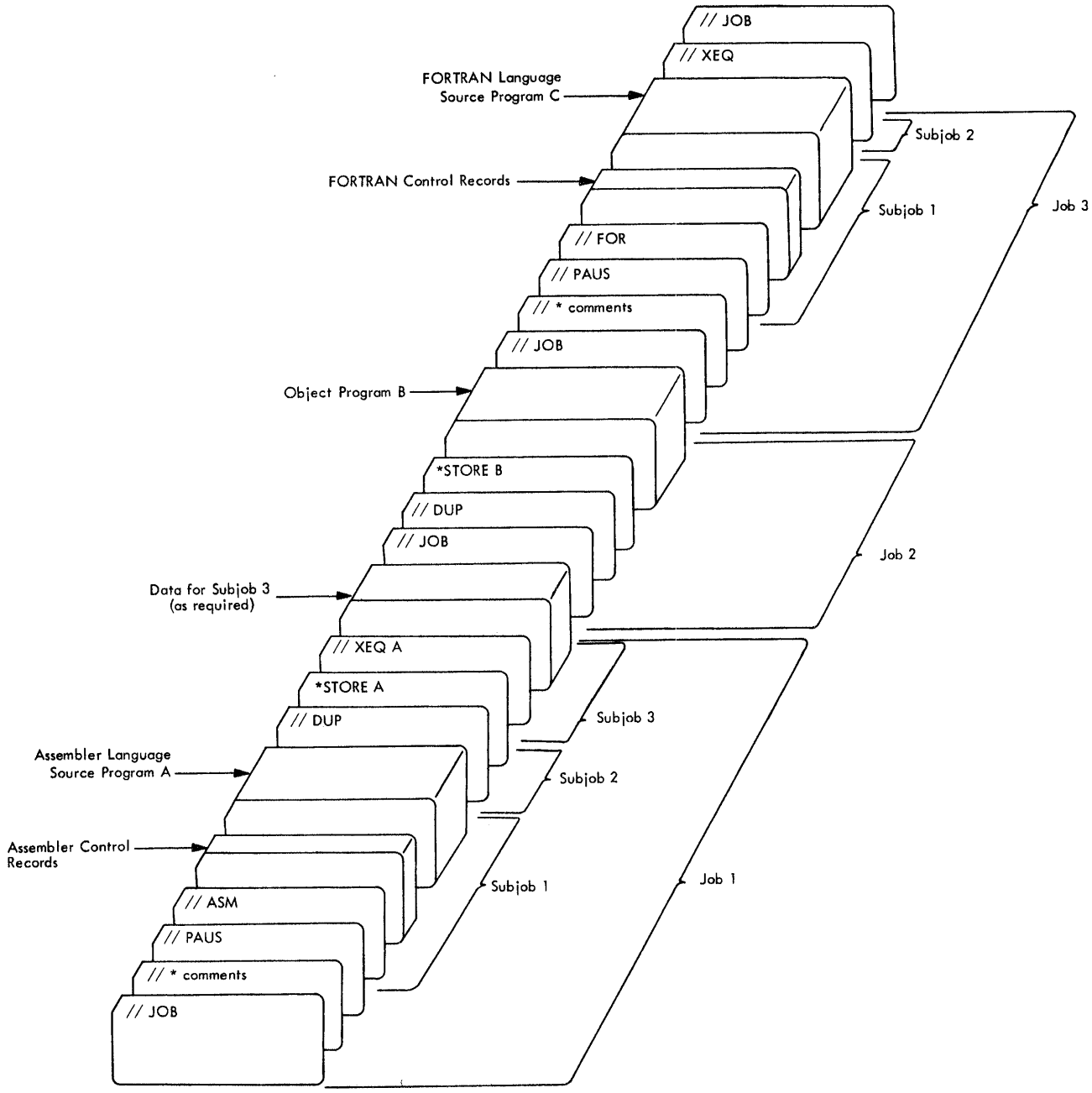


Figure 1. Stacked Job Input

If a core load terminates with a CALL LINK, the Skeleton Supervisor is entered at the LINK entry point. The Skeleton Supervisor calls the Core Image Loader to (1) save any COMMON defined below location 4096_{10} by the previous core load, (2) look up the next link (the DSF program or core image program specified in the CALL LINK as the next program to be executed) in LET/FLET, (3) build a core image program, if necessary, via the Core Load Builder, and (4) fetch the core load and pass control to it.

If a core load terminates with a CALL EXIT, the Skeleton Supervisor is entered at the EXIT entry point. The Skeleton Supervisor calls the Core Image Loader, which, in turn, calls the Supervisor into core to read and process the next monitor control record from the input stream.

If a dynamic dump of the contents of core is desired during the execution of a user program, the Skeleton Supervisor is entered at the DUMP entry point. The Skeleton Supervisor saves core (below location 4096_{10}) on the Core Image Buffer (CIB) and calls the Core Image Loader, which, in turn, calls the Supervisor DUMP program. The contents of core are printed on the principal printer, core (below location 4096_{10}) is restored from the CIB, and the DUMP program returns control to the user program at the core location following the call to the DUMP program.

If a terminal dump of the contents of core is desired following the execution of a user program, the Supervisor is entered at the DUMP entry point plus 1. The contents of core are dumped as described above and the DUMP program terminates with a CALL EXIT.

Figure 2 shows the overall flow of the 1130 Disk Monitor System, Version 2.

DISK-RESIDENT SYSTEM

The 1130 Disk Monitor System, Version 2, is a disk-resident system; this means that:

- The complete monitor system resides on disk
- Only a minimal amount of core storage is taken up by the core-resident program (the Resident Monitor)
- Only the program required at any one time is fetched for execution

The monitor system is initially loaded to a disk cartridge, called a system cartridge, by means of the System Loader provided by IBM. Placement of a system cartridge on any physical drive readies the system for the user-initiated Cold Start procedure. The Cold Start establishes the physical drive on which a system cartridge has been placed as logical drive 0, which is, by definition, the system drive. In addition, the system cartridge on logical drive 0 becomes the master cartridge; all other cartridges, system or non-system, are satellite cartridges.

Figure 3 shows the layout of a system cartridge. Figure 4 shows the layout of a non-system cartridge, a cartridge that contains no monitor programs. Such a cartridge on multi-drive 1130 systems can be used exclusively for the storage of data and/or programs. Note that no scale is intended in these figures.

CYLINDER 0 ON A SYSTEM CARTRIDGE

The cartridge identification or ID (a hexadecimal number in the range 0001-7FFF that uniquely identifies the cartridge) and the addresses of any defective cylinders (up to 3) on the cartridge reside on the first sector (see Figure 5). The remainder of the first sector is unused.

The Disk Communications Area resides on the second sector (see the description of DCOM, below).

The third sector of cylinder 0 contains the Resident Image, that is, the disk image of the Resident Monitor without the disk I/O subroutine (see "Cold Start Procedure" and "Resident Monitor" under Supervisor, below).

The System Location Equivalence Table (SLET) resides on the fourth and fifth sectors. SLET is composed of an identification number, core loading address, word count, and sector address for every phase of every monitor program.

The sixth sector is occupied by the Cold Start program (see the description of the Cold Start program, below).

The seventh sector contains the Reload Table, which is used by the System Loader program when reloading a cartridge and by DUP when deleting the Assembler or the FORTRAN Compiler.

The last sector of cylinder 0 is unused.

CYLINDER 0 ON A NON-SYSTEM CARTRIDGE

The first sector of cylinder 0 on a non-system cartridge contains the same kind of information as cylinder 0 on a system cartridge. The second sector contains only that information from DCOM applicable to this non-system cartridge. (See the description of DCOM, below.)

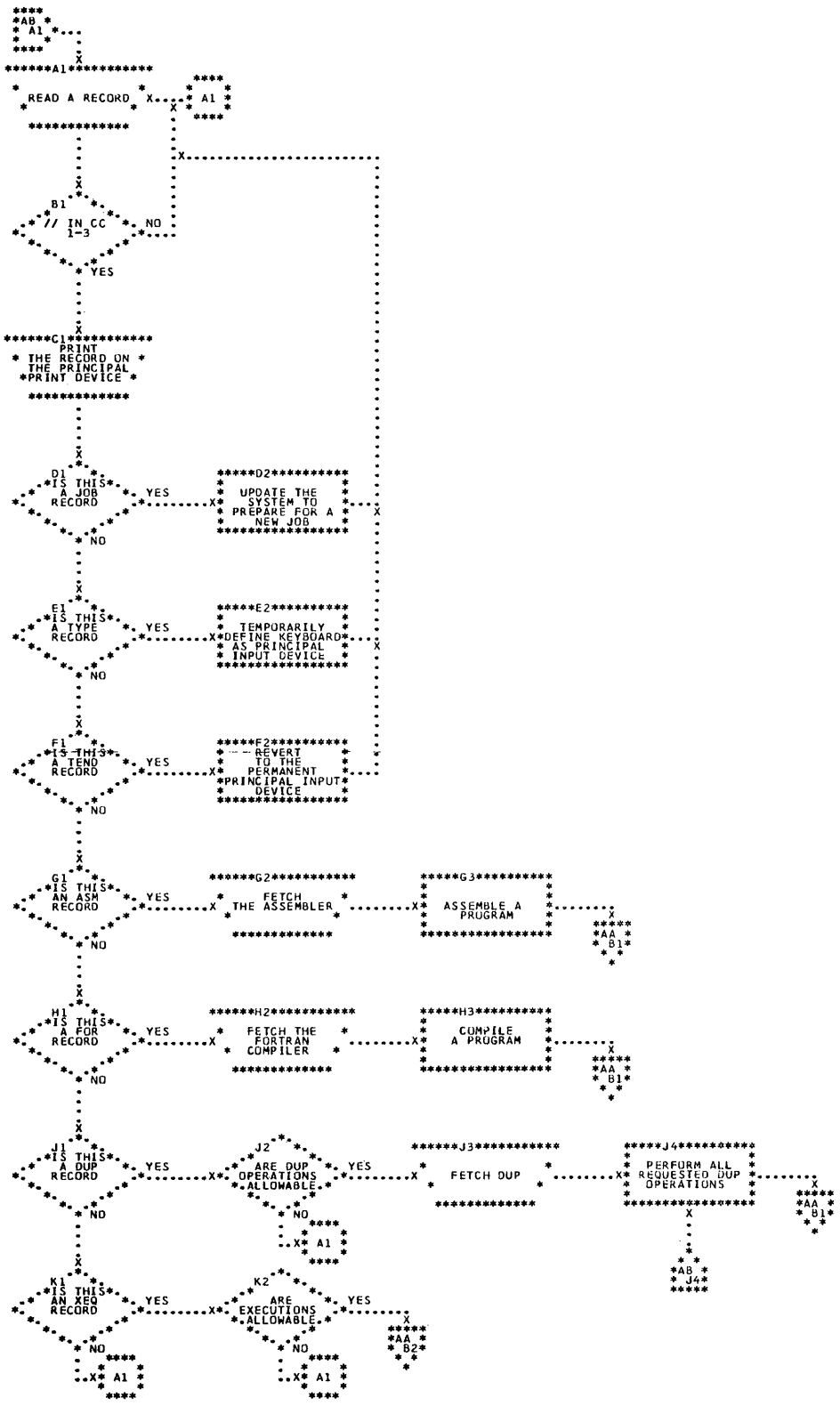


Figure 2. Overall Flow of the System (Part 1 of 2)

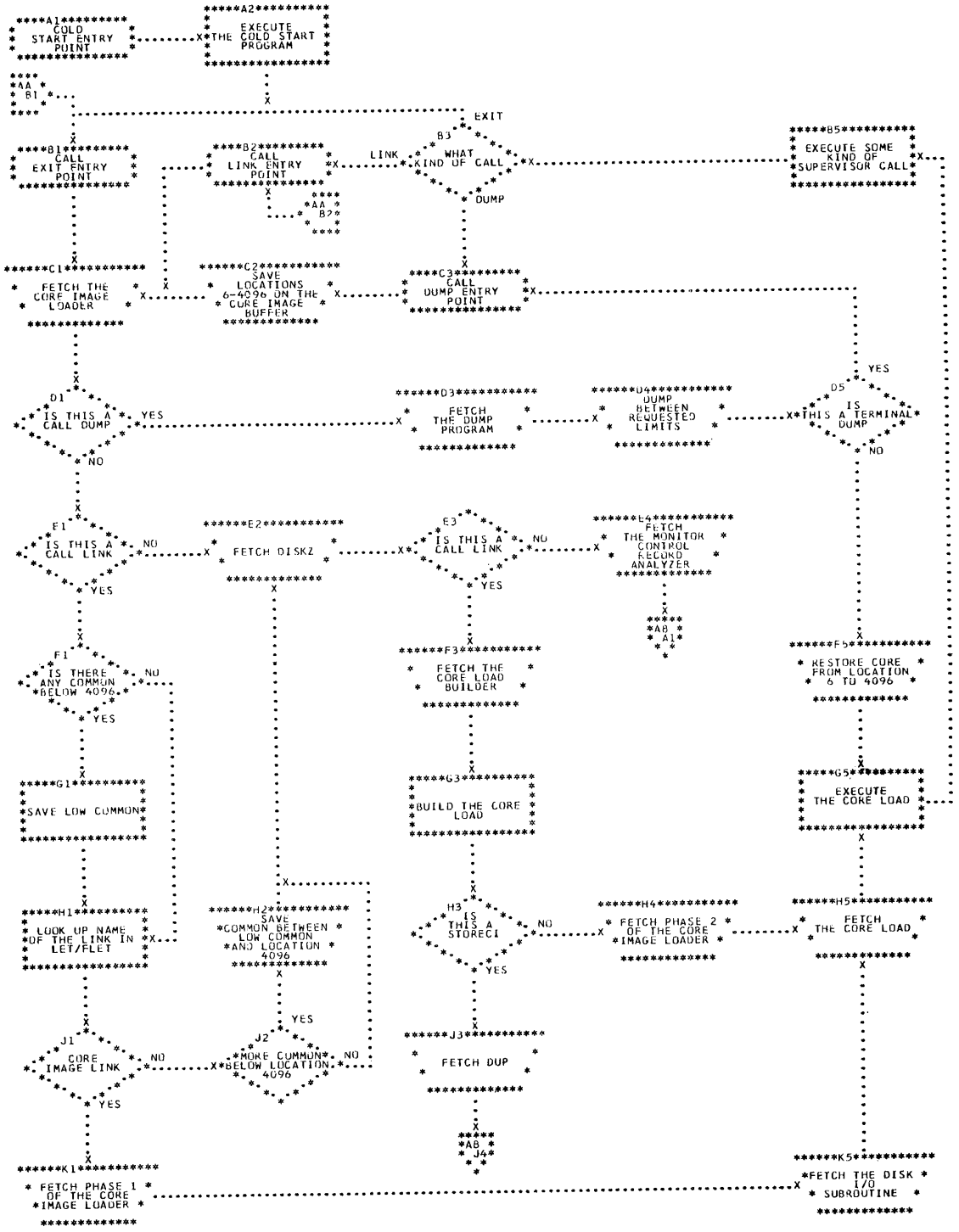
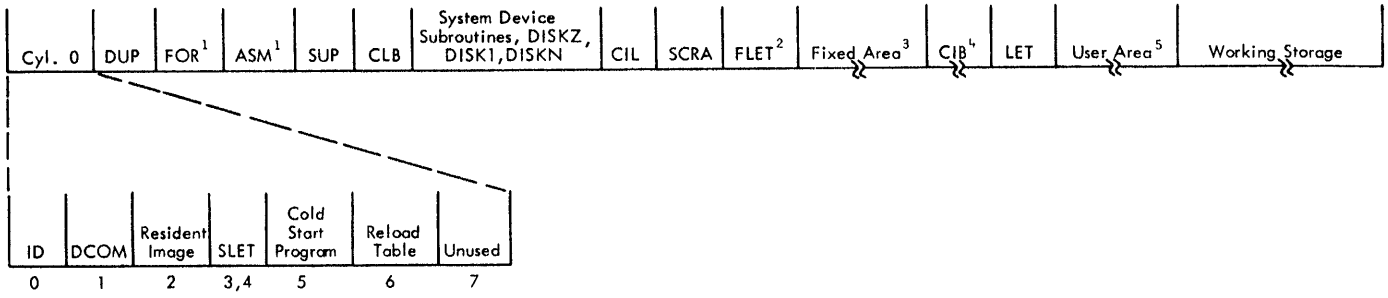


Figure 2. Overall Flow of the System (Part 2 of 2)



1. Can be deleted from the system by the user
2. Present only if a Fixed Area is defined for this cartridge by the user
3. Optionally defined by the user
4. May not be deleted by the user from a system cartridge.
5. Initially contains only the System Library; user-written programs may be added

Figure 3. Layout of a System Cartridge

The Location Equivalence Table (LET) for the cartridge (see the description of the Location Equivalence Table, below) occupies the remaining six sectors of cylinder 0.

SYSTEM AREA

The System Area is that area occupied by the elements described below. This area is found only on a system cartridge.

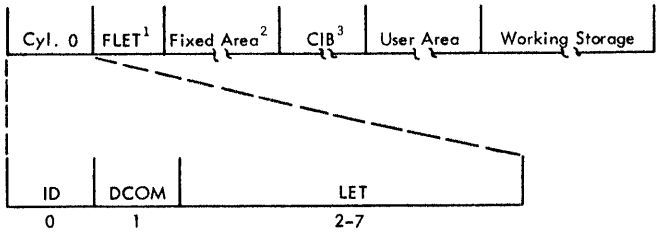
Disk Communication Area (DCOM)

DCOM contains the parameters that must be passed from one monitor program to another and that must be accessed through disk storage (as opposed to core storage). Generally speaking, parameters that are not required when fetching a link stored in disk core image format are found in DCOM.

The first of the two parts of DCOM contains the parameters that are not related to all the disk cartridges, for example, the core map switch. The second contains the cartridge-related parameters. Each of the parameters in this section is in the form of a five-word table, one word for the corresponding value for each of the five possible cartridges. The five words of each table are arranged in the order of logical drive numbers; that is, the first is for logical drive 0, the second for logical drive 1, etc.

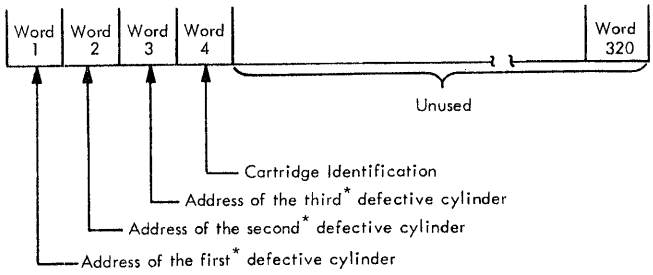
Resident Image

The Resident Image is a copy of the Resident Monitor without a disk I/O subroutine; that is, it is a reflection of COMMA and the Skeleton Supervisor (see "Resident Monitor" in the section Supervisor). It is used to initialize the Resident Monitor during the Cold Start procedure.



1. Present only if a Fixed Area is defined for this cartridge by the user
2. Optionally defined by the user
3. May be deleted by the user. However, a CIB must be present on at least one of the cartridges on the system at any given time.

Figure 4. Layout of a Non-System Cartridge



*In the order found defective by DCIP

Figure 5. Layout of Sector 0, Cylinder 0, on any Cartridge

Cold Start Program

The Cold Start Program initializes the 1130 Disk Monitor System, Version 2. It is read into core as a result of the Cold Start procedure (see "Cold Start Procedure" under Supervisor).

DUP

The Disk Utility Program (DUP) is actually a group of programs provided by IBM to perform certain frequently-required operations involving the disk, such as storing, moving, deleting, and dumping data and/or programs. These operations are called, for the most part, by user-supplied DUP control records.

FORTRAN Compiler

The FORTRAN Compiler translates source programs written in 1130 Basic FORTRAN IV language into machine language object programs. The compiler also provides for the calling of the necessary arithmetic, function, conversion, and input/output subprograms at execution time.

The compiler is initially loaded onto the system cartridge with the rest of the system; however, it can subsequently be deleted from the system at the user's option (see "DEFINE" under Disk Utility Program, below).

Assembler

The Assembler converts source programs written in Assembler language into machine language object programs.

The Assembler is initially loaded onto the system cartridge with the rest of the system; however, it can subsequently be deleted from the system at the user's option (see "DEFINE" under Disk Utility Program, below).

Supervisor

The Supervisor provides the linkage between user programs and monitor programs. The Supervisor is directed by the monitor control records in the stacked job input.

Core Load Builder

The Core Load Builder builds a specified mainline program into a core image program. The mainline program, with its required programs (LOCALs and SOCALs included), is converted from disk system format to disk core image format. During the conversion, the Core Load Builder also builds the core image header record and the transfer vector. The resultant core image program is suitable for immediate execution or for storing on the disk in disk core image format for future execution.

System Device Subroutine Area

The System Device Subroutine Area contains the following:

1. The subroutines used by the monitor programs to operate the following print devices:
   1132 Printer
   1403 Printer
   Console Printer
2. The subroutines used by the monitor programs to operate the following I/O devices:
   1134/1055 Paper Tape Reader
   Punch
   1442 Card Read Punch, Model 6 or 7
   2501 Card Reader/1442 Card Punch, Model 5, 6, or 7
   Keyboard/Console Printer
3. The I/O character code conversion subroutines used in conjunction with the I/O subroutine for the following devices:
   1134/1055 Paper Tape Reader
   Punch
   2501 Card Reader/1442 Card Read Punch
   Keyboard/Console Printer
4. The disk I/O subroutines:
   DISKZ
   DISK1
   DISKN
   These subroutines are found in this area rather than in the System Library because they are processed by the Core Load Builder differently than the subroutines stored in the System Library.

All of the subroutines in the System Device Subroutine Area, except the disk I/O subroutines, are

naturally relocatable and are intended for system use only.

Core Image Loader

The Core Image Loader is the program that is called to handle the three entries to the Skeleton Supervisor – LINK, DUMP, and EXIT. The Core Image Loader is assigned this task in order to achieve the fastest possible link-to-link transfer of control (via CALL LINK).

On a LINK entry to the Skeleton Supervisor, the Core Image Loader handles the locating and fetching of the core load and the calling of the Core Load Builder, if necessary. On an EXIT or DUMP entry, the Core Image Loader calls the appropriate Supervisor program into operation.

Supervisor Control Record Area

The Supervisor Control Record Area (SCRA) is the area in which Supervisor control records (LOCAL, NOCAL, and FILES) are saved. They are read from the input stream (following an XEQ or STORECI control record) and are stored in the SCRA for subsequent processing by the Core Load Builder.

CORE IMAGE BUFFER (CIB)

The CIB is the area on disk in which the Core Load Builder builds any portion of a core load which resides below location 4096_{10}. It is also used to save any COMMON defined below location 4096_{10} during the transfer of control from one link to the next.

SYSTEM LIBRARY

The System Library consists of (1) a complete library of input/output (except disk I/O) subroutines, data conversion subroutines, and arithmetic and function subprograms, (2) selective dump subroutines, and (3) special mainline programs for disk maintenance (see "Disk Maintenance Programs" under System Library).

The System Library is initially loaded into the User Area on a system cartridge. However, the user may, at his option, move the System Library from a system cartridge to the User Area on a non-system cartridge. Certain programs in the System Library, namely, the disk maintenance programs, are required for the operation of the system; these programs may not be deleted from the System Library. Other portions of the System Library may be deleted at the user's option.

FIXED AREA (FX)

The Fixed Area is the area in which the user may store core image programs in disk core image format and/or data files in disk data format if it is desired that these core image programs and data files always occupy the same sectors. The Fixed Area is optionally defined on any cartridge by the use of the DUP operation, DEFINE FIXED AREA. This operation is also used to increase or decrease the size of the Fixed Area.

When a core image program or data file is stored in the Fixed Area, it is stored starting at the nearest sector boundary. When a core image program or data file is deleted from the Fixed Area, no packing of the Fixed Area occurs. Hence, core image programs and data files in this area reside at fixed sector addresses and can be referenced as such by the user.

FIXED LOCATION EQUIVALENCE TABLE (FLET)

The Fixed Location Equivalence Table (FLET) is a directory to the contents of the Fixed Area on the cartridge on which it appears. There is one FLET entry for:

1. Each core image program stored in disk core image format
2. Each data file stored in disk data format
3. The padding required to permit a core image program or data file to be stored on a sector boundary

Each FLET entry specifies the name of the core image program or data file, its format, and its size in disk blocks.

Each cartridge on the system having a Fixed Area defined on it has a FLET. Regardless of the size of the Fixed Area, the FLET for a cartridge occupies one cylinder, which immediately precedes the Fixed Area.

The sector address of FLET on a given cartridge is obtained from the LET on the same cartridge.

USER AREA (UA)

The User Area is the area in which the user can store programs in disk system format, core image programs in disk core image format, and/or data files in disk data format. The User Area is defined on any cartridge when the cartridge is initialized. However, its size is 0 sectors until the first DSF program, core image program, or data file is stored in the User Area on that cartridge. The User Area occupies as many sectors as are required to contain the DSF programs, core image programs, and data files stored there.

When a DSF program, core image program, or data file is to be added to the User Area, it is stored at the start of Working Storage, that is, immediately following the end of the User Area. The area occupied by the new DSF program, core image program, or data file is then incorporated into the User Area, and Working Storage is decreased by the size of that area.

DSF programs are stored in the User Area starting at the beginning of a disk block; core image programs and data files are stored starting at the nearest sector boundary.

When a DSF program, core image program, or data file is deleted from the User Area, the User Area is packed; that is, the DSF programs, core image programs, and/or data files in the User Area are moved so as to occupy the vacancy (the area formerly occupied by the deleted DSF program, core image program, or data file). In packing, DSF programs are moved to the first disk block boundary in the vacancy; core image programs and data files are moved to the first sector boundary in the vacancy. All following DSF programs, core image programs, and data files are similarly packed.

LOCATION EQUIVALENCE TABLE (LET)

The Location Equivalence Table (LET) on a cartridge is a directory to the contents of the User Area on that cartridge. There is one LET entry for:

1. Each entry point for each program stored in disk system format
2. Each core image program stored in disk core image format
3. Each data file stored in disk data format
4. The padding required to permit a core image program or data file to be stored on a sector boundary

Each LET entry specifies the name of an entry point, core image program, or data file; the format of the DSF program, core image program, or data file; and its size in disk blocks.

Each cartridge on the system has a LET. However, a cartridge has a User Area only if there is an entry in the LET on that cartridge other than a dummy entry. On a system cartridge LET occupies the cylinder preceding the User Area.

COMMA contains the sector address of the first sector of LET for each cartridge being used in a given job.

WORKING STORAGE (WS)

Working Storage is that area on all cartridges that is not defined as the User/Fixed Area or, on the system cartridge, as the System Area. Working Storage is available to monitor and user programs alike as temporary disk storage. It extends from the sector boundary immediately following the User Area to the end of the cartridge (cylinder 199).

SUPERVISOR

The Supervisor performs the control functions for the monitor system. The Supervisor reads control records included in the stacked job input, decodes them, and fetches the appropriate monitor program to perform the specified operation.

COLD START PROCEDURE

The Supervisor initially achieves control over the 1130 Computing System through the user-initiated Cold Start procedure. The Cold Start procedure begins with the IPL (Initial Program Load) of the Cold Start record, which causes the Cold Start program to be read into core storage from the system cartridge and control to be transferred to it.

The Cold Start program, in turn, loads the Resident Monitor into its location in lower core storage. The Cold Start procedure ends when control is given to the job initialization program in the Supervisor.

RESIDENT MONITOR

The resident portion of the monitor system consists of (1) a data area used for system parameters and for communication between monitor programs (COMMA), (2) the Skeleton Supervisor, and (3) a disk I/O subroutine (either DISK1, DISKN, or DISKZ).

CORE COMMUNICATIONS AREA (COMMA)

COMMA can generally be defined as that information required by the Core Image Loader to perform a link-to-link transfer of control without referring to DCOM. This information is interspersed with parts of the Skeleton Supervisor.

SKELETON SUPERVISOR

On any entry to the Resident Monitor (EXIT, LINK, or DUMP), the Skeleton Supervisor calls the Core Image Loader, which determines where the Skeleton Supervisor was entered and either calls the Supervisor if the entry was at EXIT or DUMP or fetches and transfers control to the core load specified in the CALL LINK statement if the entry was at LINK.

This use of the Core Image Loader as an intermediate supervisor allows the monitor system to achieve efficient link-to-link transfer of control.

The Skeleton Supervisor occupies approximately 90 words in low core storage, interspersed with COMMA. The Skeleton Supervisor consists of the subroutines and entry points described below.

LINK Entry Point

LINK is the entry point in the Skeleton Supervisor used to accomplish a link-to-link transfer of control.

EXIT Entry Point

EXIT is the entry point in the Skeleton Supervisor used to accomplish a link-to-Supervisor transfer of control.

DUMP Entry Point

DUMP is the entry point in the Skeleton Supervisor used to obtain a printout of the contents of core storage between specified limits. Dynamic dumps are obtained through the DUMP entry point; terminal dumps are obtained through the DUMP entry point plus 1.

ILS02 Subroutine

The ILS02 subroutine handles the servicing of interrupts on level 2. All of the disk devices on the system, and only they, interrupt on level 2. Due to the necessary usage of the disk, the ILS02 subroutine is necessarily a part of the Resident Monitor.

ILS04 Subroutine

The ILS04 subroutine handles the servicing of interrupts on level 4. One of the devices which interrupt

on level 4 is the Keyboard. Since the user may, at any time, perform a console interrupt request, the ILS04 subroutine is necessarily a part of the Resident Monitor.

Preoperative Error Trap

The preoperative error trap is entered by all ISS subroutines when an error is found during the preoperative parameter checking. The trap consists only of a WAIT and a branch. When the PROGRAM START key is pressed, execution resumes following the branch to this trap.

PROGRAM STOP Key Trap

The PROGRAM STOP key trap is entered if a level 5 interrupt occurs and there is no user-written device subroutine associated with level 5. The trap consists only of a WAIT and a branch. When the PROGRAM START key is pressed, the interrupt level is turned off and execution resumes following the point of the level 5 interrupt.

This trap allows the user to stop the entire 1130 Computing System with the ability to continue execution without disturbing the system status or the contents of core storage.

DISK I/O SUBROUTINE

The disk I/O subroutine required by the program in control resides in core storage following the Skeleton Supervisor. The following table lists the disk I/O subroutines, their approximate sizes, and the corresponding addresses of the end of the Resident Monitor.

Subroutine	Size (in words)	End of Resident Monitor (Core Location)
DISKZ	225_{10}	450_{10}
DISK1	500_{10}	725_{10}
DISKN	700_{10}	925_{10}

DISKZ is the disk I/O subroutine used by all system programs, the subroutine initially loaded with the Resident Monitor.

Prior to execution of a core load requiring DISK1 or DISKN, the Core Image Loader overlays the required disk I/O subroutine on DISKZ. When control is returned to the Supervisor, the Core Image Loader restores DISKZ for use by the monitor programs, using the disk I/O subroutine currently in core storage (DISK1 or DISKN). User programs, including those written in the FORTRAN language, may use any of the three disk I/O subroutines. However, only one disk I/O subroutine may be referenced in a given core load.

SUPERVISOR PROGRAMS

The programs described below are the disk-resident programs which constitute the Supervisor. One of these programs is fetched and given control by the Core Image Loader, depending upon the entry made in the Skeleton Supervisor; the Monitor Control Record Analyzer is called following an EXIT entry, the DUMP program following a DUMP entry.

MONITOR CONTROL RECORD ANALYZER

The Monitor Control Record Analyzer (1) reads a monitor control record or Supervisor control record from the input stream, (2) prints the control record on the principal print device, and (3) fetches the required monitor program and transfers control to it.

Supervisor Control Record Area

The Supervisor Control Record Area is the area on disk, within the System Area, in which the Supervisor places the FILES, LOCAL, and NOCAL control records read from the input stream. The Core Load Builder reads these records from this area on disk for analysis during the building of the core image program.

DUMP PROGRAM

The DUMP program provides the user with a printout of the contents of core storage. See the description of the PDMP and DUMP statements in the Assembler Language section for details on the use of the DUMP program.

Terminal and Dynamic Dumps

The DUMP entry point in the Skeleton Supervisor (and, thus, the DUMP program in the Supervisor) can be entered (1) by a BSI to the DUMP entry point, (2) by a manually executed transfer to the DUMP entry point plus 1, or (3) by a branch to location zero, which contains an MDX to the DUMP entry point plus 1.

If the DUMP entry point is entered from any location but zero, a dump is given in hexadecimal format of the area of core storage bounded by the limit parameters. Execution of the core load in progress then resumes at the location following the call to the DUMP entry point.

If the DUMP entry point is entered by a branch through location zero or if the DUMP entry point plus 1 is entered by a branch or a manual transfer, a dump is given in hexadecimal format of the entire contents of core storage. The DUMP program then executes a CALL EXIT, thereby terminating the execution of the core load in progress.

MONITOR CONTROL RECORDS

The monitor control records are described below. Where shown in the control record format, the blank character (b) is required. Any unused columns following the end of the control record options are available for remarks.

JOB

The JOB control record defines the start of a new job. It causes the Supervisor to perform the job initialization procedure, which includes:

1. The initialization of COMMA
2. The initialization of the parameters in DCOM that are not related to all the disk cartridges.
3. The setting of the temporary mode indicator if a T is present in column 8 of the control record. If set, the temporary mode indicator causes all DSF programs, core image programs, or data files stored in the User Area by DUP during the current job to be deleted automatically from that area at the end of the job (that is, at the beginning of the next job).
4. The definition of the cartridges to be used during the current job. IDs 1 through 5 on the JOB control record specify the cartridges to be used. These cartridges may be mounted on the physical drives in any order. The order of the IDs in the JOB control record specifies the logical assignments for the cartridges. IDs 1 through 5 correspond to logical drives 0 through 4. The cartridge-related entries of COMMA and DCOM are filled in according to the logical order specified by the user.
5. The definition of the cartridge on which the CIB for the current job is to be found. The ID of the cartridge containing the CIB must follow the field of the fifth cartridge ID. If the CIB ID is omitted, the CIB on the master cartridge is used. Core image programs can be built faster if the CIB is assigned to a cartridge other than the master cartridge.
6. The definition of the cartridge containing the Working Storage to be used by the monitor programs. The ID of the cartridge to be used for Working Storage must follow the ID of the CIB cartridge. If the Working Storage ID is omitted, the monitor programs use the Working Storage on the master cartridge. Core image programs can be built faster, however, if the system Working Storage is on some cartridge other than the master cartridge. They can be built even faster if the CIB, the system Working Storage, and the monitor system itself are on separate cartridges. Assemblies are also faster if Working Storage is on a separate cartridge.
7. The starting of a new page. A skip to channel 1 is executed on the 1132 and 1403 Printers; ten consecutive carriage returns are made on the Console Printer. The page count is reset to 1, and the date information is replaced with whatever appears in columns 46-53 of the JOB control record.

The format of the JOB control record is described below.

Card Column	Contents	Notes
1-6	//ƀ JOB	
7	Reserved	
8	Temporary mode indicator	A T indicates that temporary mode is desired for this job.
9-10	Reserved	
11-14	First ID	This is the ID of the master cartridge (logical drive 0)
15	Reserved	
16-19	Second ID	This is the ID of the cartridge on logical drive 1.
20	Reserved	
21-24	Third ID	This is the ID of the cartridge on logical drive 2.
25	Reserved	
26-29	Fourth ID	This is the ID of the cartridge on logical drive 3.
30	Reserved	
31-34	Fifth ID	This is the ID of the cartridge on logical drive 4.
35	Reserved	
36-39	CIB ID	This is the ID of the cartridge containing the CIB to be used during this job.
40	Reserved	
41-44	Working Storage ID	This is the ID of the cartridge containing the Working Storage to be used by the system during this job.
45	Reserved	
46-53	Header Data, Date, Name, etc.	This information is printed at the top of every page of the listing on the principal print device during this job.
54-80	Not used	

ASM

This control record causes the Supervisor to read the Assembler into core storage and transfer control to it. Any Assembler control records and the source statements to be assembled must follow this control record. Comments control records may not follow this control record.

The format of the ASM control record is described below.

Card Column	Contents	Notes
1-6	//ƀ ASM	
7-80	Not used	

FOR

This control record causes the Supervisor to read the FORTRAN Compiler into core storage and transfer control to it. Any FORTRAN control records and the source statements to be compiled must follow this control record. Comments control records may not follow this control record.

The format of the FOR control record is described below.

Card Column	Contents	Notes
1-6	//ƀ FOR	
7-80	Not used	

DUP

This control record causes the Supervisor to read the control portion of the Disk Utility Program into core storage and transfer control to it. The DUP control record(s) must follow this control record. Only one DUP monitor control record is required to process a stack of DUP control records provided no monitor control record other than the Comments control record is encountered.

The format of the DUP control record is described below.

Card Column	Contents	Notes
1-6	//ƀ DUP	
7-80	Not used	

XEQ

This control record causes the Supervisor to initialize for core load execution. If the name specified in this control record (in columns 8 through 12) is that of a mainline program stored in disk system format, the Supervisor reads the Supervisor control records, if any, from the input stream and writes them in the Supervisor Control Record Area (SCRA). The Core Load Builder is then called to build a core image program from the mainline program.

If no name is specified on the control record, a mainline program in disk system format is assumed

to be stored in Working Storage. The Supervisor then processes the Supervisor control records and calls the Core Load Builder as outlined above.

After the core image program has been built, or if the name in the control record was that of a core image program already stored on disk in core image format, the Core Image Loader is called to read the core load into core storage and to transfer control to it.

If an L is punched in column 14 of the control record, a core map is printed by the Core Load Builder during the building of the core image program. In addition, a core map is printed for all links during the current execution that are stored in disk system format. These core maps include:

1. The execution address of the mainline program
2. The names and execution addresses of all subprograms in the core load
3. All file allocations, with the file number, cartridge ID, sector address, and size (in sectors)

Columns 16 and 17 of the control record contain the right-justified decimal count of Supervisor control records to be read by the Supervisor before calling the Core Load Builder.

Column 19 contains the character indicating the disk I/O subroutine to be used by the core load at execution time. If column 19 contains zero or one, DISK1 is fetched by the Core Image Loader along with the core load. If column 19 contains an N, DISKN is fetched. If column 19 contains any other character, including a blank, no disk I/O subroutine is fetched (that is, DISKZ, which is in core storage for use by the monitor programs, is used by the core load). The only restriction is that all links in disk system format that are called during a given execution utilize the same disk I/O subroutine.

The format of the XEQ control record is described below.

Card Column	Contents	Notes
1-6	//ƀ XEQ	
7	Reserved	
8-12	Name	This is the name of the DSF program or core image program to be executed.
13	Reserved	
14	Core Map indicator	An L indicates that a core map is to be printed for this and all following links in disk system format during this execution.
15	Reserved	
16-17	Count	This is the decimal number of Supervisor control records which follow.
18	Reserved	
19	Disk I/O subroutine indicator	This column specifies the disk I/O subroutine to be loaded into core by the Core Image Loader for use by the core load at execution time.
20-80	Not used	

PAUS

This control record causes the Supervisor to WAIT. When PROGRAM START is pressed, the Supervisor continues processing monitor control records from the input stream.

The format of the PAUS control record is described below.

Card Column	Contents	Notes
1-7	//ƀ PAUS	
8-80	Not used	

TYP

This control record causes the Supervisor to temporarily assign the Keyboard as the principal input device. The Keyboard instead of the card or paper tape reader is the principal input device until the detection of the next TEND control record.

The format of the TYP control record is described below.

Card Column	Contents	Notes
1-6	//ƀ TYP	
7-80	Not used	

TEND

This control record causes the Supervisor to reassign the card or paper tape reader as the principal input device. The reassignment is to whichever unit was the principal input device prior to the detection of the last TYP control record.

The format of the TEND control record is described below.

Card Column	Contents	Notes
1-7	//b TEND	
8-80	Not used	

COMMENTS

This control record allows the user to print alphameric text on the listing printed on the principal print device by the Supervisor and DUP. The Supervisor and DUP simply print the control record and continue reading control records from the input stream. The Comments control record may not immediately follow an XEQ, ASM, or FOR control record.

The format of the Comments control record is described below.

Card Column	Contents	Notes
1-4	//b *	
5-80	User comments	Any alphameric character(s) may be used.

SUPERVISOR CONTROL RECORDS

The control records described below (LOCAL, NOCAL, and FILES) are used by the Core Load Builder to:

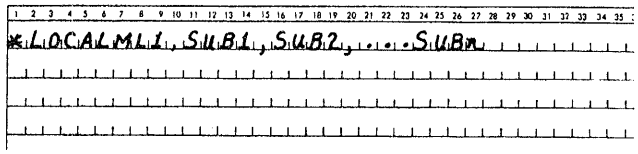
1. Provide for subprogram overlays at execution time (LOCAL)
2. Include subprograms not called in the core load (NOCAL)
3. Equate disk storage files defined in the mainline program during compilation or assembly to specific files stored on the disk (FILES)

These control records are placed in the input stream following a XEQ monitor control record that names a mainline program stored in disk system format or following an STORECI DUP control record. In both cases the control records are written on disk in the Supervisor Control Record Area (SCRA), from which the Core Load Builder reads them for processing.

Up to 99 of each of the types of Supervisor control records may follow the XEQ or STORECI control record. There is no specified order (by type) to be followed; however, the types may not be intermixed.

LOCAL

LOCAL (Load-On-Call) subprograms are subprograms specified by the user to be loaded at execution time into a LOCAL overlay area as they are called. The LOCAL subprograms are specified on the LOCAL control record as follows:

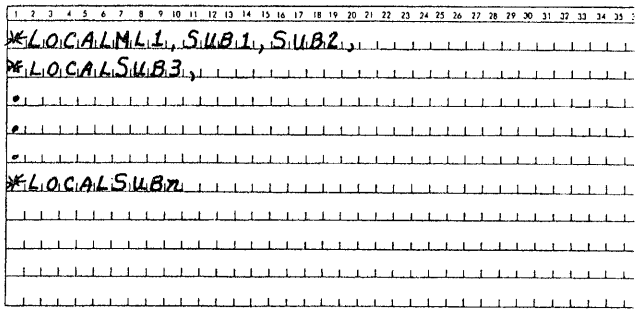


where

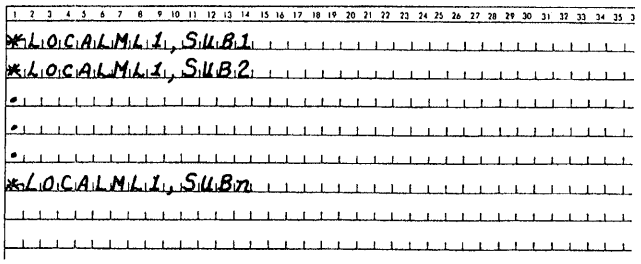
ML1 is the name of the mainline program,

SUB1 through SUBn are the names of the LOCAL subprograms for that mainline program.

In the case illustrated below, all the LOCAL control records except the last end with a comma, which indicates continuation, and the mainline program name appears on the first LOCAL control record only.

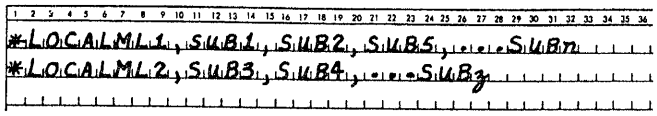


The same results would have been obtained if the records had been:



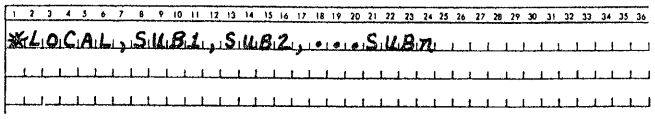
All of the LOCAL subprograms for each mainline program in an execution must be specified on the LOCAL control records which follow the XEQ monitor control record initiating the execution.

Separate LOCAL control records must be used for each mainline program that calls LOCAL subprograms in the execution, for example,



where ML2 is a link called by ML1.

If the mainline program is to be executed from Working Storage, the mainline program name must be omitted from the LOCAL control record, for example,



No embedded blanks are allowed in the LOCAL control record.

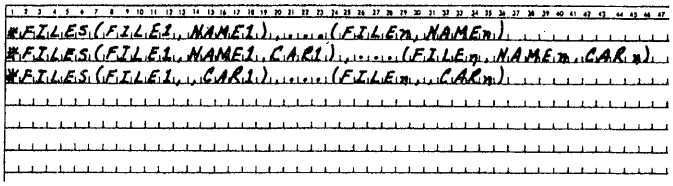
NOCAL

NOCAL (Load-Although-Not-Called) subprograms are subprograms specified by the user to be included in the core load, even though they are not called. They are specified on the NOCAL control record under the same rules that apply for LOCAL control records except that *NOCAL is used in place of *LOCAL.

FILES

By means of FILES control records the file numbers specified at compilation or assembly time in FORTRAN DEFINE FILE statements or in Assembler FILE statements are equated to the names of data files stored in the User and Fixed Areas. All the files to be used by all the core loads in an execution must be defined in the FILES control records following the XEQ monitor control record initiating the execution. All the files thus defined are available to each core load in the execution.

The format of the FILES control record is as follows:



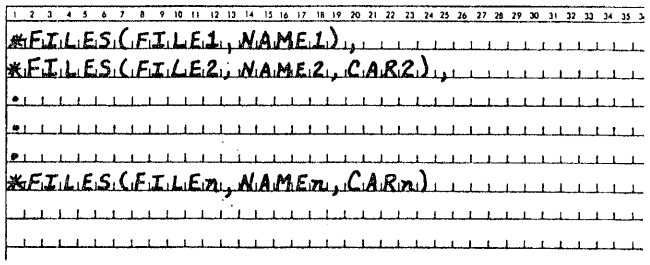
where

FILE1 through FILEn are the file numbers specified in the FORTRAN DEFINE FILE statements or Assembler FILE statements,

NAME1 through NAMEn are the names of data files already stored on disk. If the name is omitted, the file is placed in Working Storage on the specified cartridge.

CAR1 through CARn are the IDs of the cartridges on which the respective data files are to be found. If the cartridge ID is omitted, it is assumed that the corresponding data file has been defined on the master cartridge.

Continuation of FILES control record may be indicated by a comma following the last file definition on the control record, as follows:



The continuation comma may only appear immediately after a right parenthesis.

The information on all the FILES control records for an execution may not exceed 640 words,

counting the file numbers as one word each, the file names as two words each, and the cartridges IDs as one word each.

No embedded blanks are allowed in the FILES control record.

RULES FOR LOCAL AND NOCAL USAGE

The user must observe the following rules in the usage of LOCAL and NOCAL control records:

1. A subprogram cannot be specified as a LOCAL subprogram if it causes another subprogram, also specified as a LOCAL subprogram in the same mainline program, to be called. For example, if A calls B and B calls C, and A is a LOCAL subprogram, neither B nor C can be specified as a LOCAL subprogram for the same mainline program.
2. NOCAL subprograms may call other NOCAL subprograms.
3. If a subprogram is specified as a LOCAL subprogram and system overlays (SOCALs) are employed, the subprogram is made a LOCAL subprogram, even if it would otherwise have been included in one of the SOCALs.
4. If a subprogram is specified as a LOCAL subprogram, it is included as a LOCAL subprogram in the core image program even if it is not called in the core load.
5. The information on all the LOCAL control records for an execution may not exceed 640 words, counting the mainline program names as three words each and the subprogram names as two words each. This restriction applies to NOCAL control records also.
6. Only subprogram types 3, 4, 5, and 6 can be named on LOCAL and NOCAL control records. Subprogram types 3 and 5 are called with LIBF statements, types 4 and 6 with CALL statements. Types 5 and 6 are ISSs, types 3 and 4 are subprograms.

DISK UTILITY PROGRAM

The Disk Utility Program (DUP) provides the user with the ability to perform the following operations through the use of control records:

- Store DSF programs, core image programs, and data files on the disk
- Make the DSF programs, core image programs, and data files on the disk available in printed, punched card, or punched paper tape format
- Remove DSF programs, core image programs, and data files from the disk
- Determine the status of disk storage through a printed copy of LET/FLET, the directory to the disk
- Alter certain system parameters and, to a limited extent, the contents of the system
- Perform other disk maintenance functions

GENERAL FLOW

DUP is called into operation when the Supervisor recognizes a DUP monitor control record. The control portion of DUP is brought into core to read the next record from the input stream, which should be a DUP control record. The DUP control record is then printed and analyzed. LET is searched for the program specified and switches and indicators are set in accordance with the information obtained from the control record. The DUP program required to perform the requested operation is then read into core from the disk and given control.

The DUP program performs its assigned tasks directed by the switches and indicators that were set according to the information on the DUP control record. Upon completion of its tasks, the DUP program prints a message and returns control to the control portion of DUP. The control portion indicates the completion of the DUP operation with a printed message and reads the next record from the input stream.

If the record read is a monitor control record other than Comments, control is returned to the Supervisor to process the record. If the record read is a DUP control record, DUP maintains control and begins the performance of the indicated operation. Comments monitor control records are simply printed; blank records are passed.

INFORMATION TRANSFER AND FORMAT CONVERSION

Table 1 summarizes the DUP operations that transfer information from one area or medium to another area or medium. In addition, the format conversions made during the transfers of information are shown. The acronyms for the various formats are described below.

Acronym	Format
DSF	Disk System Format
DDF	Disk Data Format
DCI	Disk Core Image Format
CDS	Card System Format
CDD	Card Data Format
CDC	Card Core Image Format
PTS	Paper Tape System Format
PTD	Paper Tape Data Format
PTC	Paper Tape Core Image Format
PRD	Printer Data Format

LET/FLET

The two tables LET and FLET constitute a directory to the contents of the User and Fixed Areas. The allocation of disk storage and, correspondingly, the contents of LET/FLET can be altered by the user only through the use of DUP.

Before storing any DSF program, core image program, or data file, DUP searches LET/FLET to ensure that the name of the DSF program, core image program, or data file does not already appear in LET/FLET on the cartridge specified on the DUP control record. Disk storage is allocated

"FROM" Area Symbols, with Formats		"TO" Area Symbols, with Formats														
		UA			FX		WS			CD			PT			PR
		DSF	DDF	DCI	DDF	DCI	DSF	DDF	DCI	CDS	CDD	CDC	PTS	PTD	PTC	PRD
UA	DSF						DUMP	DUMPDATA		DUMP	DUMPDATA		DUMP	DUMPDATA		DUMP DUMPDATA
	DDF							DUMP DUMPDATA			DUMP DUMPDATA			DUMP DUMPDATA		DUMP DUMPDATA
	DCI							DUMPDATA			DUMPDATA	DUMP		DUMPDATA	DUMP	DUMP DUMPDATA
FX	DDF							DUMP DUMPDATA			DUMP DUMPDATA			DUMP DUMPDATA		DUMP DUMPDATA
	DCI							DUMPDATA			DUMPDATA	DUMP		DUMPDATA	DUMP	DUMP DUMPDATA
WS	DSF	STORE STOREMOD	STOREDATA	STORECI	STOREDATA	STORECI				DUMP	DUMPDATA		DUMP	DUMPDATA		DUMP DUMPDATA
	DDF		STOREMOD STOREDATA		STOREMOD STOREDATA						DUMP DUMPDATA			DUMP DUMPDATA		DUMP DUMPDATA
	DCI		STOREDATA	STOREMOD STOREDATACI	STOREDATA	STOREMOD STOREDATACI					DUMPDATA	DUMP		DUMPDATA	DUMP	DUMP DUMPDATA
CD	CDS	STORE	STOREDATA	STORECI	STOREDATA	STORECI	STORE	STOREDATA								
	CDD		STOREDATA	STOREDATACI	STOREDATA	STOREDATACI		STOREDATA								
	CDC		STOREDATA	STOREDATACI	STOREDATA	STOREDATACI		STOREDATA	STOREDATACI							
PT	PTS	STORE	STOREDATA	STORECI	STOREDATA	STORECI	STORE	STOREDATA								
	PTD		STOREDATA	STOREDATACI	STOREDATA	STOREDATACI		STOREDATA								
	PTC		STOREDATA	STOREDATACI	STOREDATA	STOREDATACI		STOREDATA	STOREDATACI							

Table 1. Summary of DUP Data Transfer Operations

to the DSF program, core image program, or data file and a corresponding entry is made in LET/FLET only if the name is not found.

When dumping or deleting a DSF program, core image program, or data file from the User/Fixed Area, the DSF program, core image program, or data file is located through LET/FLET using the name specified by the user in the DUP control record.

DCOM INDICATORS

WORKING STORAGE INDICATOR

DCOM contains a Working Storage indicator word for each cartridge on the system. The Working Storage indicator word for a cartridge is set to the disk block count of any DSF program, core image program, or data file placed in Working Storage on that cartridge.

The Working Storage indicator for a cartridge is set (1) at the completion of a DUP operation in which information is transferred to Working Storage and (2) at the completion of any assembly or a successful compilation, at which time the Assembler or FORTRAN Compiler places the assembled/compiled object program in Working Storage.

The Working Storage indicator for a specific cartridge is reset following any STORE operation to the User Area on that cartridge. Because the User Area is increased at the expense of Working Storage, it is assumed that any STORE operation to the User Area overlays some part of Working Storage, that is, that which was stored. Therefore, the Working Storage indicator is reset.

FORMAT INDICATOR

DCOM contains a Format indicator word for each cartridge on the system. The Format indicator word for a cartridge is set to indicate the format of any DSF program, core image program, or data file placed in Working Storage on that cartridge.

The Format indicator for a cartridge is set and reset under the same conditions as the Working Storage indicator for the same cartridge.

TEMPORARY MODE INDICATOR

The temporary mode indicator in DCOM is set by the Supervisor when temporary mode is indicated by the user in the JOB monitor control record (see "JOB" under Monitor Control Records, above). Table 2 shows the DUP operations and the restrictions, if any, when in temporary mode.

Table 2. Restrictions on DUP Operations in Temporary Mode

DUP Operations	Restrictions
DUMP	None
DUMPDATA	None
STORE	None
STORECI	To UA only
STOREDATA	To UA and WS only
STOREDATACI	To UA only
STOREMOD	Not allowed
DUMPLET	None
DUMPFLET	None
DWADR	Not allowed
DELETE	Not allowed
DEFINE FIXED AREA	Not allowed
DEFINE PRINC PRINT	None
DEFINE VOID ASSEMBLER	Not allowed
DEFINE VOID FORTRAN	Not allowed

CONTROL RECORD FORMAT

DUP control records generally follow the format described below. Note that all fields in the control record, except the Count field, are always left-justified and that, unless stated otherwise, all fields are required.

COLUMN 1

Column 1 always contains an asterisk (*). This character identifies the DUP control record.

OPERATION NAME

Columns 2 through 12 (21 in the case of the DEFINE operation) contain the name of the desired DUP operation. Columns 2 through 6 identify the basic operation (STOREDATACI); columns 7 through 12 (or 21) identify the extended operation (STOREDATACI). Where shown, the blank character (b) is required within or following the operation name.

"FROM" AND "TO" SYMBOLS

Columns 13 and 14 contain the "FROM" symbol, that is, the symbol specifying the disk area or I/O device from which information is to be obtained (the source). Columns 17 and 18 contain the "TO" symbol, that is, the symbol specifying the disk area or I/O device to which information is to be transferred (the destination). The symbols that must be used as the "FROM" and "TO" symbols are shown below.

Symbol	Disk Area or I/O Device
UA	User Area, Disk
FX	Fixed Area, Disk
WS	Working Storage, Disk
CD	Card I/O device. If the monitor system has been loaded from paper tape, CD is equivalent to PT.
PT	Paper Tape
PR	Principal print device

When used, the symbols UA, FX, and WS each specify an area on disk but do not identify the cartridge on which the area is to be found.

NAME

Columns 21 through 25 contain the name of the DSF program, core image program, or data file involved in the specified DUP operation. The name may consist of up to five alphameric characters. The first character must be alphabetic, and no embedded blank characters are allowed.

When referencing a DSF program, core image program, or data file already stored on disk, the name must be an exact duplicate of the LET/FLET entry.

COUNT

Columns 27 through 30 contain the count. The count is always a right-justified, decimal integer. The count is defined in the control record layouts for those operations requiring it.

"FROM" AND "TO" CARTRIDGE IDs

Columns 31 through 34 contain the cartridge ID of the cartridge on which is found the disk area from which information is to be obtained, that is, the "FROM" (source) cartridge ID. Columns 37 through 40 contain the cartridge ID of the cartridge on which is found the disk area to which information is to be transferred, that is, the "TO" (destination) cartridge ID.

Either one or both of these cartridge IDs may be omitted. If a cartridge ID is omitted, a search is made of the LET/FLET on each cartridge, starting with the cartridge on logical drive zero (the master cartridge) and continuing through logical drive four. If a cartridge ID is specified, the LET/FLET on the specified cartridge only is searched.

Use of the "FROM" and "TO" cartridge IDs makes it possible for DUP (1) to transfer DSF programs, core image programs, and data files from one cartridge to another without deleting them from the source cartridge, and (2) to operate on a DSF program, core image program, or data file even though the same name appears in the LET/FLET on more than one cartridge.

UNUSED COLUMNS

All unused columns between columns 2 and 40 must be left blank. Columns 41 through 80 are ignored by DUP. These columns are available for user's remarks.

DUP OPERATIONS

The following are descriptions of the various DUP operations. Each description consists of (1) a brief description of the processing performed, (2) a breakdown of the control record for the operation, and (3) a table of the transfers and format conversions possible in the operation.

DUMP

The DUMP operation moves information from the User/Fixed Area on disk to Working Storage or makes information from the User/Fixed Area and Working Storage available in punched card, punched paper tape, or printed format.

The movement of DSF programs from the User/Fixed Area to the output devices is accomplished in two phases; that is, the information is first moved to the Working Storage in use by the monitor programs and then to the output device. Hence, information residing in Working Storage on the cartridge defined in the JOB monitor control record by the Working Storage ID (see "JOB" under Monitor Control Records, above) is destroyed during the DUMP operation. Data files and core image programs are moved directly from the User/Fixed Area to the output devices.

The number of disk blocks to be dumped is obtained from the LET/FLET entry, or, if the dump is from Working Storage, from the appropriate Working Storage indicator in DCOM.

The control record format is described below.

Card Column	Contents	Notes
1-6	*DUMPb	
7-12	Reserved	
13-14	"FROM" symbol	If the dump is from Working Storage and the corresponding Working Storage indicator is zero, an error message is printed.
15-16	Reserved	
17-18	"TO" symbol	If the dump is to cards, each card is checked to see that it is blank before it is punched.
19-20	Reserved	
21-25	Program name	The name is required except when the dump is from Working Storage to the printer.
26-30	Reserved	
31-34	"FROM" cartridge ID	
35-36	Reserved	
37-40	"TO" cartridge ID	
41-80	Not used	

The following is a summary of the information transfers and format conversions performed by DUMP.

Possible Sources, including Formats	Possible Destinations, including Formats
UA (DSF)	WS (DSF)
UA (DSF) WS (DSF)	CD (CDS) PT (PTS) PR (PRD)
UA (DDF) FX (DDF)	WS (DDF) CD (CDD) PT (PTD) PR (PRD)
UA (DCI) FX (DCI)	WS (DCI)
UA (DCI) WS (DCI) FX (DCI)	PR (PRD) CD (CDC) PT (PTC)
WS (DDF)	CD (CDD) PT (PTD) PR (PRD)

DUMPDATA

The DUMPDATA operation moves information from the User/Fixed Area on disk to Working Storage or makes information from the User/Fixed Area and Working Storage available in punched card, punched paper tape, or printed format. The DUMPDATA operation differs from the DUMP operation in that the information, after transfer, is always in a data format.

Information is moved directly from the User/Fixed Area to the output devices. The contents of Working Storage are not changed.

The count in the DUMPDATA control record specifies the number of sectors to be dumped. This

number of sectors is dumped regardless of the length of the DSF program, core image program, or data file, as indicated in the LET/FLET entry or in the Working Storage indicator.

The control record format is described below.

Card Column	Contents	Notes
1-10	*DUMPDATA␣	
11-12	Reserved	
13-14	"FROM" symbol	
15-16	Reserved	
17-18	"TO" symbol	If the dump is to cards, each card is checked to see that it is blank before it is punched.
19-20	Reserved	
21-25	Program name	The name is required except when the dump is from Working Storage to the printer.
26	Reserved	
27-30	Count	The count specifies the number of sectors to be dumped. The count overrides both the contents of the Working Storage indicator and the disk block count in the LET/FLET entry.
31-34	"FROM" cartridge ID	
35-36	Reserved	
37-40	"TO" cartridge ID	
41-80	Not used	

The following is a summary of the information transfers and format conversions performed by DUMPDATA.

Possible Sources, including Formats	Possible Destinations, including Formats
UA (DSF, DDF, DCI)	WS (DDF)
FX (DDF, DCI)	CD (CDD) PT (PTD) PR (PRD)
WS (DSF, DDF, DCI)	CD (CDD) PT (PTD) PR (PRD)

DUMPLET

The DUMPLET operation prints the contents of LET on the principal print device. In addition, the contents of FLET are also printed on the principal print device if a Fixed Area has been defined by the user.

If the name of a DSF program, core image program, or data file is specified in the DUMPLET control record, only the LET/FLET entry corresponding to that name is printed. If a cartridge ID is specified in the control record, the LET/FLET on that cartridge only is printed. Otherwise, the entire contents of both LET and FLET on each cartridge on the system are printed.

The control record format is described below.

Card Column	Contents	Notes
1-8	*DUMPLET	
9-20	Reserved	
21-25	Program name	Use of the name specifies that the LET/FLET entry for that name only is to be printed.
26-30	Reserved	
31-34	"FROM" cartridge ID	If an ID is specified, the LET/FLET on that cartridge only is printed.
35-80	Not used	

DUMPFLET

The DUMPFLET operation prints the contents of FLET on the principal print device.

If the name of a core image program or data file is specified in the DUMPFLET control record, only the FLET entry corresponding to that name is printed. If a cartridge ID is specified in the control record, the FLET on that cartridge only is printed. Otherwise, the entire contents of the FLET on each cartridge on the system are printed.

The control record format is described below.

Card Column	Contents	Notes
1-10	*DUMPFLET␣	
11-20	Reserved	
21-25	Program name	Use of the name specifies that the FLET entry for that name only is to be printed.

Card Column	Contents	Notes
26-30	Reserved	
31-34	"FROM" cartridge ID	If an ID is specified, the FLET on that cartridge only is printed.
35-80	Not used	

STORE

The STORE operation moves information from Working Storage to the User Area or accepts information from the input devices and moves it to Working Storage or the User Area.

All movement of information from the input devices to the User Area is accomplished in two phases; that is, the information is first moved to the Working Storage in use by the monitor programs and then to the User Area. Hence, information residing in Working Storage on the cartridge defined in the JOB monitor control record by the Working Storage ID (see "JOB" under Monitor Control Records, above) is destroyed during the STORE operation.

Since the User Area and Working Storage are adjacent areas, and since the User Area expands as needed into what had been Working Storage, DUP assumes that, on any STORE operation to the User Area from Working Storage on the same cartridge, the contents of Working Storage are destroyed. Therefore, the appropriate Working Storage indicator is reset to zero following the STORE operation to the User Area.

DUP automatically makes the required LET entry (or entries) for each program stored. A LET entry is made for each entry point in the program. DUP supplies the disk block count required in the LET entry for each entry point.

The control record format is described below.

Card Column	Contents	Notes
1-6	*STORE	
7-10	Reserved	
11-12	Subtype (for type 3 and type 4 subprograms only)	See "System Overlays" under Core Load Builder, below.
13-14	"FROM" symbol	If the STORE operation is from Working Storage and the corresponding Working Storage indicator is zero, an error message is printed.
15-16	Reserved	
17-18	"TO" symbol	
19-20	Reserved	
21-25	Program name	The name is required except when the STORE operation is to Working Storage.
26-30	Reserved	
31-34	"FROM" cartridge ID	
35-36	Reserved	
37-40	"TO" cartridge ID	
41-80	Not used	

The following is a summary of the information transfers and format conversions performed by STORE.

Possible Sources, including Formats	Possible Destinations, including Formats
WS (DSF)	UA (DSF)
CD (CDS) PT (PTS)	WS (DSF) UA (DSF)

STOREDATA

The STOREDATA operation moves information from Working Storage to the User/Fixed Area or accepts information from the input devices and moves it to Working Storage or the User/Fixed Area. The input to the STOREDATA operation is assumed by DUP to be in a data format; the output from the STOREDATA operation is always in a data format.

Information is moved directly from the input devices to the User/Fixed Area. The contents of Working Storage are not changed.

DUP automatically makes the required LET/FLET entry. The name specified on the STOREDATA control record is the name used to generate the LET/FLET entry and is the name that must be used in all subsequent references to the data file. DUP supplies the disk block count required in the LET/FLET entry if the source is cards or paper tape. If the source is Working Storage, the sector count specified in the STOREDATA control record is used.

The control record format is described below.

Card Column	Contents	Notes
1-10	*STOREDATA	
11-12	Reserved	
13-14	"FROM" symbol	
15-16	Reserved	
17-18	"TO" symbol	
19-20	Reserved	
21-25	Program name	The name is not required when the STORE operation is from cards or paper tape to Working Storage.
26	Reserved	
27-30	Count	If the source is Working Storage, the count is the decimal number of sectors of data to be stored. This count overrides the contents of the Working Storage indicator. If the source is cards, the count is the decimal number of cards to be read. If the source is paper tape, the count is the decimal number of paper tape records to be read.
31-34	"FROM" cartridge ID	
35-36	Reserved	
37-40	"TO" cartridge ID	
41-80	Not Used	

The following is a summary of the information transfers and format conversions performed by STOREDATA.

Possible Sources, including Formats	Possible Destinations, including Formats
WS (DSF, DDF, DCI)	UA (DDF) FX (DDF)
CD (CDS, CDD, CDC)	UA (DDF) FX (DDF)
PT (PTS, PTD, PTC)	WS (DDF)

STOREDATACI

The STOREDATACI operation moves information from Working Storage to the User/Fixed Area on disk or accepts information from the input devices and moves it to Working Storage or to the User/Fixed Area. If the input is from cards or paper tape, the STOREDATACI operation assumes the input format to be card or paper tape core image format. If the input is from Working Storage (the information has been previously dumped to Working Storage or stored in Working Storage from an input device), the appropriate Format indicator must indicate disk core image format; otherwise, no STORE operation is performed. The output from the STOREDATACI operation is always in disk core image format.

All movement of information from the input devices to the User/Fixed Area is done directly; that is, the transfer is not made via Working Storage. Hence, the contents of Working Storage are not changed by the STOREDATACI operation when storing information from an input device to the User/Fixed Area.

DUP automatically makes the required LET/FLET entry. The name specified on the STOREDATACI control record is the name used to generate the LET/FLET entry and is the name which must be used in all subsequent references to the core image program or data file. DUP computes the disk block count required in the LET/FLET entry from the count specified in the STOREDATACI control record.

The control record format is described below.

Card Column	Contents	Notes
1-12	*STOREDATACI	
13-14	"FROM" symbol	
15-16	Reserved	
17-18	"TO" symbol	
19-20	Reserved	
21-25	Program name	If the STORE operation is to Working Storage, the name is not required.
26	Reserved	
27-30	Count	The count is the number of records in the core image input. The count is not required if the source is Working Storage.
31-34	"FROM" cartridge ID	
35-36	Reserved	
37-40	"TO" cartridge ID	
41-80	Not used	

The following is a summary of the information transfers and format conversions performed by STOREDATACI.

Possible Sources, including Formats	Possible Destinations, including Formats
WS (DCI)	UA (DCI) FX (DCI)
CD (CDC, CDD) PT (PTC, PTD)	WS (DCI) UA (DCI) FX (DCI)

STORECI

The STORECI operation obtains an object program from Working Storage or from an input device, converts it into a core image program using the Core Load Builder, and stores the core image program into the User/Fixed Area.

The Core Load Builder is fetched to build a core image program for the STORECI operation as if execution were to follow; that is, that portion of the core load residing above core location 4095_{10} is placed into core, that portion of the core load residing below core location 4096_{10} is placed into the CIB, and LOCALs and/or SOCALs are placed in Working Storage. The STORECI operation stores all these portions of the core image program into the destination ("TO") area.

The core image program stored in the User/Fixed Area includes the transfer vector built by the Core Load Builder. Neither the disk I/O subroutine nor any COMMON area is included in the core image program stored. Figure 6 shows the layout of a core image program as it is stored in the User/Fixed Area. Note that no scale is intended in this illustration.

DUP automatically makes the required LET/FLET entry for the core image program as it is stored. The name specified on the STORECI control record is the name used to generate the LET/FLET entry and is the name which must be used in all subsequent references to the core image program. DUP obtains from the Core Load Builder the disk block count required in the LET/FLET entry.

The control record format is described below.

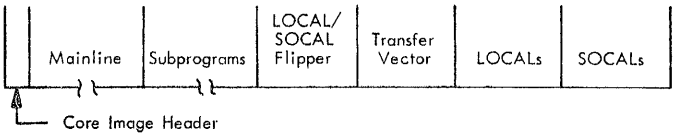


Figure 6. Layout of a Core Image Program Stored in the User/Fixed Area

Card Column	Contents	Notes
1-8	*STORECI	
9	Disk I/O subroutine indicator	This column specifies the disk I/O subroutine to be loaded into core by the Core Image Loader for use by the core load at execution time.

Card Column	Contents	Notes
		Disk Indicator Subroutine 0, 1 DISK1 N DISKN all others, including blank DISKZ
10-12	Reserved	
13-14	"FROM" symbol	If the STORE operation is from Working Storage and the corresponding Working Storage indicator is zero, an error message is printed.
15-16	Reserved	
17-18	"TO" symbol	
19-20	Reserved	
21-25	Program name	
26	Reserved	
27-30	Count	The count is the decimal number of FILES, NOCAL, and LOCAL control records which follow the STORECI control record. This number of records are read by DUP for use by the Core Load Builder before the STORE operation is performed.
31-34	"FROM" cartridge ID	
35-36	Reserved	
37-40	"TO" cartridge ID	
41-80	Not used	

The following is a summary of the information transfers and format conversions performed by STORECI.

Possible Sources, including Formats	Possible Destinations, including Formats
WS (DSF) CD (CDS) PT (PTS)	UA (DCI) FX (DCI)

STOREMOD

The STOREMOD operation moves information from Working Storage into the User/Fixed Area. If the name of the DSF program, core image program, or data file specified on the STOREMOD control record is identical to an entry in LET/FLET (that is, a DSF program, core image program, or data field of the same name already resides in the User/Fixed Area), the information in Working Storage overlays (replaces) that DSF program, core image program, or data file in the User/Fixed Area.

If the name on the STOREMOD control record does not match an entry in LET/FLET, a simple STORE operation is performed (see above).

The STOREMOD operation permits the user to modify a DSF program, core image program, or data file in the User/Fixed Area without changing its name or its relative position within the area. However, the length of the DSF program, core image program, or data file in Working Storage cannot be greater than the length of the DSF program, core image program, or data file that it replaces in the User/Fixed Area.

In the replacement of a DSF program or a data file, no change is made to the LET/FLET entry. In the replacement of a core image program, the LET/FLET entry is updated with the length of the replacement core image program.

The control record format is described below.

Card Column	Contents	Notes
1-10	*STOREMODb	
11-12	Reserved	
13-14	"FROM" symbol	The source is always Working Storage.
15-16	Reserved	
17-18	"TO" symbol	
19-20	Reserved	
21-25	Program name	
26-30	Reserved	
31-34	"FROM" cartridge ID	
35-36	Reserved	
37-40	"TO" cartridge ID	
41-80	Not used	

The following is a summary of the information transfers and format conversions performed by STOREMOD.

Possible Sources, including Formats	Possible Destinations, including Formats
WS (DSF)	UA (DSF)
WS (DDF)	UA (DDF) FX (DDF)
WS (DCI)	UA (DCI) FX (DCI)

DELETE

The DELETE operation removes a specified DSF program, core image program, or data file from the User/Fixed Area. The deletion is accomplished by the removal of the LET/FLET entry (or entries) for the DSF program, core image program, or data file, including the dummy entry for associated padding, if any.

If a DSF program, core image program, or data file is deleted from the User Area, that area is packed so that (1) the areas represented by LET entries are contiguous, and (2) Working Storage can be increased by the amount of disk storage formerly occupied by the deleted DSF program, core image program, or data file.

If a core image program or data file is deleted from the Fixed Area, no packing of that area occurs. The FLET entry for the deleted core image program or data file, including the dummy entry for associated padding, if any, is replaced by a single dummy entry representing the area formerly occupied by the deleted core image program or data file and its padding.

The control record format is described below.

Card Column	Contents	Notes
1-8	*DELETE␣	
9-20	Reserved	
21-25	Program name	
26-30	Reserved	
31-34	"FROM" cartridge ID	The deletion is performed on the specified cartridge only.
35-80	Not used	

DEFINE

The DEFINE operation (1) initially establishes the size of the Fixed Area, (2) increases or decreases the size of the Fixed Area, (3) deletes the Assembler or the FORTRAN Compiler, or both, from the System Area, (4) defines the device to be used as the principal print device for the system, (5) defines the device to be used as the principal I/O device for the system, and (6) defines to the system the size of core.

Definition of a Fixed Area on disk allows the user to store core image programs and data files in fixed locations, which can subsequently be referred to by sector address. The Fixed Area is defined as a whole number of cylinders, two cylinders being the minimum. One cylinder of the Fixed Area is always reserved for FLET.

Increases and decreases in the size of the Fixed Area are made in whole numbers of cylinders. However, the Fixed Area cannot be increased by a number greater than the number of unused cylinders at the end of the Fixed Area. If all core image programs and data files have been deleted from the Fixed Area and the Fixed Area is decreased to less than two cylinders, the remaining Fixed Area, as well as FLET, is deleted.

The control record format for definition of the Fixed Area is described below.

Card Column	Contents	Notes
1-8	*DEFINE␣	
9-18	FIXED␣AREA	
19-26	Reserved	
27-30	Count	In initial definition of the Fixed Area, the count is the decimal number of cylinders to be allocated as the Fixed Area. A minimum of two cylinders must be specified. After initial definition, the count is the number of cylinders by which the Fixed Area is to be increased or decreased in size.

Card Column	Contents	Notes
31	Sign	If the Fixed Area is being decreased, this column contains a minus sign; otherwise, it is blank.
32-36	Reserved	
37-40	Cartridge ID	This ID specifies the cartridge which is to be altered.
41-80	Not Used	

Deletion of the Assembler and/or FORTRAN Compiler causes the specified monitor program(s) to be removed from the System Area on the master cartridge. The System Area is then packed so that following programs and areas occupy the area(s) formerly occupied by the deleted monitor program(s). SLET entries are updated to reflect the new disk storage allocation for the monitor programs. The Reload Table is used to make this adjustment. If the Assembler and/or FORTRAN Compiler is to be deleted, the user must do so before defining the Fixed Area on the master cartridge.

The control record format for deletion of the Assembler and/or FORTRAN Compiler is described below.

Card Column	Contents	Notes
1-8	*DEFINE␣	
9-13	VOID␣	
14-22	ASSEMBLER or FORTRAN␣␣	
23-80	Not used	

Definition of the principal print device establishes the device to be used by the system for the printing of all system messages. DUP copies the system device subroutine for the printer specified in the control record into an area from which all monitor programs obtain the subroutine used to print on the principal printer.

The control record format for definition of the principal print device is described below.

Card Column	Contents	Notes
1-8	*DEFINE␣	
9-20	PRINC␣PRINT␣	
21-24	Device number	The device number is 1403 if the 1403 Printer is to be the principal print device or 1132 if the 1132 Printer is to be the principal print device. If the device number is blank, the Console Printer is assigned as the principal print device.
25-80	Not used	

Definition of the principal I/O device establishes the device (exclusive of the Keyboard) to be used by the system for reading and punching control records, programs, and data. DUP copies the system device subroutine and conversion subroutine for the I/O device specified in the control record into an area from which all monitor programs obtain the subroutine to be used to read from and punch on the principal I/O device.

The control record format for definition of the principal I/O device is described below.

Card Column	Contents	Notes
1-8	*DEFINE␣	
9-20	PRINC␣INPUT␣	
21-24	Device Number	This number is the number of the device to which the IPL is wired.
25-80	Not Used	

Definition of the core size establishes the upper limit of the core storage within which the monitor system is to operate. DUP alters the word in COMMA that the system uses to determine core size to the size specified on the control record.

The control record format for definition of the core size is described below.

Card Column	Contents	Notes
1-8	*DEFINE␣	
9-18	CORE␣SIZE␣	

Card Column	Contents	Notes	
19-22	Size (nominal)	Cols.19-22	Core Size
		4Kbb	4096 words
		8Kbb	8192 words
		16Kb	16384 words
		32Kb	32768 words
23-80	Not Used		

DWADR

The DWADR operation writes a sector address on every sector of Working Storage on the cartridge specified. The operation restores correct disk sector addresses in Working Storage if they have been modified during execution of a user's program. The contents of Working Storage prior to the operation are destroyed.

Following the sector address word, the first two words of each sector contain (in hexadecimal) D120 2663. The next 238 words of each sector all contain Axxx, where xxx is the hexadecimal sector address. The remaining words of each sector contain zeros.

The control record format is described below.

Card Column	Contents	Notes
1-6	*DWADR	
7-36	Reserved	
37-40	Cartridge ID	This ID specifies the cartridge which is to be altered.
41-80	Not used	

The basic language for the Assembler in the monitor system is described in the publication IBM 1130 Assembler Language (Form C26-5927). Therefore, this section contains only a general description of the operation and the control records for the monitor Assembler and the additions to the language provided in the 1130 Disk Monitor System, Version 2.

The monitor Assembler cannot be operated independently of the monitor system; however, the Assembler can be deleted from the monitor system if desired (see "DEFINE" under Disk Utility Program, above).

An ASM monitor control record is used to call the Assembler into operation. The Assembler reads the source program, including control records, from the principal input device. After assembly, the object program resides in Working Storage, and can be (1) called for execution with an XEQ monitor control record, (2) stored in the User/Fixed Area with a DUP STORE or STORECI operation, or (3) punched as a binary deck or tape with a DUP DUMP operation.

ASSEMBLER CONTROL RECORDS

Assembler control records are used to specify options affecting the assembly and the output from it. These control records must precede the source program and can be in any order. Assembler control records can be entered in card or paper tape form along with the source program deck or tape, or they may be entered from the Keyboard along with the source statements (see "TYP" under Monitor Control Records, above).

All Assembler control records have the following format:

Column 1: * (asterisk)
2-71: Option

If an Assembler control record contains an asterisk in column 1, but the option does not agree, character for character, with its valid format, as described below, the asterisk is replaced by a minus sign on the control record listing. The erroneous control record is ignored in the assembly. The option is not performed; however, no error results.

Assembler control records can be written in free form; that is, any number of blanks may occur between the characters of the option. However, only one blank must separate the last character in the option and the first character of any required numeric field. Remarks may be included in the control record following the option or numeric field; however, at least one blank must separate the last character of the option or numeric field and the remarks.

TWO PASS MODE

This control record causes the Assembler to read the source deck (or tape) 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) fills the Working Storage area of disk.

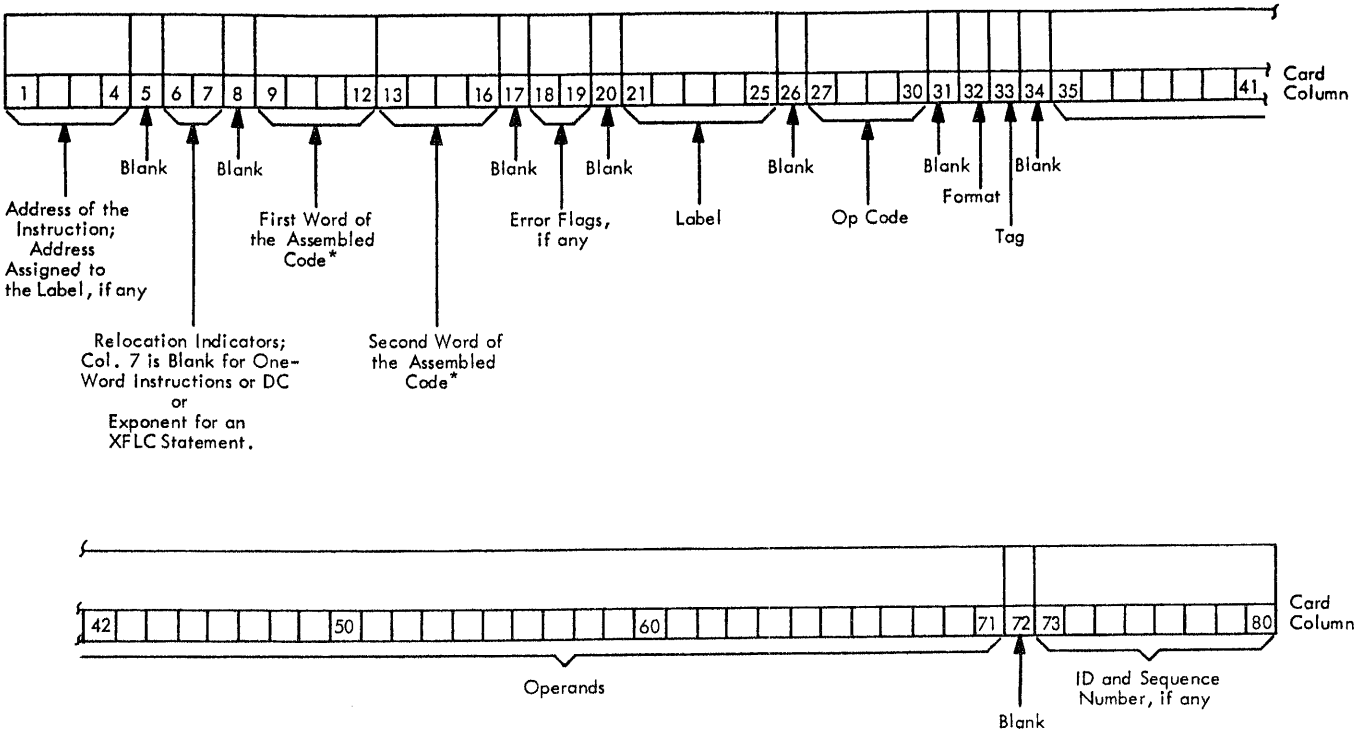
This control record is ignored when source statements are entered through the Keyboard.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	TWO PASS MODE	
72-80	Not used	

LIST

This control record causes the Assembler to provide a printed listing on the principal print device (Console Printer, 1403 Printer, or 1132 Printer). The format of the printed listing corresponds to that of the list deck (see Figure 7).



*For EBC statements, columns 9-12 contain the number of EBC characters.
For BSS and BES statements, columns 9-12 contain the number of words reserved for the block.
For ENT, ILS, and ISS statements, columns 9-16 contain the entry label in packed EBCDIC code.

Figure 7. List Deck Format

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	LIST	
72-80	Not used	

LIST DECK

This control record causes the Assembler to punch a list deck, but only if the principal I/O device is a card reader. This option requires two passes (TWO PASS MODE). The list deck format is shown in Figure 7. Object information is punched into columns 1-19 of the source deck in pass 2 to make the list deck.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	LIST DECK	
72-80	Not used	

LIST DECK E

This control record causes the Assembler to punch the assembly error codes only (columns 18-19) in the list deck output (see LIST DECK).

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	LIST DECK E	
72-80	Not used	

PRINT SYMBOL TABLE

This control record causes the assembler to provide a printed listing of the symbol table on the principal print device (Console Printer, 1403 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.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	PRINT SYMBOL TABLE	
72-80	Not used	

PUNCH SYMBOL TABLE

This control record causes the assembler to punch the symbol table as a series of EQU source cards. These cards can be used as source input to the system symbol table when the SAVE SYMBOL TABLE control record is used with an assembly in which they are included. The symbol table is punched only when the principal I/O device is a card reader.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	PUNCH SYMBOL TABLE	
72-80	Not used	

SAVE SYMBOL TABLE

This control record causes the assembler to save the symbol table generated in this assembly on the disk as a System Symbol Table. The System Symbol Table is saved until the next assembly having a 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 that this assembly be absolute. If any assembly errors are detected, or if the symbol table exceeds the allowable size of the System Symbol Table, the symbol table is not saved as a System Symbol Table, and an assembly error message is printed.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	SAVE SYMBOL TABLE	
72-80	Not used	

SYSTEM SYMBOL TABLE

This control record causes the assembler to copy the System Symbol Table (previously built by a SAVE SYMBOL TABLE assembly) into the symbol table for this assembly before the assembly begins. 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 it is used together with the SAVE SYMBOL TABLE control record when it is desired to add symbols to the System Symbol Table. All symbols in the System Symbol Table have absolute values.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	SYSTEM SYMBOL TABLE	
72-80	Not used	

LEVEL

This control record specifies the interrupt levels serviced by an ISS and, hence, the associated ILS subroutines. It is required for the assembly of an ISS routine. The interrupt level number is a decimal number in the range 0-5. If the device operates on more than one level of interrupt (for example, 1442 Card Read Punch), one LEVEL control record is required for each level of interrupt. At least one blank must separate the word LEVEL and the interrupt level number.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	LEVEL␣n	n is an interrupt level number.
72-80	Not used	

OVERFLOW SECTORS

This control record specifies the number of sectors of Working Storage for possible symbol table overflow to be allowed by the Assembler. The number of overflow sectors is a decimal number in the range 1-32. If this number is zero or blank, no overflow sectors are allowed. If the number is greater than 32, thirty-two overflow sectors are allowed. If this control record is not used, no overflow sectors are allowed.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	OVERFLOW SECTORS␣nn	nn is the number of sectors assigned to symbol table overflow.
72-80	Not used	

COMMON

This control record specifies the decimal length (in words) of COMMON as defined by a previously executed FORTRAN core load. Use of this control record provides for a COMMON area to be saved in linking from a FORTRAN mainline to an assembler mainline and back to a FORTRAN mainline. At least one blank must separate the word COMMON and the decimal number.

The format of this control record is described below.

Card Column	Contents	Notes
1	* (asterisk)	
2-71	COMMON␣nnnnn	nnnnn is the number of words of COMMON to be saved between links.
72-80	Not used	

Figure 8 shows the layout of an Assembler input deck.

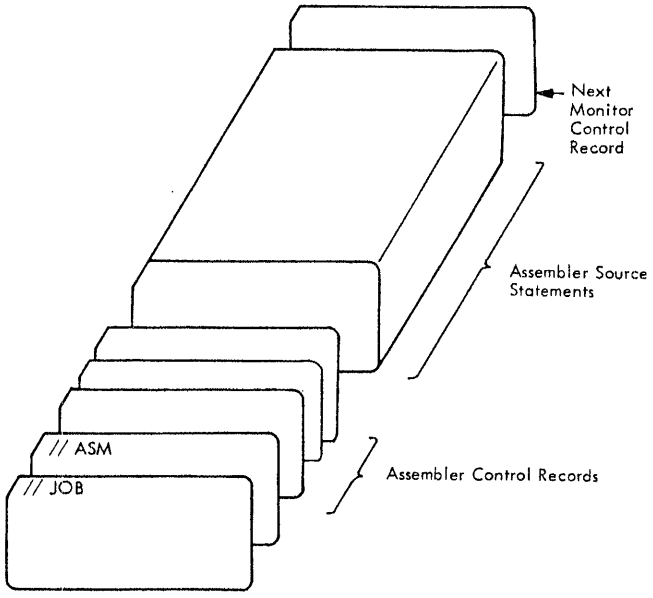


Figure 8. Layout of an Assembler Input Deck

ORIGIN OF MAINLINES

The origin of a relocatable mainline program is always set at relative zero unless otherwise specified in the source program.

The origin of an absolute mainline program, if not otherwise specified in an ORG statement, is set at the end of DISKN. If the program requires DISKN or DISK1, the origin is set at the end of the requested disk I/O subroutine. If no disk I/O subroutine is used by the program, the origin is set at the end of DISKZ.

ASSEMBLER PAPER TAPE FORMAT

The paper tape input to the Assembler is punched in PTTC/8 code, 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 leading 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 new line characters (code DD). The delete character (code 7F) is ignored whenever it is read, but the reader stop character (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 LANGUAGE

The following information describes the additions to the basic language provided in the 1130 Disk Monitor System, Version 2. This information supplements the publication IBM 1130 Assembler Language (Form C26-5927).

NEW (EXTENDED) MACHINE INSTRUCTION MNEMONICS

A list of the IBM 1130 machine instruction mnemonics including the new (extended) mnemonics is given in Table 3.

Branch or Skip on Condition

The mnemonic Branch or Skip on Condition (BSC) has been extended to simplify the coding of conditional transfers.

Skip on Condition (SKP). The condition codes (+, -, Z, E, O, and C) are specified as with a short BSC instruction.

Branch Unconditionally (B). If the Format field contains an L or I, the BSC operation code is used with bit 5 set to one. Condition codes are not allowed after the address expression in the Operand field. If the Format field is left blank or contains an X, the MDX operation code is used, and the expression in the Operand field is used to form the displacement.

Branch Accumulator Positive (BP). Condition codes for accumulator zero (Z) and accumulator negative (-) are set to one.

Branch Accumulator Not Positive (BNP). Condition code for accumulator positive (+) is set to one.

Branch Accumulator Negative (BN). Condition codes for accumulator zero (Z) and accumulator positive (+) are set to one.

Branch Accumulator Not Negative (BNN). Condition code for accumulator negative (-) is set to one.

Branch Accumulator Zero (BZ). Condition codes for accumulator positive (+) and accumulator negative (-) are set to one.

Branch Accumulator not Zero (BNZ). Condition code for accumulator zero (Z) is set to one.

Branch on Carry (BC). Condition code for Carry indicator off (C) is set to one.

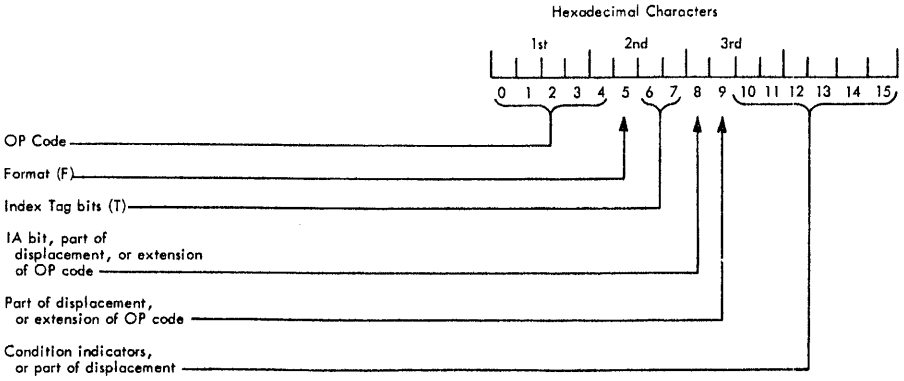
Branch on Overflow (BO). Condition code for Overflow indicator off (O) is set to one.

Table 3. Machine Instruction Mnemonics

Mnemonic		OP Code (Hexadecimal Representation)[1]	Instruction
Load and Store			
LD		C00	Load Accumulator
LDD		C80	Load Double
LDX		600	Load Index
LDS*		200	Load Status
STO		D00	Store Accumulator
STD		D80	Store Double
STX		680	Store Index
STS		280	Store Status
Arithmetic			
A		800	Add
AD		880	Add Double
S		900	Subtract
SD		980	Subtract Double
M		A00	Multiply
D		A80	Divide
AND		E00	And
OR		E80	Or
EOR		F00	Exclusive Or
MDM	†5	740	Modify Memory
Branch			
B	†4	700 or 4C0	Branch
BSI		400	Branch and Store Instruction Counter
BSC		480	Branch or Skip Conditionally
BP	†6	4C30	Branch Accumulator Positive
BNP	†6	4C03	Branch Accumulator Not Positive
BN	†6	4C28	Branch Accumulator Negative
BNN	†6	4C10	Branch Accumulator Not Negative
BZ	†6	4C18	Branch Accumulator Zero
BNZ	†6	4C20	Branch Accumulator Not Zero
BC	†6	4C02	Branch on Carry
BO	†6	4C01	Branch on Overflow
BOD	†6	4C04	Branch Accumulator Odd
SKP*	†	480	Skip on Condition(s)
BOSC	2	484	Branch Out or Skip Conditionally
MDX		700	Modify Index and Skip
Shift			
SLA*		100	Shift Left Accumulator
SLT*		108	Shift Left Accumulator and Extension
SLC*		10C	Shift Left and Count Accumulator and Extension
SLCA*		104	Shift Left and Count Accumulator
SRA*		180	Shift Right Accumulator
SRT*		188	Shift Right Accumulator and Extension
RTE*		18C	Rotate Right
XCH*	†3	18D	Exchange Accumulator and Extension
Input/Output			
XIO		080	Execute I/O
Miscellaneous[3]			
NOP*		100	No Operation
WAIT*		300	Wait

*Valid in short format only
†Not included in card/paper tape Assembler or monitor Assembler, Version 1

1. The hexadecimal representation of the machine operation code is derived from the instruction format in the manner shown below. Bits 5, 6, 7, 10, and 11 are assumed to be zeros because they do not enter into the makeup of any operation codes.
2. Same as BSC with Bit 9 set to one.
3. An operand should not be specified.
4. When branch is short (Blank or X format), this operation code is assembled as an MDX (700). If the branch is long (L or I format), this operation code is assembled as a BSC with Bit 5 set to one (4C0).
5. This instruction is automatically assembled as a long instruction (L is not required in the format field). Note that an attempt to use indirect addressing will result in a syntax error. Indexing is not permitted with this extended operation code.
6. Extended conditional branch operation codes are assembled automatically as long instructions. Note that the proper condition code bits are preset, and further condition bits may not be specified following the operand.



Branch Accumulator Odd (BOD). Condition Code for accumulator even (E) is set to one.

NOTE: Condition codes may not be used with any of the above instructions, except SKP, since the condition code is implicit in the extended mnemonic. The conditional branch instructions (all except SKP and B) are always assembled as long instructions; thus, the Format field need not contain an L, although the instruction is not classed as an error if L is specified. Indirect addressing may be specified.

Modify Index and Skip

The mnemonic Modify Index and Skip (MDX) has been extended to provide for incrementing or decrementing the contents of a core storage location.

Modify Memory (MDM). Contents of the location specified by the first operand is incremented or decremented by the value of the second operand. The second operand must be in the range -128 to +127.

NOTE: This instruction is always assembled as a long instruction; thus, the Format field need not contain an L, although the instruction is not classed as an error if L is specified. Indexing and indirect addressing may not be specified.

Rotate Right

The mnemonic Rotate Right (RTE) has been extended to provide a means for exchanging the Accumulator and Extension.

Exchange Accumulator and Extension (XCH). Exchange is identical to a RTE of 16. No operand is specified with this instruction.

Examples of the new (extended) machine instruction mnemonics are shown in Table 4.

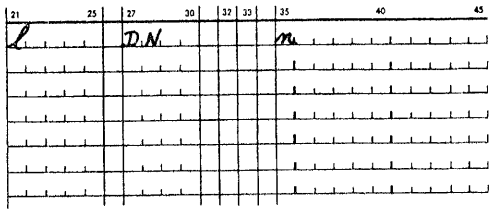
NEW ASSEMBLER INSTRUCTIONS

The Assembler instruction set has been expanded by the addition of eight new statements. These fall into three categories; New Data Definition Statements, New Listing Control Statements, and New Monitor System Statements.

New Data Definition Statements

DN – Define Name

The Define Name statement is used to convert a name specified in the Operand field of the statement to a name in name code in the object program. The format of this statement is shown below:



where

l is any valid label (optional),

n is any valid label or name.

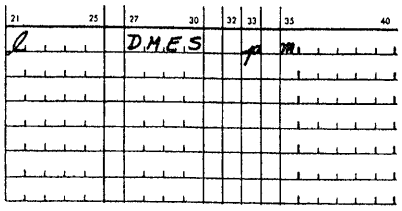
If a label is used, the address assigned to it is the location of the first word of the two words generated and is equal to the current value of the Location Assignment Counter. Columns 32 and 33 must be blank. The operand can have up to five characters that comply with the rules for writing labels. The name to be converted must be left-justified in the Operand field. If remarks are used, one blank must be left between the operand and the remarks. The Location Assignment Counter is incremented by two for this statement.

DMES – Define Message

The DMES statement is used to store a message within a program in a form that is acceptable to the printer output subroutines. The format of the DMES statement follows:

Table 4. Examples of New (Extended) Machine Instruction Mnemonics

New Instruction Statements	Equivalent Statements	Operations Performed
S.K.P. +,	B.S.C. +,	Skip if accumulator is positive
S.K.P. +,-	B.S.C. +,-	Skip if accumulator is positive or negative
S.K.P. Z,	B.S.C. Z,	Skip if accumulator is zero
S.K.P. O,	B.S.C. O,	Skip if Overflow indicator is on
S.K.P. C,	B.S.C. C,	Skip if Carry indicator is on
S.K.P. +,-,C,	B.S.C. +,-,C,	Skip if accumulator is positive or negative or if Carry indicator is on
B. EXIT	M.D.X. EXIT	Branch unconditionally to EXIT, where EXIT must be within normal displacement range.
B. L ALPH	B.S.C. L ALPH	Branch unconditionally to ALPH
BZ. BETA	B.S.C. L BETA,+,-	Branch to BETA if accumulator is zero
BN. BETA	B.S.C. L BETA,Z,	Branch to BETA if accumulator is negative
BNZ. I BETA	B.S.C. I BETA,Z,	Branch indirectly to BETA (i.e., the address specified by contents of BETA) if accumulator is non-zero
BN. RTNA	B.S.C. L RTNA,Z,+	Branch to RTNA if accumulator is negative
BNN. RTNB	B.S.C. L RTNB,-	Branch to RTNB if accumulator is non-negative (zero or positive)
BP. SUB@	B.S.C. L SUB@,Z,-	Branch to SUB@ if accumulator is positive
BP. I SUB$	B.S.C. I SUB$,Z,-	Branch indirectly to SUB$ (i.e., the address specified by the contents of SUB$) if accumulator is positive
BNP. SUB#	B.S.C. L SUB#,+,	Branch to SUB# if accumulator is non-positive (zero or negative)
BC. ENTR+1	B.S.C. L ENTR+1,,C	Branch to ENTR+1 if Carry indicator is on
BC. I 1 Ø	B.S.C. L 1 Ø,C	Branch indirectly to address specified by contents of index register 1 if Carry indicator is on
BO. 2 5,	B.S.C. 2 5,O	Branch to address specified by contents of index register 2 plus 5 if Overflow indicator is on
BOD. $AFE	B.S.C. L $AFE,E	Branch to $AFE if accumulator is odd
MDM. SAVA,+5	M.D.X. L SAVA,+5	Increment contents of core location SAVA by 5
MDM. /1D6A,100	M.D.X. L /1D6A,100	Increment contents of core location $1D6A_{16}$ by 100
MDM. A,-12	M.D.X. L A,-12	Decrement contents of core location A by 12
X.C.H.	R.T.E. 16	Exchange the accumulator and extension (rotate right 16)



where

l is any valid label (optional),

p is the printer type code,

m is any string of valid message and control characters.

If a label is present, it is assigned to the location of the first word generated. The Tag field (column 33) is used to specify the printer type code:

Tag	Printer
ↄ or 0	Console Printer
1	1403 Printer
2	1132 Printer

If the Tag field (printer type code) contains a character other than blank, zero, one, or two, an error results and the message is stored two EBCDIC characters per word.

The Operand field contains the control and message characters. Remarks are permitted only after an 'E or 'ↄ control character.

The output generated by one DMES statement cannot exceed 60 words (120 characters). If an odd number of characters is generated, the last word is filled in with a blank, except when the statement ends with 'ↄ. In this case, the first character of the next DMES statement is used to fill out the word.

Control characters are used to specify certain printer operations and to define message parameters. Each control character is actually two characters, the first of which is always an apostrophe. The apostrophe is a control delimiter and therefore is not included in the character count. The control characters and their functions or meanings are as follows:

Control Character	Function or Meaning
'X	Blank (or space)
'T	Tabulate
'D	Backspace
'B	Print black
'A	Print red
'S	Space (or blank)
'R	Carriage return
'L	Line feed
'F	Repeat following character
'E	End of message
'ↄ	(ↄ=blank) continue text with next DMES statement

All the above characters can be used when the printer is the Console Printer. Only 'E, 'F, 'S, 'X and 'ↄ are valid control characters when the 1132 or 1403 Printer is specified; any other control characters are considered as errors.

The characters 'X and 'S are interchangeable. A blank character is generated for either 'X or 'S if the 1132 or 1403 Printer is specified; a space is generated for either 'X or 'S if the Console Printer is specified.

The character 'F (repeat following character) refers only to message characters. The control characters themselves, except 'A, 'B, 'E, and 'ↄ, can be repeated up to 99 times by inserting a number (1-99) between the apostrophe and unique control definition character. For example, '32S results in 32 space characters being inserted in the generated message.

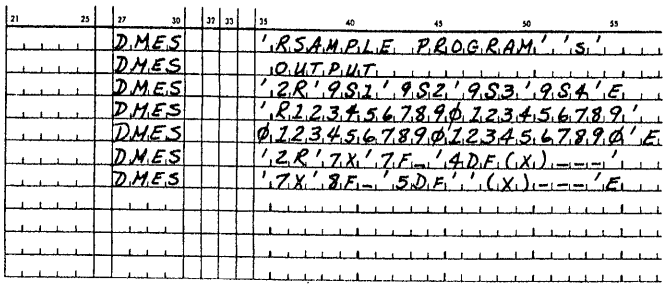
The character 'E is used to designate the end of the message line. The character 'ↄ is used to designate that the message is continued on the following DMES statement. If neither 'E nor 'ↄ is included, 'E is assumed to follow column 71. DMES statements that end with 'ↄ must be followed by another DMES statement.

Text apostrophes are generated by writing two successive apostrophes.

The message characters can be any valid character for the printer being used. Invalid characters are replaced with blanks.

The following example illustrates the DMES statement.

Assembler input:



Printed output:

```
SAMPLE PROGRAM'S OUTPUT

          1         2         3         4
1234567890123456789012345678901234567890

     F(X)                F'(X)
```

Note that the device code specified in the preceding example is blank in order to generate a message for the Console Printer.

DSA – Define Sector Address

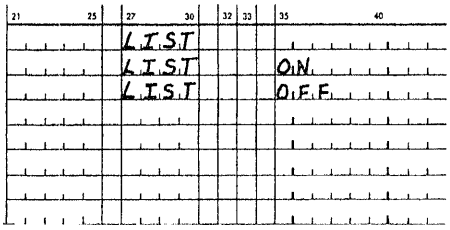
The DSA statement is specified as stated in the publication IBM 1130 Assembler Language (Form C26-5927), except for the following change: the third word generated for a data file is no longer zero; instead, it is the sector count of the file.

New List Control Statement

These list control statements – LIST, SPAC, and EJCT – provide the user with the means to control the assembler output listing; however, these statements themselves are never printed on the output listing by the assembler.

LIST – List Segments of Program

The LIST statement allows the user to list certain segments of a program and to avoid listing other segments. The three variations of the LIST statement are shown below:



The Label, Tag, and Format fields are not used with the LIST statement and should be left blank. The Operand field may be left blank or may contain the operand ON or OFF.

The LIST statement does not cause the Location Assignment Counter to be incremented.

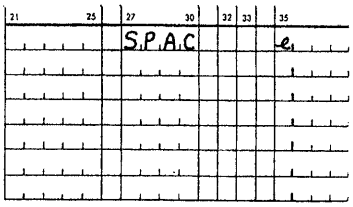
If a LIST statement with the operand ON is encountered, the following statements (excluding list control statements), up to the next LIST statement, are listed by the Assembler.

If a LIST statement with the operand OFF is encountered, the following statements, up to the next LIST statement, are not listed by the Assembler.

If a LIST statement with no operand is encountered, the Assembler assumes an operand depending on the use of the LIST control record. If the LIST control record preceded the assembly, the ON operand is assumed and the Assembler acts accordingly. If the LIST control record did not precede the assembly, the OFF operand is assumed and the Assembler acts accordingly.

SPAC – Space Listing

The SPAC statement is used to insert one or more blank lines in the listing immediately following the SPAC statement. The format of the SPAC statement is as follows:



where e is any valid positive expression.

The Label, Format, and Tag fields are not used and should be left blank.

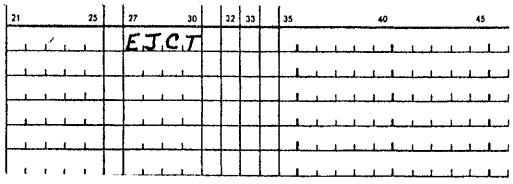
The number of blank lines inserted in the listing is determined by the operand in the statement. The operand can be any valid expression. The operand (expression) value must be positive; otherwise, the Assembler ignores the statement.

When the number of blank lines specified exceeds the number of lines left on the page, the page is spaced to the bottom, a restore occurs, a new heading is printed, and spacing is resumed until the number of blank lines specified has been exhausted.

The SPAC statement does not cause the Location Assignment Counter to be incremented.

EJCT – Start New Page

The EJCT statement causes the next line of the listing to appear at the top of a new page following the page heading. The format of the EJCT statement is as follows:



The Label, Tag, Format, and Operand fields are not used and should be left blank.

A page overflow occurs immediately following the EJCT statement. EJCT statements may be used in succession to obtain blank pages (except for the headings printed).

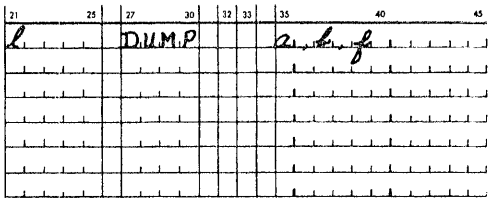
The EJCT statement does not cause the Location Assignment Counter to be incremented.

New Monitor System Statements

DUMP – Dump and Terminate Execution

The DUMP statement provides an entry to the DUMP program (see Supervisor), which prints the contents of core storage on the principal print device in hexadecimal format.

The DUMP statement allows for flexible specification of the upper and lower limits to be dumped without altering core storage. The DUMP program, when called by a DUMP statement, executes a CALL EXIT following the printout. The DUMP statement is written as follows:



where

l is any valid label (optional),

a is any valid expression specifying the lowest-addressed core location to be dumped,

b is any valid expression specifying the highest-addressed core location to be dumped,

f is the dump format code (either a blank or a zero).

The label, if used, is assigned the location of the first of the six words generated. The Tag and Format fields must be left blank.

The format of the DUMP program output is as follows:

AAAA xxxx xxxx xxxx ≈ xxxx xxxx xxxx

The contents (xxxx) of 16 core storage locations are printed per line. At the left is the address (AAAA) of the first location printed on that line.

The Location Assignment Counter is incremented by six for a DUMP statement.

The DUMP statement is translated by the Assembler into a long BSI instruction branching to the address of the DUMP entry point in the Skeleton Supervisor. The parameters (operands) are converted into three data words; the first is the starting location of the core dump, the second is the ending location of the core dump, and the third, the format indicator (always zero). Unspecified operands are assigned values of: location zero for the first data word; the highest core location for the second data word. Following the data words the Assembler generates a short branch to the EXIT entry point in the Skeleton Supervisor.

A DUMP statement can be used at any point in a program; however, the user is reminded that DUMP causes a terminal DUMP to be printed. At the completion of the dump printout, the branch to EXIT is executed, thus transferring control to the Skeleton Supervisor for processing of the next job or subjob.

PDMP – Dump and Continue Execution

The PDMP statement provides the ability to dump core storage between specified limits and to continue execution. The core dump is printed on the principal print device without altering core. The PDMP statement is specified in the same way as DUMP, except that PDMP appears in columns 27-30 instead of DUMP.

The PDMP statement is translated by the Assembler into a long BSI instruction branching to the DUMP entry point in the Skeleton Supervisor. The parameters (operands) are converted as described in the DUMP statement (see above).

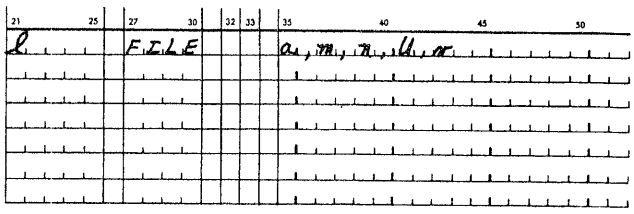
Upon completion of the printout of the core dump, control is returned to the next instruction following the PDMP statement to continue execution.

FILE – Define Disk File

The FILE statement specifies to the Assembler the file identification, the number of file records in a file, and the size of each record in a disk data file that will be used with a particular mainline and its associated subprograms.

The FILE statement is used to divide the disk into files. As the core load is constructed by the Core Load Builder, these files are equated to data files already assigned in the User/Fixed Area or to files in Working Storage (see "FILES" under Supervisor Control Records, above).

The FILE statement must not appear in a subprogram; it is permitted only in a relocatable mainline program. Therefore, all subprograms used by the mainline must use the defined files of the mainline. The format of the FILE statement is as follows:



where

l is any valid label (optional),

a is the file identification number, a decimal integer in the range 1-32767,

m is a decimal integer that defines the number of records in the file,

n is a decimal integer in the range 1-320 that defines the length (in words) of each record in the file,

U is a required constant, specifying that the file must be read/written with no data conversion,

v is the associated variable, the label of a core location (variable) defined elsewhere in the program.

FILE statements must precede all other statements except HDNG, EPR, SPR, EJCT, SPAC, and LIST in the source program. The label, if used, is assigned the location of the first word of the seven words generated. The Format and Tag fields are not used and should be left blank.

Each FILE statement causes the Location Assignment Counter to be incremented by seven. The data stored in these seven words, which constitute a DEFINE FILE Table entry in the object program is as follows:

Word	Contents
1	a, the file identification number
2	m, the number of records per file
3	n, the record length (in words)
4	The address of the associated variable, v.
5	Zeros. This word is filled by the Core Load Builder with the absolute sector address of the data file if the file is already stored in the User or Fixed Area. If the file is assigned to Working Storage by the Core Load Builder, the sector address is relative to the beginning of Working Storage.
6	r, the number of records per sector. The number, computed by the Assembler, is the quotient of $\frac{320}{n}$ (remainder ignored)
7	b, the number of disk blocks per file. This number, computed by the Assembler, is the quotient of $\frac{16(m)}{r}$ (remainder ignored)

FORTRAN COMPILER

The basic language for the FORTRAN Compiler in the monitor system is described in the publication IBM 1130 Basic FORTRAN IV Language (Form C26-5933). Therefore, this section contains only a general description of the operation and the control records for the monitor FORTRAN Compiler and the additions to the basic language provided in the 1130 Disk Monitor System, Version 2.

The FORTRAN Compiler cannot be operated independently of the monitor system; however, the compiler can be deleted from the monitor system if desired (see "DEFINE" under Disk Utility Program, above).

An FOR monitor control record is used to call the FORTRAN Compiler into operation. The compiler reads the source program, including control records, from the principal input device. After compilation, the object program resides in Working Storage and can be (1) called for execution with an XEQ monitor control record, (2) stored in the User/Fixed Area with a DUP STORE or STORECI operation, or (3) punched as a binary deck or tape with a DUP DUMP operation.

For 1130 FORTRAN I/O logical unit definitions, the I/O unit numbers are defined in Table 5.

FORTRAN CONTROL RECORDS

Before a FORTRAN program is compiled, the user can specify certain options affecting both the compilation and execution of the program by means of control records. These control records must precede the source program and can be in any order.

FORTRAN control records can be entered in card or paper tape form along with the source program deck or tape, or they may be entered from the Keyboard along with the source statements (see "TYP" under Monitor Control Records, above). The IOCS and NAME control records can be used only in mainline programs; the others can be used in both mainline programs and subprograms.

All FORTRAN control records have the following format:

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

If a FORTRAN control record contains an asterisk in column 1, but the option does not agree, character for character, with its valid format, as described below, the asterisk is replaced by a minus sign on the control record listing. The erroneous control record is ignored in the compilation. The option is not performed; however, no error results.

FORTRAN control records can be written in free form; that is, any number of blanks may occur between the characters of the option. No remarks are allowed.

IOCS

This control 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 control record can appear only in the mainline program, it must include

Table 5. FORTRAN Logical I/O 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.
5	1403 Printer	Output only	1 carriage control + 120
6	Keyboard	Input only	80
7	1627 Plotter	Output only	120
8	2501 Card Reader	Input only	80
9	1442 Card Punch	Output only	80
10	Disk	Input/output without Data Conversion	320

all the I/O devices used by all FORTRAN subprograms that are called. The device names must be in parentheses with a comma between each name. The valid names and the devices to which they correspond are listed below:

Name	Device
CARD	1442 Card Read Punch, Models 6 and 7
2501 READER	2501 Card Reader
1442 PUNCH	1442 Card Punch, Model 5
TYPEWRITER	Console Printer
KEYBOARD	Keyboard
1132 PRINTER	1132 Printer
1403 PRINTER	1403 Printer
PAPER TAPE	1134/1055 Paper Tape Reader Punch
PLOTTER	1627 Plotter
DISK	Disk

Note that CARD is used for the 1442 Card Read Punch, Models 6 and 7 and that 1442 PUNCH is used for the 1442 Card Punch, Model 5. These two names are mutually exclusive.

Subprograms which are a part of a FORTRAN core load but which are written in Assembler language can use any I/O subroutine for any device that is not specified on the IOCS control record.

Any number of IOCS control records can be used to specify the required device names.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	IOCS (d, d, . . . d)	d is a valid device name (see table above).
73-80	Not used	

LIST SOURCE PROGRAM

This control record causes the compiler to list the source program on the principal print device as it is read in.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	LIST SOURCE PROGRAM	
73-80	Not used	

LIST SUBPROGRAM NAMES

This control record causes the compiler to list the names of all subprograms (including EXTERNAL subprograms) called directly by the compiled program on the principal print device.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	LIST SUBPROGRAM NAMES	
73-80	Not used	

LIST SYMBOL TABLE

This control record causes the compiler to list the following items on the principal print device:

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

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	LIST SYMBOL TABLE	
73-80	Not used	

LIST ALL

This control record causes the compiler to list the source program, subprogram names, and symbol table on the principal print device. If this control record is used, the other LIST control records are not required.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	LIST ALL	
73-80	Not used	

EXTENDED PRECISION

This control record causes the compiler to store variables and real constants in three words instead of two and to generate linkage to extended precision subprograms.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	EXTENDED PRECISION	
73-80	Not used	

ONE WORD INTEGERS

This control record causes the compiler to allocate one word of storage for integer variables 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 to the USASI Basic FORTRAN standard for data storage and may require modification in order to be used with other FORTRAN systems.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	ONE WORD INTEGERS	
73-80	Not used	

NAME

This control record causes the compiler to print the specified program name on the listing. The name is five consecutive characters (including blanks) starting at the first non-blank column following NAME. This control record is used only on mainline programs, since subprogram names are taken from the FUNCTION or SUBROUTINE statement.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	NAME xxxxx	xxxxx is the name of the mainline object program.
73-80	Not used	

HEADER INFORMATION

This control record causes the compiler to print the information in columns 3-72 at the top of each page of compilation printout when an 1132 or a 1403 Printer is the principal print device.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2	*(asterisk)	
3-72	Any string of characters	
73-80	Not used	

ARITHMETIC TRACE

This control record causes the compiler to generate linkage to the trace subprograms, 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.

If tracing is requested, an IOCS control record must also be present to indicate that either the Typewriter (that is, the Console Printer), 1132 Printer, or 1403 Printer is needed. If more than one print device is specified in the IOCS control record, the fastest device is used for tracing.

The traced value for a variable to the left of an equal sign of an arithmetic statement is printed with one leading asterisk.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	ARITHMETIC TRACE	
73-80	Not used	

TRANSFER TRACE

This control record causes the compiler to generate linkage to the trace subprograms, which are executed whenever an IF statement or Computed GO TO 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 GO TO index is printed.

If tracing is requested, an IOCS control record must also be present to indicate that either the Typewriter (that is, the Console Printer), 1132 Printer, or 1403 Printer is needed. If more than one print device is specified in the IOCS control record, the fastest device is used for tracing.

The traced value for the expression in an IF statement is printed with two leading asterisks. The traced value for the index of a Computed GO TO statement is printed with three leading asterisks.

The format of this control record is described below.

Card Column	Contents	Notes
1	*(asterisk)	
2-72	TRANSFER TRACE	
73-80	Not used	

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 TSTOP to stop tracing or a CALL TSTRT to start tracing. Thus, tracing occurs at execution time only if:

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

Figure 9 shows the layout of a FORTRAN Compiler input deck.

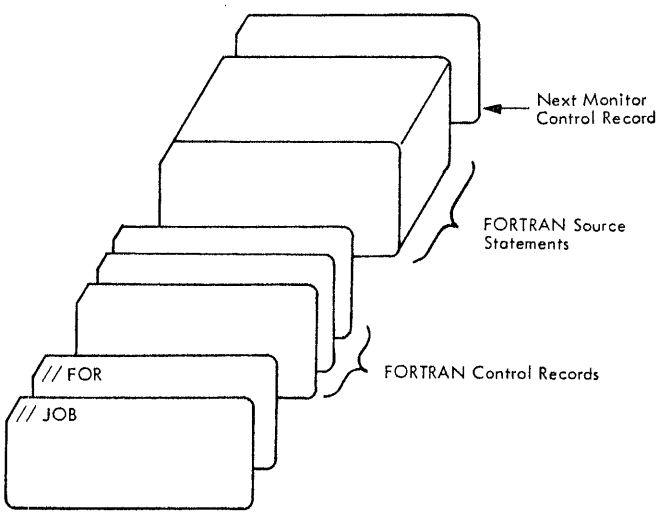


Figure 9. Layout of a FORTRAN Compiler Input Deck

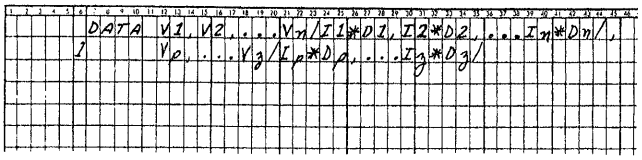
FORTRAN LANGUAGE

The following information describes the additions to the basic FORTRAN IV language provided in the 1130 Disk Monitor System, Version 2. This information supplements the publication IBM 1130 Basic FORTRAN IV Language (Form C26-5933).

DATA STATEMENT

The DATA statement is used to define initial values of variables and array elements assigned to areas other than COMMON. Values assigned to variables or array elements during execution override values assigned via the DATA statement.

General Form



where

V1 through Vz are variables or subscripted variables. Subscripts must be integer constants.

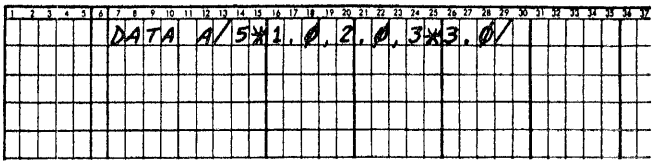
D1 through Dz are data constants. They may be integer, real, hexadecimal, or literal data constants. Integer and real constants may be specified as negative. For the valid variable and constant combinations, see Data-Variable Combinations, below.

I1 through Iz are optional, unsigned integer constants which indicate the number of variables and/or array elements which are to be assigned the value of the data constant. They are separated from the data constants by asterisks. Each data constant must be of the same type (integer or real) as its corresponding variable.

The slash is used to separate and enclose data constants. When an array name is specified, constants are assigned starting at the element specified and continuing toward the end of the array. If no subscript is specified, the first element is assumed to be the starting element.

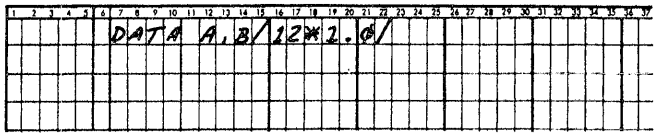
The DATA statement is a specification statement and, as such, must precede all statement function definition statements and the first executable statement in the source program. Also, the DATA statements cannot be intermixed with EQUIVALENCE statements.

Example 1:



If A is a nine-element array for real variables, the first five elements are initialized to 1.0, the sixth to 2.0, and the remaining three to 3.0.

Example 2:



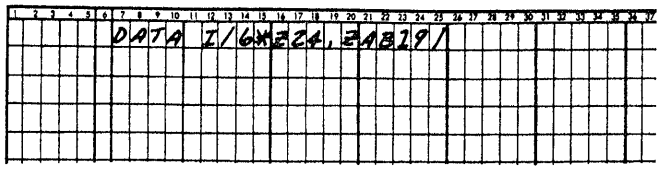
If A is a nine-element array for real variables and B is an array for real variables of three or more elements, all nine elements of A and the first three elements of B are initialized to 1.0.

An error condition occurs if all constants are not exhausted when the last variable or array has been satisfied. Similarly, an error occurs when a variable or array is specified for which no constants are available. Partial filling of an array is permitted, as shown in this example, assuming that B is an array containing more than three elements.

Hexadecimal Constants

Hexadecimal constants are written as the letter Z followed by one to four hexadecimal digits (0 through F). Each constant is assigned one 16-bit word and the constant is right-justified if three or less hexadecimal digits are used. Constants must be separated by commas.

Example 3:



The first 6 elements of array I are initialized to the following configuration:

0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
			0				0				2				4

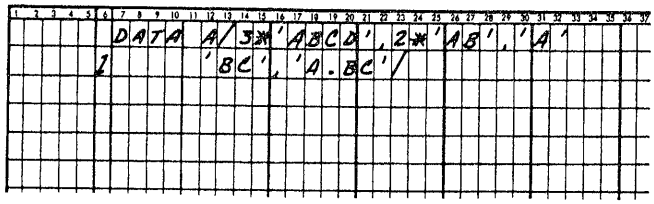
The seventh element is initialized to:

1	0	1	0	1	0	1	1	0	0	0	1	1	0	0	1
			A				B				1				9

Literal Data

Literal data must be enclosed by apostrophes. An apostrophe within a literal field is represented by two consecutive apostrophes. A literal constant may not exceed the element length of the variable or array to which it is assigned. Where necessary, blanks are included, with the constant left-justified.

Example 4:



If the array A contains at least seven elements, and is of standard (two-word) precision, the first three elements are assigned the value ABCD, the fourth and fifth the value AB␣␣ (␣ is a blank), the sixth element the value A'BC, and the seventh A.BC.

Data-Variable Combinations

variable \ data	real	integer	hexadecimal	literal
real	yes	no	no	yes
integer	no	yes	yes	yes

MANIPULATIVE INPUT/OUTPUT STATEMENTS

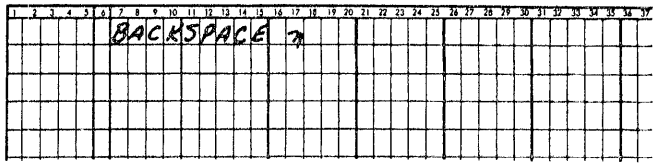
The statements BACKSPACE, REWIND, and END FILE are used to manipulate the FORTRAN data file on disk for unformatted I/O.

The purpose and use of the manipulative I/O statements in conjunction with the Unformatted I/O Area on disk is to allow the user to simulate the operation of a magnetic tape device. The Unformatted I/O Area, if desired, must be defined by the user by first defining a Fixed Area and then storing a data file with the name $$$$$ in the Fixed Area. This data file constitutes the Unformatted I/O Area. Only one Unformatted I/O Area is permitted during a given job.

BACKSPACE Statement

The BACKSPACE statement causes the file on unit n to be backspaced one logical record. This statement has no effect if the unit it refers to is positioned at its initial point (the first sector of the data file $$$$$, the equivalent of the load point on a magnetic tape).

General Form

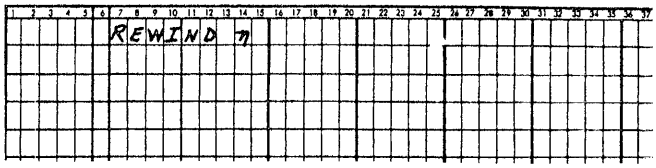


where n is an unsigned integer constant or integer variable specifying the unit number.

REWIND Statement

The REWIND statement causes the file on unit n to be repositioned to its initial point.

General Form

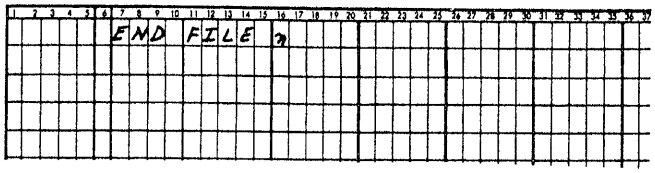


where n is an unsigned integer constant or integer variable specifying the unit number.

END FILE Statement

The END FILE statement causes an end-of-file record to be written on unit n.

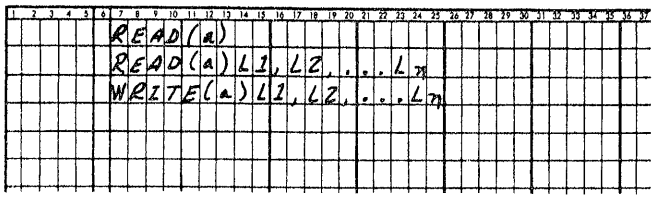
General Form



where n is an unsigned integer constant or integer variable specifying the unit number.

I/O WITHOUT DATA CONVERSION

The generalized READ and WRITE statements for I/O without data conversion appear as:



where

a is an unsigned integer constant or integer variable that specifies a logical unit number to be used for I/O data. The logical unit number is associated with the actual I/O unit when the core load is built.

L1 through Ln is a list of variable names, separated by commas, for the I/O data.

The form READ (a) L1, L2,...Ln is used to read a core image record, without data conversion, into core storage from unit a. No FORMAT statement is required; the amount of data that is read is determined by the number of list items and their attributes.

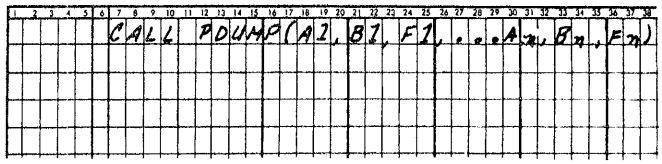
The total length of the list of variable names must not be longer than the logical record length. If the length of the list is equal to the logical record length, the entire record is read. If the length of the list is shorter than the logical record length, the unread items in the record are skipped.

The form READ (a) is used to skip an unedited record on unit a.

The form WRITE (a) L1, L2, ... Ln is used to write a core image record, without data conversion, on unit a.

DUMP

The dump program PDUMP can be called by a FORTRAN program or subprogram as follows:



where

A1 and B1 are variable data names, subscripted or nonsubscripted, indicating the inclusive limits of the first block of storage to be dumped,

An and Bn are variable data names, subscripted or nonsubscripted, indicating the inclusive limits of the nth block of storage to be dumped. Either An or Bn may represent upper or lower limits.

F1 through Fn are integers indicating the dump format desired in the first through nth blocks of storage to be dumped. Fn is assigned in the following manner:

0 = Hexadecimal
4 = Integer
5 = Real

A-CONVERSION

Spacing, tabulating, and shifting on the Console Printer can be controlled by outputting a unique value for the operation desired. These values must be assigned as integer constants and outputted through A-conversion.

The operations that can be performed and the unique values assigned to them are:

Operation	Value
Backspace	5696
Carrier Return	5440
Line Feed	9536
Shift to print black	5184
Shift to print red	13632
Space	16448
Tabulate	1344

As an example of Console Printer control, assume that a variable, X, is to be printed in the existing black ribbon shift and that another variable, Y, is to be printed in red following a tabulation. Following the printing of Y, the ribbon is to be shifted back to black. This can be accomplished as follows:

```
      I=1344
      J=13632
      K=5184
      WRITE(1,3)X,I,J,Y,K
    3 FORMAT(F12.6,2A1,F12.6,A1)
```

FORTRAN logical unit 1, as specified in the WRITE statement, is the Console Printer. The sequence of operations to be performed are: print the variable X, tabulate, shift to print red, print the variable Y, shift to print black.

Each control variable counts as one character and must be included in the count of the maximum line length.

T-FORMAT CODE

Input and output may begin at any position in a FORTRAN record by using the format code Tw where w is an unsigned integer constant specifying the position where the transfer of data is to begin.

When the output is to be printed on an 1132 or 1403 Printer, w is not equal to the print position at which printing begins. In this case, because of the carriage control character, the first print position is equal to w-1. For example, the following statements

```
      WRITE(3,5)
    5 FORMAT(T40,'1964 INVENTORY REP
    1 ORT',T80,'DECEMBER',T2,
    2 'PART NO. 10095')
```

result in a printed line as follows:

```
PART NO. 10095 1964 INVENTORY REPORT DECEMBER
↑              ↑                     ↑
print          print                 print
position 1     position 39           position 79
```

The following statements

```
      READ(2,5)
    5 FORMAT(T40,'HEADINGS')
```

cause the first 39 characters of the input data to be skipped; the next 9 characters then replace the blank (b) and the characters H, E, A, D, I, N, G, and S in storage.

The T-format code may be used in a FORMAT statement with any type of format code. For example, the following statement is valid:

```
    5 FORMAT(T100,F10.3,T50,#9.3,
    1 T1,'ANSWER IS')
```

The Core Load Builder constructs a core image program from a mainline object program.

The Core Load Builder is called by:

- The Supervisor. After the Supervisor has detected the XEQ monitor control record in the input stream and has read the Supervisor control records, if any, and written them in the Supervisor Control Record Area (SCRA) on disk, the Core Load Builder is called to construct a core image program if the mainline program named in the XEQ control record is in disk system format (see Core Load Construction, below). After the core image program has been built, the Core Load Builder transfers control to the Core Image Loader, which reads any portion of the core load that resides below location 4096_{10} into core from the CIB (the remainder of the core load has already been placed in core by the Core Load Builder). Control is then transferred to the core load itself.

- DUP. After DUP has detected the STORECI control record, it reads the Supervisor control records, if any, and writes them in the Supervisor Control Record Area (SCRA) on disk. Unless the program is already in Working Storage, DUP fetches the program, converts it to disk system format, if necessary, and stores it in Working Storage. Next, the Core Load Builder is fetched to construct the core image program (see Core Load Construction, below). After the core image program has been built, the Core Load Builder returns control to DUP to store the core image program in the User or Fixed Area.

- The Core Image Loader. When the Resident Monitor is entered at the LINK entry point, the Core Image Loader is called to accomplish the link-to-link transfer of control. The Core Image Loader determines the format of the link from the LET/FLET entry and, if the program to be executed is in disk system format, calls the Core Load Builder to construct the core image program (see Core Load Construction, below). After the core image program has been built, the Core Load Builder returns control to the Core Image Loader to fetch the core load and transfer control to it. If the link is in disk core image format, the Core Image Loader fetches it and transfers control to it, without calling any other monitor program.

CORE LOAD CONSTRUCTION

The following paragraphs describe the functions of the Core Load Builder in the construction of a core image program. These functions are not necessarily performed in this order.

Figure 10 (see Use of the Core Image Buffer and Working Storage, below) shows the core image program being built. Figure 6 (see "STORECI" under Disk Utility Programs) shows the core image program stored on disk. Figure 12 (see "Fetching a Link" under Core Image Loader) shows the core load ready for execution.

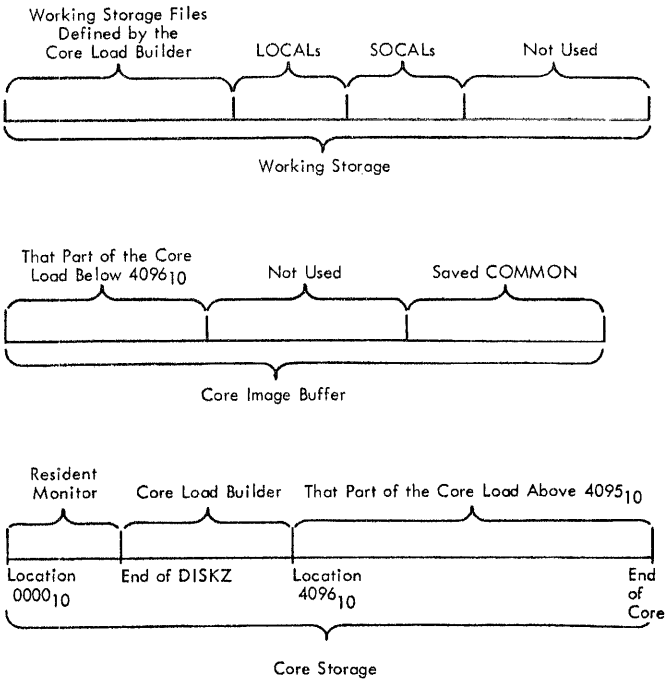


Figure 10. Distribution of a Core Image Program being Built

PROCESSING THE CONTENTS OF THE SCRA

The LOCAL, NOCAL, and FILES control records are read from the Supervisor Control Record Area (SCRA) on disk and analyzed. Tables are built from the information obtained from the respective control record types. These tables are used in later phases of the construction of the core image program (see below).

CONVERSION OF THE MAINLINE OBJECT PROGRAM

The mainline object program is converted from disk system format to disk core image format.

INCORPORATION OF SUBPROGRAMS

All the subprograms called by the mainline program and by other subprograms are included in the core load, except for (1) the disk I/O subroutine, (2) any LOCAL subprograms specified, and (3) SOCALS (see System Overlays, below), if they are employed.

If LOCALs have been specified or if SOCALS are employed by the Core Load Builder, the LOCAL/SOCAL flipper is included in the core load.

PROVISION FOR LOCALs AND SOCALs

If LOCALs have been specified, a LOCAL area as large as the largest LOCAL is reserved in the core load, into which the LOCAL subprograms are read by the LOCAL/SOCAL flipper. In addition, the subprograms specified on the LOCAL control records are written in Working Storage following any files defined in Working Storage. If the core load is executed immediately, each LOCAL is read from Working Storage into the LOCAL area by the LOCAL/SOCAL flipper as it is called. If the core load is stored in disk core image format before it is executed, the LOCALs are stored following the core load. During the execution the LOCAL/SOCAL flipper fetches them from the User/Fixed Area.

If SOCALs are employed by the Core Load Builder, a SOCAL area as large as the largest overlay is reserved in the core load, into which the SOCALs are read by the LOCAL/SOCAL flipper. In addition, the subprograms comprising the SOCALs are written in Working Storage following any files defined in Working Storage and any LOCALs stored there. If the core load is executed immediately, each SOCAL is read from Working Storage into the SOCAL area by the LOCAL/SOCAL flipper as it is called. If the core load is stored in disk core image format before it is executed, the SOCALs are stored following the core load and the LOCALs, if any. During the execution the LOCAL/SOCAL flipper fetches the SOCALs from the User/Fixed Area.

CONSTRUCTION OF THE CORE IMAGE HEADER

During the construction of the core image program, the Core Load Builder also constructs the core image header record, which contains the information required by the Core Image Loader to initialize the core load for execution. This record becomes a part of the core image program and at execution time resides in core along with the rest of the core load.

PROCESSING DEFINED FILES

The Core Load Builder uses the information in the FILES control record to equate files defined in the mainline program by the FORTRAN DEFINE FILE statement or by the Assembler FILE statement to data files on disk. The processing consists of comparing the file number in a 7-word DEFINE FILE Table entry with each of the file numbers from the FILE control records, which have been stored in the SCRA by the Supervisor or DUP. If a match occurs, the name of the disk area associated with the file number on the FILES control record is found in LET/FLET and the sector address of that disk area is placed in word 5 of the DEFINE FILE Table entry. If none of the file numbers from the FILES control records matches the number in the DEFINE FILE Table entry or if no name is specified on the FILES control record, the Core Load Builder assigns an area in Working Storage for the data file, and the sector address of the data file, relative to the start of Working Storage, is placed in word 5 of the DEFINE FILE Table entry. This procedure is repeated for each 7-word DEFINE FILE Table entry in the mainline program.

USE OF THE CORE IMAGE BUFFER (CIB) AND WORKING STORAGE

The Core Load Builder places in the CIB any parts of the core load which, when loaded, are to reside below location 4096_{10}. Any parts of the core load which are to reside above location 4095_{10} are placed directly into core storage.

Working Storage is used by the Core Load Builder to contain (1) any data files defined by a FILES control record and assigned by the Core Load Builder in Working Storage, (2) all the LOCAL subprograms specified, and (3) the SOCALs, if they are required. Figure 10 shows the distribution of a core image program being built between core storage, the CIB, and Working Storage.

ASSIGNMENT OF THE CORE LOAD ORIGIN

The Core Load Builder origins core loads built from relocatable mainline programs at the next higher-addressed word above the end of the disk I/O subroutine to be used by the core load:

Disk I/O Subroutine	Core Load Origin
DISKZ	451_{10}
DISK1	726_{10}
DISKN	926_{10}

The origins for core loads built from absolute mainline programs are not controlled by the Core Load Builder. Therefore, the user must origin absolute mainline programs above the end of the disk I/O subroutine to be used by the core load.

TRANSFER VECTOR

The transfer vector is a table included in each core load that provides the linkage to the subprograms. It is composed of the LIBF TV, the transfer vector for subprograms referenced by LIBF statements, and the CALL TV, the transfer vector for subprograms referenced by CALL statements.

Each CALL TV entry is a single word; each word contains the absolute address of an entry point in a subprogram included in the core load that is referenced by a CALL statement. In the case of a subprogram referenced by a CALL statement but specified as a LOCAL, the CALL TV entry contains the address of the special LOCAL linkage instead of the subprogram entry point address. If SOCALs are required, the CALL TV entries for function subprograms contain the address of the special SOCAL linkage instead of the subprogram entry point address.

Each LIBF TV entry consists of three words. Word 1 is the link word in which the return address is stored. Words 2 and 3 contain a branch to the subprogram entry point. In the case of a subprogram referenced by a LIBF statement but specified as a LOCAL, the LIBF TV entry for its entry point contains a branch to the special LOCAL linkage instead of to the subprogram entry point address. If SOCALs are required, the LIBF TV entry for a SOCAL subprogram contains a branch to a special entry in the LIBF TV for the SOCAL of which the subprogram is a part. This special entry provides the linkage to the desired SOCAL subprogram.

SYSTEM OVERLAYS

SOCALs (System-Overlays-to-be-loaded-on-call) are subprogram groups (by type and subtype) which are made into overlays by the Core Load Builder. They may make it possible for a FORTRAN core load which would otherwise not fit into core to be loaded and executed. If, in constructing a core image program from a FORTRAN mainline program, the Core Load Builder determines that the core load will not fit into core, SOCALs are created by the Core Load Builder for the core load. In addition, the LOCAL/SOCAL flipper, which fetches the SOCALs when they are required at execution time, is included in the core load along with the area into which the SOCALs are loaded (the SOCAL area).

The SOCALs are created by subprogram type and subtype (see the description of program type and subtype under "Disk System Format" in Appendix C. Formats). The following table describes the SOCALs:

Subprogram Class	Type	Subtype	Overlay
Arithmetic	3	2	1
Function	4	8	1
Non-disk FORTRAN I/O and "Z" conversion subroutines	3	3	2
"Z" device subroutines	5	3	2
Disk FORTRAN I/O	3	1	3

There are two SOCAL options. The Core Load Builder first attempts to make the core load fit into core by using overlays 1 and 2 only (option 1). If the core load still will not fit into core, overlays 1, 2, and 3 are used (option 2). If use of option 2 still does not make it possible for the core load to fit into core, an error condition is indicated.

Option 1 reduces the core requirement of the core load by an amount equal to the size of the smaller of the two overlays used minus approximately 15 additional words required for the special SOCAL linkage. Option 2 reduces the core requirement by an amount equal to the sum of the sizes of the two smallest overlays minus approximately 15 additional words required for the special SOCAL linkage.

Each SOCAL does not contain all the available subprograms of the specified type(s) and subtype(s); only those subprograms of the specified type(s) and subtype(s) required by the core load are contained in the SOCAL.

If a subprogram which would otherwise be included in a SOCAL is specified as a LOCAL subprogram, that subprogram is made a LOCAL and is not included in the SOCAL in which it would ordinarily be found.

SOCALs are never built for core loads in which the mainline program is written in Assembler language.

LOCAL/SOCAL FLIPPER

The LOCAL/SOCAL flipper is included in each core load in which LOCAL subprograms have been specified or in which SOCALs have been employed. If execution of the core load immediately follows the building of the core image program, this subroutine reads a LOCAL/SOCAL from Working Storage into the LOCAL/SOCAL area as it is called at execution time. If the core image program was stored in the User or Fixed Area in disk core image format prior to execution, the flipper reads a LOCAL/SOCAL at execution time from the User or Fixed Area, where it was stored following the core load, into the LOCAL/SOCAL area.

The flipper is entered via the special LOCAL/SOCAL linkage. A check is made to determine if the required LOCAL/SOCAL is already in core. If it is not in core, the flipper reads the required LOCAL/SOCAL into the LOCAL/SOCAL area from the disk, modifies the special LOCAL/SOCAL linkage, and transfers the LOCAL/SOCAL subprogram via the special linkage.

CORE IMAGE LOADER

The Core Image Loader serves both as a loader for core loads and as an interface for some parts of the monitor system.

On any entry to the Skeleton Supervisor, the Core Image Loader is fetched and control is transferred to it. The Core Image Loader determines where the Skeleton Supervisor was entered, i.e., at LINK, DUMP, or EXIT.

FETCHING THE SUPERVISOR

If an entry was made to the Skeleton Supervisor at the EXIT entry point, the Core Image Loader first restores to core the disk I/O subroutine used by the monitor programs (DISKZ), if it is not already in core. It then fetches and transfers control to the Monitor Control Record Analyzer to read monitor control records from the input stream.

If an entry was made to the Skeleton Supervisor at the DUMP entry point, the Core Image Loader first saves the contents of the core below location 4096_{10} in the CIB and fetches and transfers control to the DUMP program to perform the core dump according to the parameters specified. At the completion of the dump, the DUMP program either restores the contents of core below location 4096_{10} from the CIB and transfers control back to the core load or terminates the execution with a CALL EXIT (see "Terminal And Dynamic Dumps" under Supervisor, above).

FETCHING A LINK

If an entry was made to the Skeleton Supervisor at the LINK entry point, the Core Image Loader first saves low COMMON (locations 1216_{10}-1535_{10} if DISK1 or DISKN is in core or locations 896_{10}-1215_{10} if DISKZ is in core). It then determines from COMMA the lowest-addressed word of COMMON, if any, defined by the core load just executed. Any COMMON below location 4096_{10} is saved in the CIB by the Core Image Loader.

Figure 11 illustrates the scheme used in saving COMMON between links.

The LET/FLET entry for the link to be fetched is then located and the Core Image Loader determines from it whether the link is in disk core image format or disk system format. If the link is in disk core image format, the Core Image Loader fetches the disk I/O subroutine required by the core load, if it is not already in core. Next, the Core Image Loader restores low COMMON if it lies within the COMMON defined by the core load just executed. The core load is then fetched and control is transferred to it.

If the link is in disk system format, the Core Image Loader calls the Core Load Builder to construct a core image program from the mainline program. When the core image program has been built, the Core Load Builder returns to the Core Image Loader, which then fetches the core load, as described above, and transfers control to it.

Figure 12 shows the layout of a core load loaded into core, ready for execution.

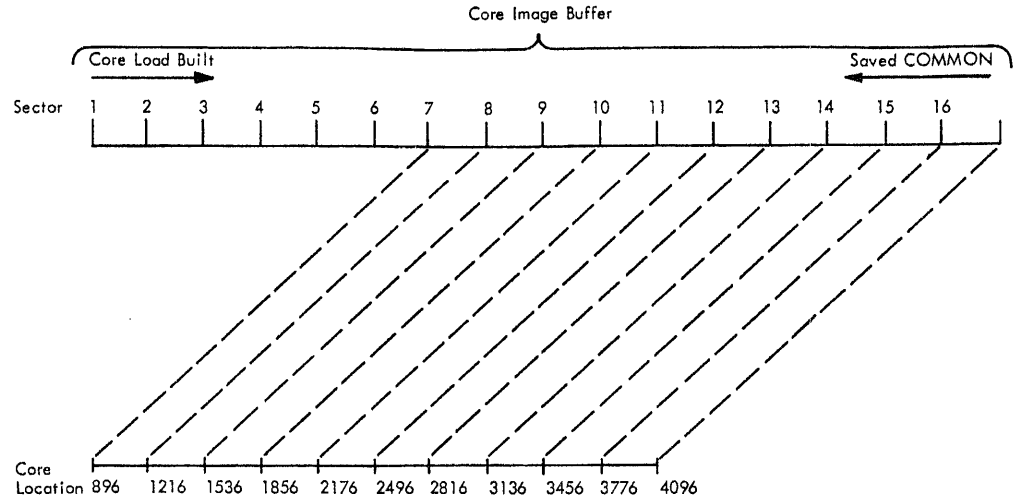


Figure 11. Scheme for Saving COMMON between Links

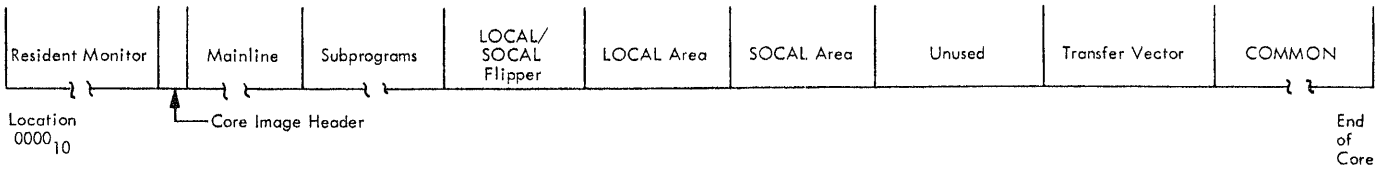


Figure 12. Layout of a Core Load Loaded for Execution

The System Library is a group of disk-resident subprograms and mainline programs which perform I/O, conversion, arithmetic, and disk initialization and maintenance functions.

This publication describes only the System Library programs found in the 1130 Disk Monitor System, Version 2. Programs carried over from Version 1 are described in the Subroutine Library publication (see preface). Introductory material on each specific type of subprogram is found in that publication.

INTERRUPT SERVICE SUBROUTINES

The interrupt service subroutines (ISSs) manipulate the I/O devices attached to the 1130 Computing System, handling all programming details peculiar to each device.

A complete description of ISS characteristics is found in the Subroutine Library publication. Table 6 lists the ISSs used by the monitor system.

Table 6. Monitor System ISS Names

Device	Subroutine
1442 Card Read Punch	CARD0 or CARD1
2501 Card Reader	READ0 or READ1
1442 Card Punch	PNCH0 or PNCH1
Disk	DISK1 or DISKN
1132 Printer	PRNT1
1403 Printer	PRNT3
Keyboard/Console Printer	TYPE0
Console Printer	WRTY0
1134/1055 Paper Tape Reader Punch	PAPT1 or PAPTN
1627 Plotter	PLOT1
1231 Optical Mark Page Reader	OMPR1

2501 CARD READER SUBROUTINES (READ0 and READ1)

These card subroutines perform read and test functions relative to the IBM 2501 Card Reader.

READ0 is shorter than READ1, provides no error parameter, and is the standard subroutine for operation of the 2501 Card Reader.

READ0 can be used if the error parameter is not needed. On an error, the subroutine branches to an error trap in the Skeleton Supervisor, waiting for operator intervention. Last card conditions cause an exit to the Preoperative Error Trap (see Supervisor).

READ1 can be used for operation of the 2501 Card Reader if a user error exit is desired rather than the trapping of READ0.

Calling Sequence

```
         LIBF    READa     CALL CARD INPUT
         DC      /baae     CONTROL PARAMETER
         DC      IOAR      I/O AREA PARAMETER
         DC      ERROR     ERROR PARAMETER
         .
         .
         .
ERROR    DC      *-*       RETURN ADDRESS
         .
         .
         .
         BSC     I         ERROR RETURN TO CALLER
         .
         .
         .
IOAR     DC      f         WORD COUNT
         BSS     h         I/O AREA
```

where

a is 0 or 1,

b is the I/O function digit,

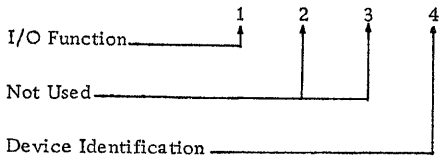
e is the device identification digit,

f is the number of columns to be read from the card,

h is the length of the I/O area. h must be equal to or greater than f.

Control Parameter

This parameter consists of four hexadecimal digits as shown below:



I/O Function

The I/O function digit specifies a particular operation to be performed on the 2501 Card Reader. The functions, associated digital values, and required parameters are listed and described below.

Function	Digital Value	Required Parameters*
Test	0	Control
Read	1	Control, I/O Area, Error**

*Any parameter not required for a particular function must be omitted.
**The error parameter is not required for READ0.

Test. Branches to LIBF+2 if the previous operation has not been completed, to LIBF+3 otherwise.

Read. Reads one card and transfers a specified number of columns of data to the user's input area. The number of columns read (1-80) is specified by the user in the first location of the input area. The subroutine initiates the Read function and returns control to the user's program.

When an operation complete interrupt occurs, the card subroutine checks for errors. If an error occurred, READ0 exits to an error trap; READ1 informs the user of the error and sets up to terminate or retry the operation.

The data in the user's input area is in IBM Card Code; that is, each 12-bit column image is left-justified in one 16-bit word.

There is no separate Feed function. However, a Feed can be obtained by a Read function with a word count of zero.

Device Identification

This digit must be zero.

I/O Area Parameter

The I/O area parameter is the label on the control word that precedes the user's input area. The control word consists of a word count that specifies the number of columns of data to be read, always starting with column 1.

Error Parameter

READ0. READ0 has no error parameter. If an error is detected while an operation complete interrupt is being processed, the subroutine branches to an error trap in the Skeleton Supervisor, with interrupt level 4 on, waiting for operator intervention. When the condition has been corrected and the 2501 made ready, the subroutine attempts the operation again.

READ1. READ1 has an error parameter. If an error is detected, the user can request the subroutine to terminate (that is, to clear the subroutine's busy indicator and turn off the interrupt level) or to branch to an error trap in the Skeleton Supervisor, with interrupt level 4 on, waiting for operator intervention.

Last Card

A Read function requested after the last card has been detected causes the last card to be ejected and a branch to be taken to the Preoperative Error Trap.

CARD PUNCH SUBROUTINES (PNCH0 and PNCH1)

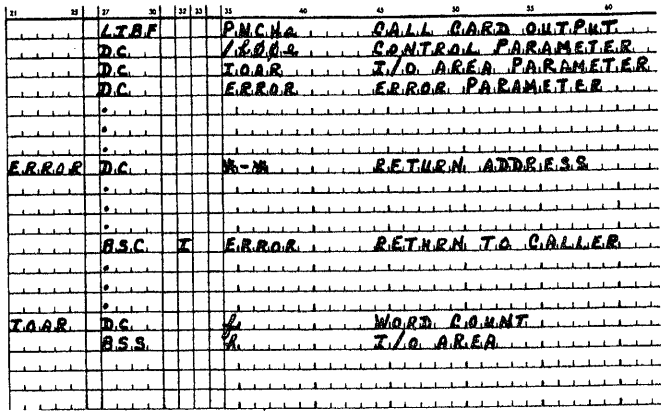
These card subroutines perform all I/O functions relative to the IBM 1442-5 Card Punch, that is, Punch and Feed. These subroutines may also be used with the 1442-6 or 1442-7 Card Read Punch for Punch and Feed functions.

The PNCH0 subroutine is shorter than PNCH1, provides no error parameter, and is the standard subroutine for operation of the 1442 Card Punch.

PNCH0 can be used if the error parameter is not needed. On an error, the subroutine branches to an error trap in the Skeleton Supervisor, waiting for operator intervention. Last card conditions cause an exit to the Preoperative Error Trap.

PNCH1 can be used for operation of the 1442 Card Punch if a user error exit is desired rather than the trapping of PNCH0.

Calling Sequence



where

a is 0 or 1,

b is the I/O function digit,

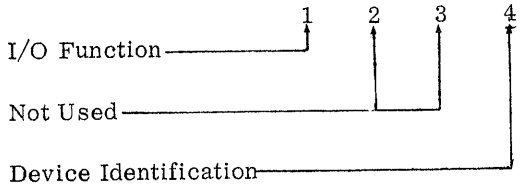
e is the device identification digit,

f is the number of columns to be punched into the card,

h is the length of the I/O area. h must be equal to or greater than f.

Control Parameter

This parameter consists of four hexadecimal digits as shown below:



I/O Function

The I/O function digit specifies a particular operation to be performed on the 1442 Card Punch. The functions, associated digital values, and required parameters are listed and described below.

Function	Digital Value	Required Parameters*
Test	0	Control
Punch	2	Control I/O Area, Error**
Feed	3	Control, Error**

*Any parameter not required for a particular function must be omitted.

**The error parameter is not required for PNCH0.

Test. Branches to LIBF+2 if the previous operation has not been completed, to LIBF+3 otherwise.

Punch. Punches into one card the number of columns of data specified by the word count found at the beginning of the user's output area. As each column comes under the punch dies, a column interrupt occurs, the subroutine transfers a word from the user's output area to the punch, and then returns control to the user's program. The character punched is the image of the leftmost 12 bits in the word.

This sequence is repeated until the requested number of columns has been punched, after which an operation complete interrupt occurs. At this time the card subroutine checks for errors. If an error occurred, PNCH0 exits to an error trap; PNCH1 informs the user of the error and sets up to terminate or retry the operation.

Feed. Initiates a card feed cycle. This function advances all cards in the machine to the next station; that is, a card at the punch station advances to the stacker, a card at the read station advances to the punch station, and a card in the hopper advances to the read station. No data is punched as a result of a Feed function and no column interrupts occur.

When the card advance is complete, an operation complete interrupt occurs. At this time the card subroutine checks for errors. If an error occurred, PNCH0 exits to an error trap; PNCH1 informs the user of the error and sets up to terminate or retry the operation.

Device Identification

This digit must be zero.

I/O Area Parameter

The I/O area parameter is the label of the control word that precedes the user's output area. The control word consists of a word count that specifies the number of columns of data to be punched, always starting with column 1.

Error Parameter

PNCH0. PNCH0 has no error parameter. If an error is detected while an operation complete interrupt is being processed, the subroutine branches to an error trap in the Skeleton Supervisor, with interrupt level 4 on, waiting for operator intervention. When the condition has been corrected and the 1442 made ready, the subroutine attempts the operation again.

PNCH1. PNCH1 has an error parameter. If an error is detected, the user can request the subroutine to terminate (that is, to clear the subroutine's busy indicator and turn off the interrupt level) or to branch to an error trap in the Skeleton Supervisor, with interrupt level 4 on, waiting for operator intervention.

DISK I/O SUBROUTINES

All disk subroutines (including DISKZ) reside in the System Area separate from the other programs in the System Library because they must be processed differently than other I/O subroutines, since the monitor system always requires a disk I/O subroutine. They are fetched by the Core Image Loader just prior to execution.

DISKZ is intended for use in a FORTRAN environment in which FORTRAN I/O is used. DISKZ makes no preoperative parameter checks and offers no file protection. It is the shortest of the three disk I/O subroutines and requires a special calling sequence. (See Subroutines Used by FORTRAN, below.)

DISK1 is intended for use by Assembler language programs in which the core storage requirement is of more importance than execution time. DISK1 is longer than DISKZ but is the shorter of the two subroutines intended for use in Assembler language programs (DISK1 and DISKN). However, DISK1 does not minimize extra disk revolutions when transferring more than 320 words nor does it provide a Write Immediate function.

DISKN minimizes extra disk revolutions in transferring more than 320 words. It provides all the functions provided by DISK1 as well as the ability to operate all five drives simultaneously. DISKN also provides the Write Immediate function.

One of the major differences among the disk subroutines is the ability to read or write consecutive sectors on the disk without taking extra revolutions. If full sectors are written, the time in which the I/O command must be given varies. DISKN is programmed so that transfers of more than 320 words are made with a minimum number of extra revolutions occurring between sectors.

Both DISK1 and DISKN have the same error handling procedures.

NOTE: In the monitor system, the disk I/O subroutines are not stored in the System Library; consequently, they do not have LET entries.

Sector Numbering and File Protection

In the interest of providing disk features permitting versatile and orderly control of disk operations, three important conventions have been adopted. They are concerned with sector numbering, file protection, and defective sector handling. Successful use of the disk I/O subroutines can be expected only if user programs are built within the framework of these conventions. The primary concern behind the conventions is the safety of data recorded on the disk. To this end, the file protection scheme plays a major role, but does so in a manner that is dependent upon the sector-numbering technique. The latter contributes to data safety by allowing the disk I/O subroutine to verify the correct positioning of the access arm before it actually performs an operation. This verification requires that sector identifications be prerecorded on each sector and that subsequent writing on the disk be done in a manner that preserves the existing identification. The disk I/O subroutines support these requirements.

Sector Numbering

Each disk sector is assigned an address from the sequence 0, 1, ... 1599, corresponding to the sector position in the ascending sequence of cylinder and sector numbers from cylinder 0, sector 0 (outermost), through cylinder 199, sector 7 (innermost).

Each cylinder contains eight sectors and each sector contains 321 words, counting the sector address. The sector address is recorded in the first word of each sector and occupies the rightmost eleven bit positions. Of these eleven positions, the three low-order positions identify the sector (0-7) within the cylinder. Utilization of this first word for identification purposes reduces the per sector availability of data words to 320; therefore, transmission of full sectors of data is performed in increments of 320 words.

Sector addresses must be initially recorded on the disk by the user (via DISC or DCIP) and are thereafter rewritten by the disk I/O subroutines as each sector is written.

NOTE: Although not actually written on the disk, the logical drive code must be part of the sector address parameter (bits 0-3) which is stored in the second word of the I/O area.

File Protection

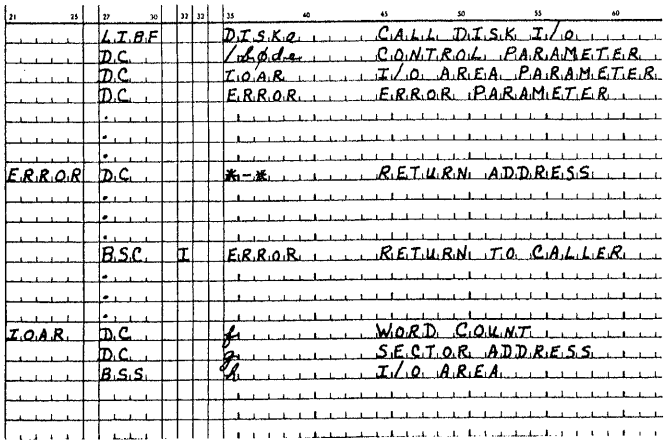
File protection is provided to prohibit the inadvertent destruction of previously recorded data. This control is achieved by having all Write functions (except Write Immediate) test for the file-protection status of sectors they are about to write.

Each cartridge has a file-protection address in COMMA. This address is the address of the first unprotected sector, i.e., the address of the beginning of Working Storage. Every sector, from sector 0 up to the sector address maintained in COMMA, is file-protected. The initial assignment of the file-protection address is done by the disk initialization program (see "DCIP" in Appendix A, Utility Programs). Subsequent updating of the file-protection address is performed by the monitor programs.

Defective Sector Handling

A defective sector is a sector on which a Write function cannot be successfully completed during initialization of the cartridge. A cylinder having one or more defective sectors is defined as a defective cylinder. The disk I/O subroutines can accommodate as many as three defective cylinders per cartridge. Since there are 203 cylinders on each disk, the disk I/O subroutines can "overflow" the 200 cylinders normally used when defective cylinders are encountered (see Effective Address Calculation).

Calling Sequence



where

a is 1 or N. Note that LIBF DISK0 is equivalent to LIBF DISK1.

b is the I/O function digit,

d is the Seek option digit,

e is the Displacement option digit,

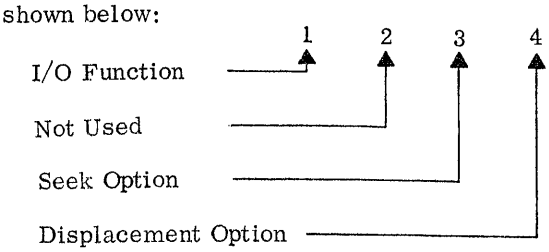
f is the number of words to be transferred to or from the disk,

g is the sector address at which the transfer is to begin,

h is the length of the I/O area. h must be equal to or greater than f.

Control Parameter

This parameter consists of four hexadecimal digits, shown below:



I/O Function

The I/O function digit specifies the operation to be performed on disk storage. The functions, their associated digital value, and the required parameters are listed and described below.

Function	Digital Value	Required Parameters*
Test	0	Control, I/O Area
Read	1	Control, I/O Area, Error
Write without RBC	2	Control, I/O Area, Error
Write with RBC	3	Control, I/O Area, Error
Write Immediate**	4	Control, I/O Area
Seek	5	Control, I/O Area, Error

*Any parameter not required for a particular function must be omitted.

**The Write Immediate function is valid only for DISKN.

Test. Branches to LIBF+3 if the previous operation on the drive has not been completed, to LIBF+4 otherwise.

NOTE: This function requires the I/O area parameter even though it is not used.

Read. Positions the access arm and reads data into the user's I/O area until the specified number of words

has been transmitted. Although sector identification words are read and checked for agreement with expected values, they are neither transmitted to the I/O area nor are they counted in the tally of words transferred.

If, during the reading of a sector, a read check occurs, up to 16 retries are attempted. If the error persists, the function is temporarily discontinued, an error code is placed in the accumulator, the address of the faulty sector is placed in the extension, and an exit is made to the error subroutine specified by the error parameter.

Upon return from the error subroutine, the operation is either reinitiated or terminated, depending on whether the accumulator is non-zero or zero, respectively.

Write With Readback Check. First checks whether or not the specified sector address is in a file-protected area. If it is, the subroutine places the appropriate error code in the accumulator and exits to the Preoperative Error Trap.

If the specified sector address is not in a file-protected area, the subroutine positions the access arm and writes the contents of the indicated I/O area onto the disk. Writing begins at the designated sector and continues until the specified number of words have been transmitted. A readback check is performed on the data written.

If a partial sector (less than 320 words) is written, the remaining words of the sector are automatically set to zero.

If any errors are detected, the operation is retried up to 16 times. If the function cannot be accomplished, an appropriate error code is placed in the accumulator, the address of the faulty sector is placed in the extension, and an exit is made to the error subroutine designated by the error parameter.

Upon return from this error subroutine, the operation is either reinitiated or terminated, depending upon whether the accumulator is non-zero or zero, respectively.

As each sector is written, the subroutine supplies the sector identification word. The identification word for the first sector is obtained from the I/O area, although it and subsequently generated identification words are not included in the word count.

Write Without Readback Check. Performs the same as the Write With Readback Check function described above except that no readback check is performed.

Write Immediate (DISKN only). Writes data with no attempt to position the access arm, check for file-protect status, or check for errors. Writing begins at the sector number specified in the user's I/O area. This function provides more rapid writing to the disk than is provided in the previously described Write functions; it provides, for example, the ability to "stream" data to the disk for temporary bulk storage or to write addresses in Working Storage (see "DWADR" under Disk Utility Program).

If a partial sector (less than 320 words) is written, the remaining words of the sector are automatically set to zero.

As each sector is written, the subroutine supplies the sector identification word. The identification word for the first sector is obtained from the I/O area, although it and subsequently generated identification words are not included in the word count.

Seek. Initiates a seek as specified by the seek option digit. If any errors are detected, the operation is retried up to 16 times.

The Seek function requires that the user set up the normal I/O area parameters (see I/O Area Parameter) even though only the sector address in the I/O area is used.

Seek Option

If digit 3 of the control parameter is zero, a Seek is executed to the cylinder whose sector address is in the I/O area; if non-zero, a seek is executed to the next non-defective cylinder toward the center, regardless of the sector address in the I/O area. This seek to the next non-defective cylinder must be taken into consideration when planning for the "streaming" of data.

This option is valid only when the Seek function is specified.

Displacement Option

If digit 4 of the control parameter is zero, the sector address word contains the absolute sector identification; if non-zero, the file-protection address for the specified cartridge is added to bits 4-15 of the sector address word to generate the effective sector identification. The file-protection address is the sector identification of the first unprotected sector.

I/O Area Parameter

The I/O area parameter is the label of the first of two control words which precede the user's I/O area. The first word contains the number of data words that are to be transferred during the disk operation. This number need not be limited by sector or cylinder size, since the subroutines cross sector and cylinder

boundaries, if necessary, in order to transmit the specified number of words.

The second word contains the sector address at which reading or writing is to begin. Bits 0-3 are the device identification (drive code) and must be 0, 1, 2, 3, or 4. Bits 4-15 specify the sector address. The user's I/O area follows the two control words.

Error Parameter

If an error is detected, the user can request the subroutine to terminate (that is, to clear the subroutine's busy indicator and turn off interrupt level 2) or to branch to an error trap in the Skeleton Supervisor with interrupt level 2 on, waiting for operator intervention.

Effective Address Calculation

Effective address calculation is as follows:

1. Obtain the sector address found in the sector address word of the I/O area.

   NOTE: This address causes a preoperative error exit to the Preoperative Error Trap if it exceeds 1599.

2. If the displacement option digit in the control parameter is non-zero, add the sector address of the first sector that is not file-protected.
3. If the resultant address is equal to or greater than the sector address of the first defective cylinder, add 8.
4. If the resultant address is equal to or greater than that of the second defective cylinder, add 8.
5. If the resultant address is equal to or greater than that of the third defective cylinder, add 8.

The address obtained from steps 1-5 is the effective sector address.

Disk Initialization

Before the monitor system is stored on a cartridge, the Disk Cartridge Initialization Program (DCIP) must be executed. The functions of this program are to write sector addresses on a disk cartridge, to detect any defective cylinders, to store defective cylinder information and a cartridge ID in sector 0 of cylinder 0 and to initialize DCOM. (See Appendix A. Utility Programs.)

1403 PRINTER SUBROUTINE (PRNT3)

The printer subroutine PRNT3 handles all print and carriage control functions relative to the 1403 Printer. Only one line of data can be printed and/or one carriage operation executed with each call to the printer subroutine.

Calling Sequence

```
         LIBF    PRNT3     CALL PRINTER OUTPUT
         D.C.    /bcdØ     CONTROL PARAMETER
         D.C.    IOAR      I/O AREA PARAMETER
         D.C.    ERROR     ERROR PARAMETER
ERROR    D.C.    *-*       RETURN ADDRESS
         BSC     I         ERROR RETURN TO CALLER
IOAR     D.C.    f         WORD COUNT
         BSS     h         I/O AREA
```

where

b is the I/O function digit,

c is the "immediate" carriage operation digit,

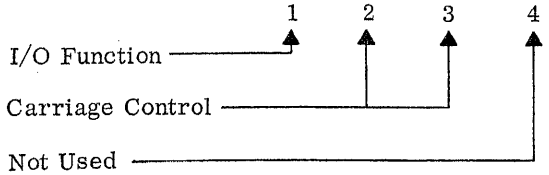
d is the "after-print" carriage operation digit,

f is the number of words to be printed on the 1403 Printer,

h is the length of the I/O area. h must be equal to or greater than f.

Control Parameter

This parameter consists of four hexadecimal digits which are used as shown below.



I/O Function

The I/O function digit specifies the operation to be performed on the 1403 Printer. The functions, their associated digital values, and the required parameters are listed and described below.

Function	Digital Value	Required Parameters*
Test	0	Control
Print	2	Control, I/O Area, Error
Control Carriage	3	Control

*Any parameter not required for a particular function must be omitted.

Test. Branches to LIBF+2 if the previous operation has not been completed, to LIBF+3 otherwise.

Print. Prints characters from the user's I/O area, checking for channel 9 and 12 indications. If either of these conditions is detected, the subroutine branches to the user's error subroutine after the line of data has been printed. Upon return from this error subroutine, a skip to channel 1 is initiated or the function is terminated, depending upon whether the accumulator is non-zero or zero, respectively.

Control Carriage. Controls the carriage as specified by the carriage control digits listed in Table 7.

Carriage Control

Digits 2 and 3 specify the carriage control functions listed in Table 7. An "immediate" request is executed before the next print operation; an "after-print" request is executed after the next print operation and replaces the normal space operation.

If the function is Print, only digit 3 is examined; if the function is Control, and digits 2 and 3 both specify carriage operations, only digit 2 is used.

NOTE: An "after-print" request is lost if it is followed by an "immediate" request. If a series of "after-print" requests is given, only the last one is executed.

I/O Area Parameter

The I/O area parameter is the label of the control word that precedes the user's I/O area. The control word consists of a word count that specifies the number of words of data to be printed. The data must be in 1403 Printer code, packed two characters per word.

Error Parameter

See Calling Sequence.

Table 7. Carriage Control Operations

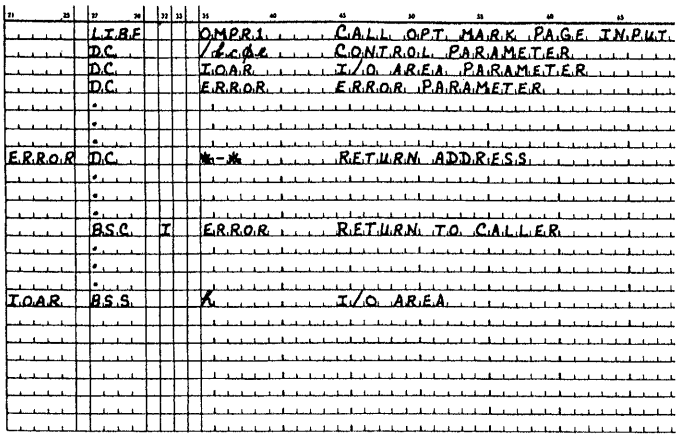
Digit #2: Immediate Carriage Operations
Print Functions Not Used Control Function 1 - Immediate Skip To Channel 1 2 - Immediate Skip To Channel 2 3 - Immediate Skip To Channel 3 4 - Immediate Skip To Channel 4 5 - Immediate Skip To Channel 5 6 - Immediate Skip To Channel 6 7 - Immediate Skip To Channel 7 8 - Immediate Skip To Channel 8 9 - Immediate Skip To Channel 9 A - Immediate Skip To Channel 10 B - Immediate Skip To Channel 11 C - Immediate Skip To Channel 12 D - Immediate Space Of 1 E - Immediate Space Of 2 F - Immediate Space Of 3
Digit #3: After-Print Carriage Operations
Print Functions 0 - Space One Line After Printing 1 - Suppress Space After Printing Control Function 1 - Skip After Print To Channel 1 2 - Skip After Print To Channel 2 3 - Skip After Print To Channel 3 4 - Skip After Print To Channel 4 5 - Skip After Print To Channel 5 6 - Skip After Print To Channel 6 7 - Skip After Print To Channel 7 8 - Skip After Print To Channel 8 9 - Skip After Print To Channel 9 A - Skip After Print To Channel 10 B - Skip After Print To Channel 11 C - Skip After Print To Channel 12 D - Space 1 After Print E - Space 2 After Print F - Space 3 After Print

1231 OPTICAL MARK PAGE READER SUBROUTINE (OMPR1)

The Optical Mark Page Reader subroutine OMPR1 handles the reading of paper documents eight and one-half inches wide by eleven inches deep through the 1231 Optical Mark Page Reader. A maximum of 100 words from one page can be read with one call to the subroutine.

When called to perform a Read function, OMPR1 performs a Feed function and reads a page into core storage according to the Master Control Sheet (see the publication IBM 1231, 1232 Optical Mark Page Readers, Form A21-9012), and the setting of the switches on the reader. Other functions performed are Feed, Stacker Select, and Disconnect.

Calling Sequence



where

b is the I/O function digit,

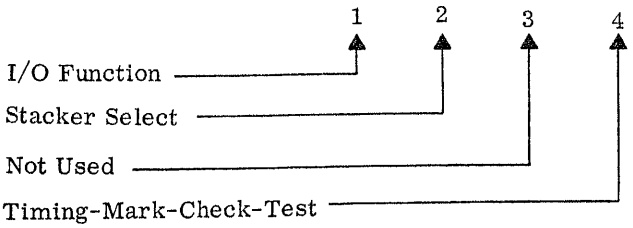
c is the stacker select digit,

e is the timing-mark-check test digit,

h is the length of the I/O area. h must be equal to or greater than the number of words designated to be read on the Master Control Sheet.

Control Parameter

This parameter consists of four hexadecimal digits as shown below.



I/O Function

The I/O function digit specifies the operation to be performed on the 1231 reader. The functions, their associated digital values, and the required parameters are:

Function	Digital Value	Required Parameters*
Test	0	Control
Read	1	Control, I/O Area, Error
Feed	3	Control
Disconnect	4	Control

*Any parameter not required for a particular function must be omitted.

Test. Branches to LIBF+2 if the previous operation has not been completed, to LIBF+3 otherwise.

The operation to be tested is specified by the fourth digit of the control parameter. A zero value in digit 4 specifies a normal device-busy test; that is, a test to determine if there is an operation in progress for which no operation complete interrupt has occurred. The subroutine is "not busy" once the operation complete interrupt takes place. A value of one for digit 4 specifies a Timing-Mark-Check-Busy test. This test indicates a "busy" condition as long as the Test-Timing-Mark-Check indication in the Device Status Word is on. If the user wishes to run with the Timing Mark Switch set on, it is recommended that digit 4 be set to one when performing a Test function.

Read. Reads words or segments from a document page into core storage starting at the I/O area address. It is not necessary for the user to perform a Feed function prior to a Read. In the absence of a Feed, the Read feeds the document before reading. When a Read function follows a Feed, the Read begins with the document started by the Feed. The number of bits per word read and the number of words per document read depends upon the way in which the Master Control Sheet is programmed (see the publication IBM 1231, 1232 Optical Mark Page Readers, Form A21-9012). OMPR1 reads a maximum of 100 words. Any word not programmed to be read (mark positions 8 or 18 not penciled on the Master Control Sheet) is skipped. Digit 2 of the control parameter specifies whether or not the document being read is to be stacker-selected. If digit 2 is set to one, the document is stacker-selected; if digit 2 is set to zero, it is not.

Feed. Initiates a feed cycle. This function advances a document from the hopper through the read station. A Read function following a Feed causes this document to be read. If a Feed function is followed by another Feed function without an intervening Read function, the document corresponding to the first Feed is disconnected; that is, it is passed through the 1231 without being read.

Disconnect. Terminates the Read function on the sheet currently being read. The subroutine's busy indicator is cleared.

I/O Area Parameter

The I/O area parameter is the label of the user's I/O area.

Error Parameter

There is an error parameter for the Read function only. Exits are made to the user's error subroutine when the following conditions are detected:

Master Control Sheet Error
Timing Mark Error
Read Error
Hopper Empty
Not Ready

(See Calling Sequence.)

SUBROUTINES USED BY FORTRAN

Many of the I/O and conversion subroutines cannot be specified in FORTRAN. Therefore, the System Library includes a set of limited-function I/O and conversion subroutines for FORTRAN programs. Any Assembler language I/O subroutines used by FORTRAN programs must employ these special subroutines for any I/O device specified in a mainline IOCS control record.

Of all the FORTRAN device subroutines, only DISKZ, PRNZ, and PLOTX return control to the caller after initiating an operation.

The System Library contains the following subroutines for FORTRAN programs:

CARDZ – 1442 Input/Output Subroutine
PNCHZ – 1442 Output Subroutine
READZ – 2501 Input Subroutine
TYPEZ – Keyboard Subroutine
WRTYZ – Console Printer Subroutine
PRNTZ – 1132 Printer Subroutine
PRNZ – 1403 Printer Subroutine
PAPTZ – Paper Tape Input/Output Subroutine
PLOTX – 1627 Plotter Subroutine
DISKZ – Disk Input/Output Subroutine
HOLEZ – IBM Card Code/EBCDIC Conversion Subroutine
EBCTB – EBCDIC/Console Printer Code Table
HOLTB – IBM Card Code Table
GETAD – Subroutine to Locate Start Address of EBCTB/HOLTB

GENERAL SPECIFICATIONS (EXCEPT DISKZ)

The "Z" device subroutines are ISS subroutines. They use a 121-word input/output buffer, contained in the non-disk FORTRAN I/O subroutine SFIO. The maximum amount of data transferable is listed in the description of each subroutine. Output data must be stored in unpacked (one character per word) EBCDIC format. Input data is converted to unpacked EBCDIC format.

The EBCDIC character set recognized by the subroutines is composed of the digits 0-9, alphabetic characters A-Z, blank, and special characters - + . & = ( ) , ' / * < % # @ . Any other character is recognized as a blank.

The accumulator, extension, and index registers 1 and 2 are used by the FORTRAN device subroutines and must be saved, if required, before entry into the subroutines. The accumulator must be set to zero for input operations.

For output operations, the accumulator must be set to 0002_{16}, except for PRNZ and PRNTZ, in which output is the only valid operation. Index register 2 set to the number of characters to be transferred, except for PRNZ and PRNTZ; for these two subroutines, index register 2 contains the number of characters to be printed plus an additional character

for carriage control. Index register 1 contains the starting address of the input buffer.

DESCRIPTIONS OF I/O SUBROUTINES

The following subroutines do not provide a check to determine the validity of parameters (contents of accumulator and index register 2). Invalid parameters cause unpredictable operation by the subroutines.

PNCHZ – 1442 Output Subroutine

Buffer Size. Maximum of 80 words.

Card Output. This subroutine punches from the I/O buffer the number of characters indicated in the location preceding the buffer. Punching is done in IBM Card Code.

Subroutines Loaded. The following subroutines are loaded with PNCHZ:

HOLEZ, GETAD, EBCTB, HOLTB

READZ – 2501 Input Subroutine

Buffer Size. Maximum of 80 words.

Card Input. This subroutine reads 80 columns from a card and stores the information in the I/O buffer in EBCDIC format.

Subroutines Loaded. The following subroutines are loaded along with READZ:

HOLEZ, GETAD, EBCTB, HOLTB

PRNZ – 1403 Printer Subroutine

Buffer Size. Maximum of 121 characters.

Index Register 2. The first character in the I/O buffer is the carriage control character, followed by up to 120 characters to be printed. Therefore, index register 2 must contain the number of characters to be printed plus one.

The carriage is controlled prior to the printing of a line; no "after-print" carriage control is performed. Following is a list of the carriage control characters and their related functions:

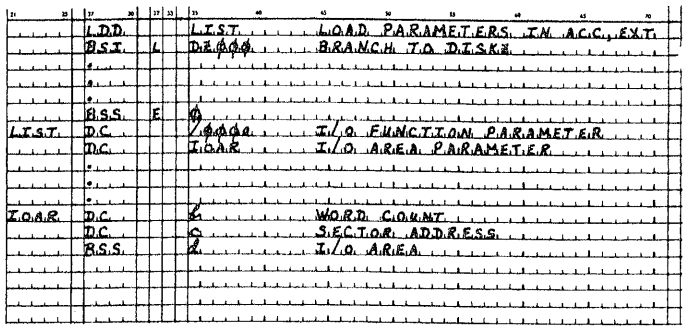
Character	Function
00F1	Skip to channel 1 prior to printing
00F0	Double space prior to printing
004E	No skip or space prior to printing
Any other character	Single space prior to printing

Channel 12 Control. If a punch in channel 12 is encountered while a line is being printed, skip to channel 1 is executed prior to printing the next line.

DISKZ – Disk Input/Output Subroutine

The DISKZ subroutine offers no file protection, no preoperative parameter checks, no Write Immediate function, and no Write Without Readback Check function. It is intended for use by the monitor programs and by FORTRAN programs in which disk FORTRAN I/O is used.

The calling sequence for DISKZ is:



where

a is the I/O function digit. Zero indicates a Read, one a Write.

b is the number of words to be transferred to or from the disk

c is the sector address at which the transfer is to begin,

d is the length of the I/O area. d must be equal to or greater than b.

The word count (first word of the buffer) must be non-negative. The sector address must be the

second word of the buffer. The drive code (0, 1, 2, 3, or 4) is in bits 0-3 of the sector address.

A word count of zero indicates a seek to the cylinder denoted in the sector address. File protection is not provided. If the access arm is not positioned at the cylinder addressed, DISKZ seeks to that cylinder before performing the requested function. A Read follows each Seek to verify that the Seek was successful. No buffer is required for this Read.

Buffer Size. Maximum of 320 words.

Operation. DISKZ performs Read, Seek, and Write With Readback Check functions. Each function returns control to the user after it has been initiated. To determine completion of a disk operation, the user may test DBUSY (in COMMA) until it is cleared to zero. However, DISKZ itself tests this word before initiating an operation. Following a Write, the subroutine performs a Readback Check on the data just written. If it detects an error, it re-executes the Write. Similarily, if a sector is not located or an error is detected during a Read, DISKZ repeats the operation. All operations are attempted 16 times before DISKZ indicates an irrecoverable error.

If a partial sector (less than 320 words) is written, the remaining words of the sector are automatically set to zero.

Subroutines Required. No other subroutines are required by DISKZ.

DATA CODE CONVERSION SUBROUTINES

These subroutines convert data to and from 16-bit binary words and I/O device codes. Refer to the Subroutine Library publication for a detailed introduction to conversion subroutines and for descriptions of the conversion subroutines and codes used by both the 1130 Disk Monitor System, Version 1, and the 1130 Card/Paper Tape System.

The following conversion subroutines are part of the 1130 Disk Monitor System, Version 2:

BINDC – Binary value to IBM Card Code decimal value.
DCBIN – IBM Card Code decimal value to binary value.
BINHX – Binary value to IBM Card Code hexadecimal value.
HXBIN – IBM Card Code hexadecimal value to binary value.
HOLEB – IBM Card Code subset to EBCDIC subset; EBCDIC subset to IBM Card Code subset.
SPEED – IBM Card Code characters to EBCDIC; EBCDIC to IBM Card Code characters.
PAPEB – PTTC/8 subset to EBCDIC subset; EBCDIC subset to PTTC/8 subset.
PAPHL – PTTC/8 subset to IBM Card Code subset; IBM Card Code subset to PTTC/8 subset.
PAPPR – PTTC/8 subset to Console Printer or 1403 Printer code.
HOLPR – IBM Card Code subset to Console Printer or 1403 Printer code.
EBPRT – EBCDIC subset to Console Printer or 1403 Printer code.
BIDEC – 32-bit binary value to IBM Card Code decimal value.
DECBI – IBM Card Code decimal value to 32-bit binary value.
ZIPCO – Supplement to all standard conversions except those involving PTTC/8.

DESCRIPTIONS OF DATA CODES

In addition to the internal 16-bit binary representation, the conversion subroutines handle the following codes:

- Hexadecimal Notation
- IBM Card Code
- Perforated Tape and Transmission Code (PTTC/8)
- Console Printer (1053) Code
- 1403 Printer Code
- Extended Binary Coded Decimal Interchange Code (EBCDIC)

1403 Printer Code

The 1403 Printer uses a six-bit binary code with one parity bit. Data format is two seven-bit characters per word, as follows:

Bit	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15
Value	* P 32 16 8 4 2 1	* P 32 16 8 4 2 1
	1st data character	2nd data character

*=Not Used
P=Parity Bit

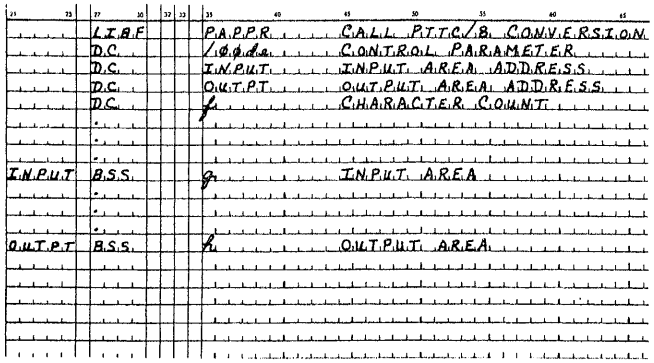
Parity bits are not assigned by the hardware. The conversion subroutine must assign the parity bits and arrange the characters in the form in which they are to be printed.

PAPPR

This subroutine converts PTTC/8 subset to either Console Printer or 1403 Printer code. The conversion to 1403 Printer code is illustrated below.

I/O Locations	Conversion Data		Bits in Core Storage 0 ← → 15			
INPUT	UC	J	0000	1110	0101	0001
INPUT+1	LC	$	0110	1110	0101	1011
OUTPT	J	$	0101	1000	0110	0010

Calling Sequence



where

d is the case initialization digit,

e is the output printer code digit,

f is the number of characters in the input area to be converted,

g is the length of the input area. g must be equal to or greater than f/2 if the character count is even, (f + 1)/2 if the character count is odd.

h is the length of the output area. h must be equal to or greater than f/2, minus the number of paper tape control characters in the input area, plus 1 if the result is odd.

Control Parameter

This parameter consists of four hexadecimal digits. Digits 1 and 2 are not used. The third digit indicates whether or not the case is to be initialized before conversion begins:

0 = Initialize case
1 = Do not alter case

The fourth digit determines the output printer code.

0 = Console Printer code
1 = 1403 Printer code

Input

Input consists of PTTC/8 characters starting in location INPUT. PTTC/8 characters are packed two per binary word. All control characters except case shift (LC or UC) characters are converted to output. Case shift characters are used only to define the case mode of the graphic characters that follow.

Output

Output consists of either Console Printer or 1403 Printer characters starting in location OUTPT. This code is packed two characters per binary word. If overlap of the input and output areas is desired, the address INPUT must be equal to or greater than the address OUTPT. This is necessary because the subroutine starts processing at location INPUT.

Character Count

This parameter specifies the number of PTTC/8 characters in the input area. The count must include

case shift characters, even though they do not appear in the output. Because the input is packed, the character count is not equal to the number of binary words in the input area.

If an odd number of output characters is produced, bits 8-15 of the last used word in the output area are set to a space character.

The conversion is halted whenever the character count is decremented to zero or whenever a new line (NL) control character is detected.

Errors Detected

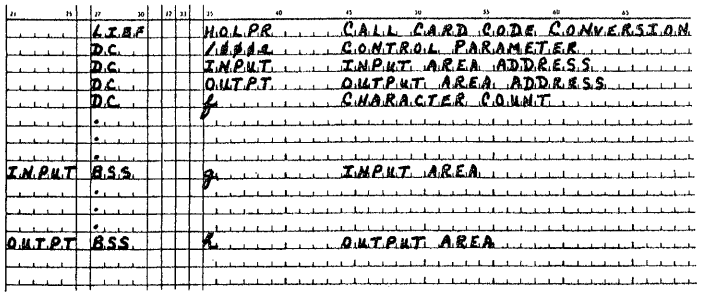
Any input character not marked by an asterisk in Appendix B is in error.

HOLPR

This subroutine converts IBM Card Code subset to either Console Printer or 1403 Printer code. The conversion to 1403 Printer code is illustrated below.

I/O Locations	Conversion Data	Bits in Core Storage 0 ← → 15
INPUT	J	0101 0000 0000 0000
INPUT+1	,	0010 0100 0010 0000
OUTPT	J,	0101 1000 0001 0110

Calling Sequence



where

e is the output printer code digit,

f is the number of characters in the input area to be converted,

g is the length of the input area. g must be equal to or greater than f.

h is the length of the output area. h must be equal to or greater than f/2.

Control Parameter

This parameter consists of four hexadecimal digits. Digits 1-3 are not used. The fourth digit determines the output printer code.

0 = Console Printer code
1 = 1403 Printer code

Input

Input consists of IBM Card Code characters, starting in location INPUT. The characters are not packed.

Output

Output consists of either Console Printer or 1403 Printer characters, starting in location OUTPT. The code is packed two characters per binary word.

The input area may overlap the output area if the address INPUT is equal to or greater than the address OUTPT. The subroutine starts processing at location INPUT.

Character Count

This number specifies the number of IBM Card Code characters to be converted and is equal to the number of words in the input area. If an odd count is specified, bits 8-15 of the last word used in the output area are not altered.

Errors Detected

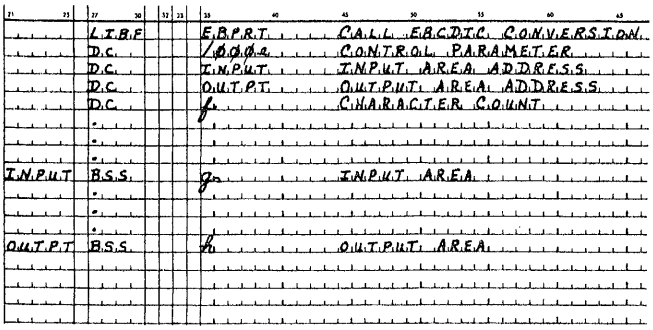
Any input character not marked with an asterisk in Appendix B is in error.

EBPRT

This subroutine converts EBCDIC subset to either Console Printer or 1403 Printer Code. The conversion to 1403 Printer code is shown below.

I/O Locations	Conversion Data	Core Storage Bits 0 ← → 15
INPUT	LE	1101 0011 1100 0101
INPUT+1	ES	1100 0101 1110 0010
OUTPUT	LE	0001 1010 0110 1000
OUTPT+1	ES	0110 1000 0000 1101

Calling Sequence



where

e is the output printer code digit,

f is the number of characters in the input area to be converted,

g is the length of the input area. g must be equal to or greater than f.

h is the length of the output area. h must be equal to or greater than f.

Control Parameter

This parameter consists of four hexadecimal digits, Digits 1-3 are not used. The fourth digit determines the output printer code.

0 = Console Printer code
1 = 1403 Printer code

Input

Input consists of EBCDIC characters starting in location INPUT. EBCDIC characters are packed two per word.

Output

Output consists of either Console Printer or 1403 Printer code starting in location OUTPT. The code is packed two characters per binary word.

The address INPUT must be equal to or greater than the address OUTPT if overlap of the input and output areas is desired. The subroutine starts processing at location INPUT.

Character Count

This parameter specifies the number of EBCDIC characters to be converted. This count is not equal to the number of words in the input area. If an odd count is specified, bits 8-15 of the last word used in the output area are not altered.

Errors Detected

Any input character not marked with an asterisk in Appendix B is in error.

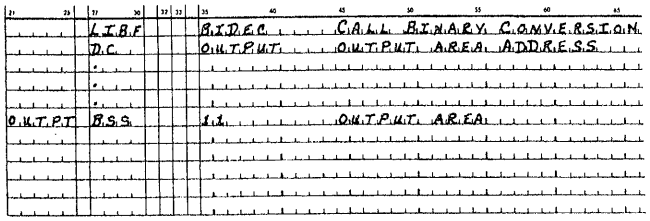
BIDEC

This subroutine converts a 32-bit binary value to its decimal equivalent in ten IBM Card Code numeric characters and one sign character. The conversion is illustrated below.

I/O Locations	Conversion Data	Core Storage Bits 0 ← → 15
Accumulator	+0016777218	0000 0001 0000 0000
Extension		0000 0000 0000 0010

I/O Locations	Conversion Data	Core Storage Bits 0 ← → 15			
OUTPT	+	1000	0000	1010	0000
OUTPT+1	0	0010	0000	0000	0000
OUTPT+2	0	0010	0000	0000	0000
OUTPT+3	1	0001	0000	0000	0000
OUTPT+4	6	0000	0000	1000	0000
OUTPT+5	7	0000	0000	0100	0000
OUTPT+6	7	0000	0000	0100	0000
OUTPT+7	7	0000	0000	0100	0000
OUTPT+8	2	0000	1000	0000	0000
OUTPT+9	1	0001	0000	0000	0000
OUTPT+10	8	0000	0000	0010	0000

Calling Sequence



Input

Input is a 32-bit binary value in the accumulator and extension.

Output

Output is an IBM Card Code sign character (+ or -) in location OUTPT, and ten IBM Card Code numeric characters in OUTPT+1 through OUTPT+10.

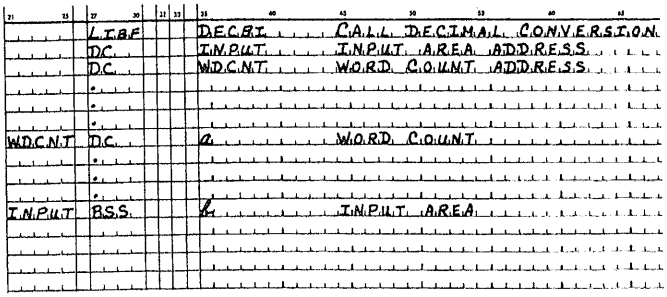
Errors Detected

The BIDEC subroutine does not detect errors.

DECBI

This subroutine converts a decimal value consisting of ten IBM Card Code numeric characters and a sign character to a 32-bit binary word. This subroutine is the inverse of the BIDEC subroutine (see above) except that fewer than ten characters may be specified.

Calling Sequence



where

a is the number of characters to be converted not including the sign character,

b is the length of the input area. b must be equal to a plus 1.

Input

Input is an IBM Card Code sign character in location INPUT, the address (WDCNT) of the number of characters (1 to 10) to be converted, and specified number of characters in IBM Card Code in locations INPUT+1 through INPUT+N (where N = 1, 2 . . . 10).

Output

Output is a 32-bit binary word, containing the converted value, in the accumulator and extension.

Errors Detected

Any of the following conditions causes the Overflow indicator to be turned on, the Carry indicator to be turned off, and an immediate exit to be made back to the caller:

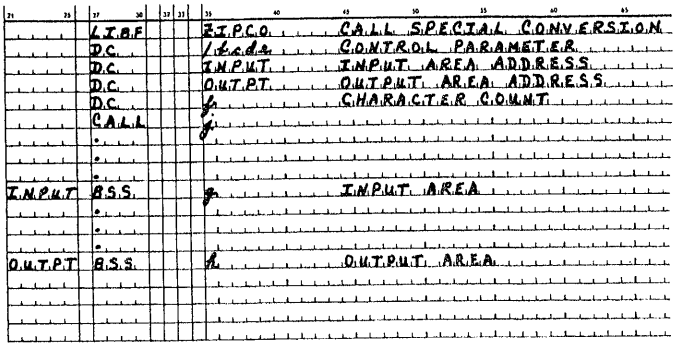
1. Any sign other than a plus, minus, blank, or ampersand.
2. Any character other than a space or 0 through 9.
3. Any converted value greater than +2,147,483,647 or less than -2,147,483,648.

ZIPCO

This subroutine supplements all standard conversions except those involving PTTC/8 code. It offers the

user the option of supplying his own conversion tables and codes. ZIPCO uses direct table access and is considerably faster than the other conversion subroutines.

Calling Sequence



where

b is the input code digit,

c is the packed input digit,

d is the output code digit,

e is the packed output digit,

f is the number of characters to be converted,

g is the length of the input area,

h is the length of the output area,

j is the name of the conversion table to be used. This CALL is not executed; however, it is required to cause the loading of the desired conversion table.

Control Parameter

This parameter consists of four hexadecimal digits as follows.

Digit 1
- 1 for 12-bit IBM Card Code input
- 0 for all other types of input

Digit 2
- 1 for unpacked input
- 0 for packed input

Digit 3
- 1 for 12-bit IBM Card Code output
- 0 for 8-bit IBM Card Code and all other types of output

Digit 4
- 1 for unpacked output
- 0 for packed output

Input

Input consists of packed or unpacked characters in the code specified by the conversion table and starting at location INPUT.

Output

Output consists of packed or unpacked characters in the code specified by the conversion table and starting at location OUTPT.

Character Count

This parameter specifies the number of input characters to be converted. If an odd count is specified with packed input, bits 8-15 of the last word used in the output area are not altered.

Table

The type of conversion is determined by the table called with ZIPCO. The user may call one of the IBM-supplied conversion tables or he may supply his own.

The following IBM-supplied System Library tables may be called with ZIPCO.

- EBCCP – EBCDIC to Console Printer Code.
- EBHOL – EBCDIC to IBM Card Code.
- EBPT3 – EBCDIC to 1403 Printer code.
- CPEBC – Console Printer code to EBCDIC.
- CPHOL – Console Printer code to IBM Card Code.
- CPPT3 – Console Printer code to 1403 Printer code.
- HLEBC – IBM Card Code to EBCDIC.
- HOLCP – IBM Card Code to Console Printer code.
- HLPT3 – IBM Card Code to 1403 Printer code.
- PT3EB – 1403 Printer code to EBCDIC.
- PT3CP – 1403 Printer code to Console Printer code.
- PTHOL – 1403 Printer code to IBM Card Code.

Each conversion table consists of 256 characters--128 words with two 8-bit characters per word. The seven low-order bits of the character to be converted (input character) are used as an address. The

address designates the position in the table of the corresponding conversion character. The high-order bit (bit 0) of the input character designates which half of the table word is to be used. When bit 0 is 1, the left half of the word is used. When bit 0 is 0, the right half of the word is used. All dummy entries of the IBM-supplied tables contain the code for a blank.

The following is an example of the conversion performed by ZIPCO. The tables show (1) the input EBCDIC values, (2) the table EBPT3 used for the conversion, and (3) the output characters in 1403 Printer code.

Input Location	Value
INPUT	1111 0010 0111 0010
INPUT+1	0000 0000 1000 0000
INPUT+2	0111 1111 1111 1111

Table Location	Value
EBPT3	0111 1111 0111 1111
EBPT3+1	0111 1111 0111 1111
.	.
.	.
.	.
EBPT3 +113	0000 0001 0111 1111
.	.
.	.
.	.
EBPT3+127	0111 1111 0111 1111

Output Location	Value	1403 Print Character
OUTPT	0000 0001 0111 1111	2, b
OUTPT+1	0111 1111 0111 1111	b, b
OUTPT+2	0111 1111 0111 1111	b, b

Errors Detected

No errors are detected by ZIPCO.

ARITHMETIC AND FUNCTION SUBROUTINES

The arithmetic and function subroutines in the 1130 Disk Monitor System, Version 2, System Library are described in the Subroutine Library publication.

WRITING ISSs AND ILSs

ISS and ILS subroutines are written as described in the Subroutine Library publication. Table 8 lists the ISS/ILS correspondence applicable to the 1130 Disk Monitor System, Version 2.

Table 8. ISS/ILS Correspondence

ISS Number	Device	Interrupt Level Assignments
1	1442 Card Read Punch, 1442 Card Punch	0,4
2	Keyboard/Console Printer	4
3	1134/1055 Paper Tape Reader Punch	4
4	2501 Card Reader	4
5	Disk	2
6	1132 Printer	1
7	1627 Plotter	3
8	Synchronous Communications Adapter	1
9	1403 Printer	4
10	1231 Optical Mark Page Reader	4

DISK MAINTENANCE PROGRAMS

The disk maintenance programs are mainline programs and subroutines that are a part of the System Library and that initialize and modify disk cartridge IDs, addresses, and tables required by the monitor system. Normally, they should never be deleted from the System Library.

The disk maintenance programs are:

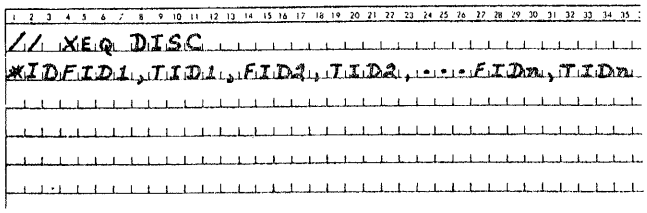
DISC – Disk Initialization Program
IDENT – Print Cartridge ID Program
ID – Change Cartridge ID Program
COPY – Disk Copy Program

ADRWS	–	Write Sector Address in Working Storage Program
DLCIB	–	Delete CIB Program
MODIF	–	System Maintenance Program
SYSUP	–	DCOM Updating Program

The disk maintenance programs (except ADRWS and SYSUP) are called by an XEQ monitor control record (see "Monitor Control Records" under Supervisor). Some disk maintenance programs require an ID control record. The format and use of the ID control record is described under the program descriptions which follow.

DISC – DISK INITIALIZATION PROGRAM

This program initializes up to four satellite cartridges -- all but the master cartridge on logical disk drive 0. It writes the sector addresses, defective cylinder addresses, cartridge ID, a LET, a DCOM, and a CIB on each cartridge initialized. The calling sequence for DISC is



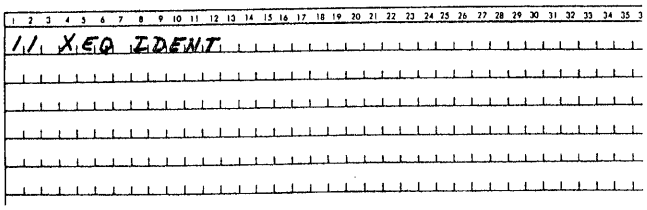
where

FID1 through FIDn are the IDs currently on the cartridges to be initialized.

TID1 through TIDn are the IDs to be written on those cartridges.

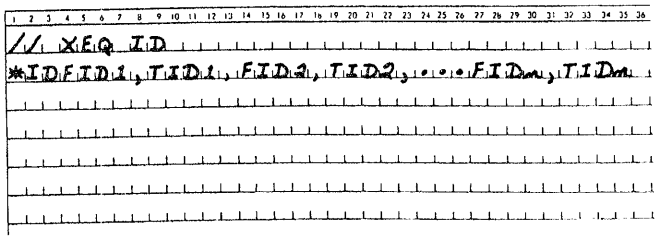
IDENT – PRINT CARTRIDGE ID PROGRAM

This program prints out the ID, the physical drive number, and the logical drive number of each disk cartridge mounted on the system. The calling sequence for IDENT is



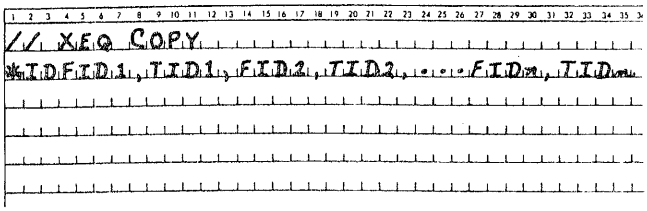
ID – CHANGE CARTRIDGE ID PROGRAM

This program changes the ID on up to four disk cartridges. The ID control record required is identical to that used by DISC (see above). The calling sequence for ID is



COPY – DISK COPY PROGRAM

This program copies the contents (except the cartridge ID) of one disk cartridge onto another. The cartridge to be copied onto must have been previously initialized (see DISC). The calling sequence for COPY is



where

FID1 through FIDn are the IDs of the cartridges to be copies,

TID1 through TIDn are the IDs of the cartridges onto which the copies are to be made

ADRWS – WRITE SECTOR ADDRESSES IN WORKING STORAGE PROGRAM

This program, linked to from DUP on detection of a DWADR DUP control record, writes the sector address on all sectors of Working Storage of the disk cartridge specified in the DWADR control record. (See "DWADR" under Disk Utility Program.)

DLCIB – DELETE CIB PROGRAM

This program deletes the CIB from a non-system cartridge. If a User Area is defined, the User Area

is moved one cylinder closer to cylinder 0. The new addresses of disk areas moved as the result of the deletion of the CIB are reflected in DCOM on the master cartridge and on the non-system cartridge from which the CIB is deleted, and in COMMA. The calling sequence for DLCIB is:

```
// XEQ DLCIB
*IDCART
```

where

CART is the ID of the non-system cartridge from which the CIB is to be deleted.

MODIF–SYSTEM MAINTENANCE PROGRAM

This program updates the master cartridge; that is, it makes changes to the version and modification level word in DCOM, the Supervisor, DUP, FORTRAN Compiler, Assembler, and/or System Library. A card deck or paper tape containing corrections to update the monitor system to the latest version and modification level is supplied by IBM. Every modification must be run to update the version and modification level even if the program affected has been deleted.

The calling sequence for MODIF is:

```
// XEQ MODIF                1
```

A one must appear in position 19 of the XEQ record to specify DISK1. Input to the program can be cards or paper tape. This program uses the Reload Table to update all references to all monitor programs which utilize SLET.

System Program Maintenance

Typical input for a system program update is shown in Figure 13.

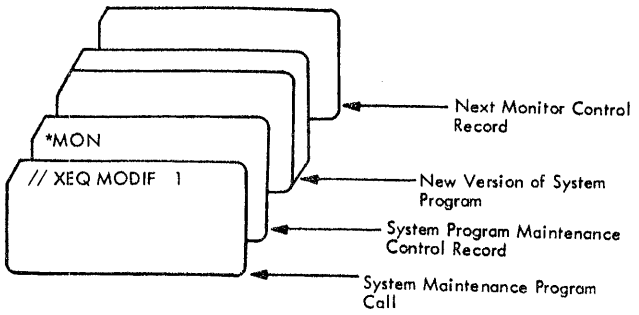


Figure 13. Layout of an Input Deck for a System Program Update

System Program Maintenance (Patch) Control Record

Each monitor program phase to be changed requires a patch control record. If the FORTRAN Compiler or the Assembler is to be changed, MODIF determines if that monitor program has been voided from the disk. If so, the modification is not made.

The format of the patch control record is:

Card Columns	Contents	Notes
1-4	*MON	These characters identify a system patch to the FORTRAN Compiler, Assembler, DUP, Supervisor, Core Load Builder, System device subroutines, or Core Image Loader.
6-9	vmmm	The version (v) and modification level (mmm) are specified in hexadecimal.
11-14	xxxx	The SLET ID of the monitor program phase to which the patch is to be made is specified in hexadecimal. 0000 = an absolute patch (see columns 28-31, 33-36).
16-19	nnnn	"nnnn" specifies (in hex) the number of patch records following this patch control record.
21	B or H	This character identifies the format of the patch records (Binary system format or Hex patch format).

Card Columns	Contents	Notes
23-26	pppp	"pppp" specifies (in hex) the total number of patches. This should be the exact number of patch control records to be processed. This parameter is read from the first patch control record.
28-31	dsss	The drive code (d) and sector address (sss) of the program to be patched are specified in hexadecimal. This field is used only when the SLET ID is 0.
33-36	cccc	"cccc" specifies (in hex) the core address of the first word of this sector.
37-80	Not Used	

Patch Data Record

Binary System Format.

Word	Contents
1	Location
2	Checksum
3	Type Code (first 8 bits) 00001010
4-9	Relocation Indicators
10-54	Data words 1 through 45
55-60	ID and sequence number

Hex Patch Format.

Card Column	Contents	Notes
1-4	aaaa	"aaaa" specifies (in hex) the core address (origin) of the patch. Each patch record must have a core address.
6-9, 11-14, 16-19, etc.		Each 4-column field contains one word of patch data (in hex). Up to 15 words of patch data can be specified per record.

System Library Maintenance

Changes to the System Library require reloading the new mainline or subroutine. MODIF updates the version and modification level word; the actual reload is performed by a DUP DELETE operation, followed by a DUP STORE operation.

Typical input for System Library maintenance is shown in Figure 14.

System Library Maintenance (Subroutine Header) Control Record

The subroutine header control record must go through MODIF even if the subroutine being modified has been deleted. The format of a subroutine header control record is:

Card Column	Contents	Notes
1-4	*SUB	These characters identify a system patch to the System Library.
6-9	vmmm	The version (v) and modification level (mmm) are specified in hexadecimal.
16-19	nnnn	"nnnn" specifies (in hex) the number of deletes and stores to be processed.
20-80	Not Used	

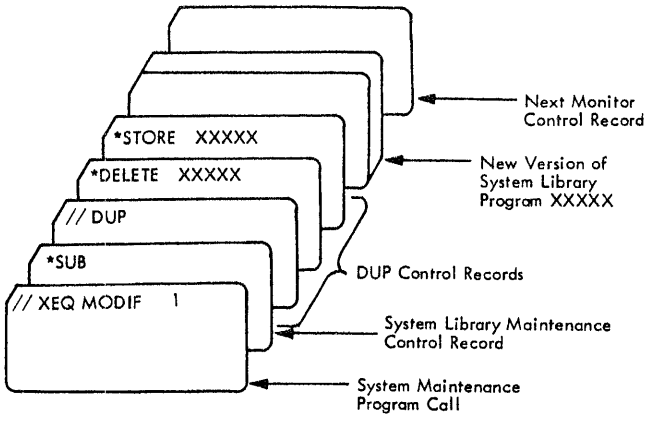


Figure 14. Layout of an Input Deck for a System Library Update

SYSUP—DCOM UPDATING PROGRAM

This subroutine is used whenever a core load requires changing disk cartridges. It updates DCOM on the master cartridge (logical drive 0) with the IDs and DCOM information from all satellite cartridges mounted on the system that are specified in the list or array in the calling sequence.

The Assembler language calling sequence for SYSUP is:

```
            CALL    SYSUP     CALL DCOM UPDATE
            DC      LIST
            .
            .
            .
LIST        DC      a
            DC      b
            DC      c
            DC      d
            DC      0
```

where

a is the ID of the first cartridge on the system,

b is the ID of the second cartridge on the system,

c is the ID of the third cartridge on the system,

d is the ID of the last cartridge on the system. This ID must be followed by a word of zeros.

The FORTRAN calling sequence for SYSUP is:

```
      CALL SYSUP(a)
```

where

a is the name of an array containing the IDs of the cartridge on the system. The element of the array following the last ID must be 0.

PAPER TAPE MAINLINE PROGRAMS

These programs perform paper tape utility functions. The paper tape mainline programs are:

PTREP – Paper Tape Reproducing Program
PTUTL – Paper Tape Utility Program

PTREP

This program reproduces paper tapes. It reads and punches characters with no intermediate conversion. The calling sequence for PTREP is:

```
// XEQ PTREP
```

PTUTL

This program accepts input from the Keyboard or the 1134 Paper Tape Reader and provides output on the Console Printer and/or the 1055 Paper Tape Punch.

PTUTL allows changes and/or additions to FORTRAN and Assembler language source records as well as monitor control records according to the Console Entry Switch options described below.

Switch	Setting	Option
0	On	Print record after reading
1	On	Read records from 1134
2	On	Accept Keyboard input
3	On	Punch records on 1055
14	On	Wait after punching
15	On	Wait after printing
All	Off	Exit to the Supervisor

The calling sequence for PUTUTL is:

```
// XEQ PTUTL
```

APPENDIX A. UTILITY PROGRAMS

These utility programs – each self-loading and complete with subroutines – are separate from the System Library and enable the user to perform operations external to the monitor system. The utility programs are:

- Console Printer Core Dump
- Printer Core Dump
- Disk Cartridge Initialization Program (DCIP)

CONSOLE PRINTER CORE DUMP

This program aids the user in debugging programs by dumping selected portions of core on the Console Printer.

Each core location is dumped as a four-digit hexadecimal word. A space separates each word. The number of words typed per line depends on the margin settings of the Console Printer. The first word dumped is the starting address of the dump (as specified in the Console Entry Switches).

PRINTER CORE DUMP

This program dumps core in hexadecimal format on either the 1403 Printer or the 1132 Printer.

Each line contains a four-digit hexadecimal address, followed by 16 four-digit hexadecimal words. A space separates the address and each word in the printed line. An additional space is inserted between each group of four words.

To decrease dump time, the program does not print consecutive duplicate lines. Before printing a line, it compares the 16 words with the 16 words printed on the previous line. If they are identical, the program goes on to the next 16 words in core. If not, it spaces one line and prints. The address printed is that of the first word on the line. The program skips to the top of a new page at the start of the dump and whenever it encounters channel 12 on the printer. NOTE: If both the 1132 and the 1403 Printers are on the system and both are ready, the dump is printed on the 1403 Printer.

DISK CARTRIDGE INITIALIZATION PROGRAM (DCIP)

The Disk Cartridge Initialization Program (DCIP) is composed of

- A disk initialization subroutine
- A disk copy subroutine
- A disk dump subroutine

INITIALIZATION

The disk initialization subroutine of DCIP

1. Writes disk sector addresses on all cylinders.
2. Determines which, if any, sectors are defective and writes the address(es) of the cylinder(s) containing the defective sectors on sector 0.
3. Establishes a file-protected area for the disk cartridge.
4. Puts an ID on the disk cartridge.
5. Establishes a DCOM, LET, and CIB.

The disk I/O subroutines 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 0 results in a read or write error after the operation has been attempted 16 times.

Cylinder zero may not be a defective cylinder. If it is, the cartridge cannot be initialized.

At the completion of DCIP, a four-word table is written on sector 0. 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 four contains the cartridge ID, in binary.

COPY

The disk copy subroutine of DCIP

1. Checks to ensure that both the cartridge to be copied and the cartridge onto which the copy is to be made have been correctly initialized.

2. Copies a cartridge from any drive onto a cartridge on any other drive.

DUMP

The disk dump subroutine of DCIP

1. Dumps any disk sector(s) from any drive.
2. Prints the dump on the fastest printer on the system (in the order of speed – 1403, 1132, or Console Printer)

The number of consecutive sectors to be dumped, as well as the address of the sector to be dumped (the first sector if more thanone) is specified in the Console Entry Switches.

Each sector printout is 20 lines - 16 four-digit hexadecimal words per line. Two sectors are printed on each page and each sector is preceded by a 3-word header. The first digit of the first header word is the number of sectors remaining to be dumped. The remaining three digits show the sector address of the sector being dumped. The second header word, which is the first word of the sector, is the correct address of the sector. The third word is the logical sector address, taking into account any defective cylinders.

APPENDIX B. CHARACTER CODE CHART

Ref No.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM Card Code Rows 12	11	0	9	8	7-1	IBM Card Code Hex	Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex (U-Upper Case L-Lower Case)	Console Printer Hex Notes	1403 Printer Hex
0	0000	0000	00	12		0	9	8	1	B030	NUL				
1		0001	01	12			9		1	9010					
2		0010	02	12			9		2	8810					
3		0011	03	12			9		3	8410					
4		0100	04	12			9		4	8210	PF Punch Off				
5*		0101	05	12			9		5	8110	HT Horiz.Tab		6D (U/L)	41 ①	
6*		0110	06	12			9		6	8090	LC Lower Case		6E (U/L)		
7*		0111	07	12			9		7	8050	DEL Delete		7F (U/L)		
8		1000	08	12			9	8		8030					
9		1001	09	12			9	8	1	9030					
10		1010	0A	12			9	8	2	8830					
11		1011	0B	12			9	8	3	8430					
12		1100	0C	12			9	8	4	8230					
13		1101	0D	12			9	8	5	8130					
14		1110	0E	12			9	8	6	80B0					
15		1111	0F	12			9	8	7	8070					
16	0001	0000	10	12	11		9	8	1	D030					
17		0001	11		11		9		1	5010					
18		0010	12		11		9		2	4810					
19		0011	13		11		9		3	4410					
20*		0100	14		11		9		4	4210	RES Restore		4C (U/L)	05 ②	
21*		0101	15		11		9		5	4110	NL New Line		DD (U/L)	81 ③	
22*		0110	16		11		9		6	4090	BS Backspace		5E (U/L)	11	
23		0111	17		11		9		7	4050	IDL Idle				
24		1000	18		11		9	8		4030					
25		1001	19		11		9	8	1	5030					
26		1010	1A		11		9	8	2	4830					
27		1011	1B		11		9	8	3	4430					
28		1100	1C		11		9	8	4	4230					
29		1101	1D		11		9	8	5	4130					
30		1110	1E		11		9	8	6	40B0					
31		1111	1F		11		9	8	7	4070					
32	0010	0000	20		11	0	9	8	1	7030					
33		0001	21			0	9		1	3010					
34		0010	22			0	9		2	2810					
35		0011	23			0	9		3	2410					
36		0100	24			0	9		4	2210	BYP Bypass				
37*		0101	25			0	9		5	2110	LF Line Feed		3D (U/L)	03	
38*		0110	26			0	9		6	2090	EOB End of Block		3E (U/L)		
39		0111	27			0	9		7	2050	PRE Prefix				
40		1000	28			0	9	8		2030					
41		1001	29			0	9	8	1	3030					
42		1010	2A			0	9	8	2	2830					
43		1011	2B			0	9	8	3	2430					
44		1100	2C			0	9	8	4	2230					
45		1101	2D			0	9	8	5	2130					
46		1110	2E			0	9	8	6	20B0					
47		1111	2F			0	9	8	7	2070					
48	0011	0000	30	12	11	0	9	8	1	F030					
49		0001	31				9		1	1010					
50		0010	32				9		2	0810					
51		0011	33				9		3	0410					
52		0100	34				9		4	0210	PN Punch On				
53*		0101	35				9		5	0110	RS Reader Stop		0D (U/L)	09 ④	
54*		0110	36				9		6	0090	UC Upper Case		0E (U/L)		
55		0111	37				9		7	0050	EOT End of Trans.				
56		1000	38				9	8		0030					
57		1001	39				9	8	1	1030					
58		1010	3A				9	8	2	0830					
59		1011	3B				9	8	3	0430					
60		1100	3C				9	8	4	0230					
61		1101	3D				9	8	5	0130					
62		1110	3E				9	8	6	00B0					
63		1111	3F				9	8	7	0070					

NOTES: Typewriter Output

① Tabulate
② Shift to black
③ Carrier Return
④ Shift to red

* Recognized by all Conversion subroutines
Codes that are not asterisked are recognized only by the SPEED subroutine

Ref No.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM Card Code Rows 12 11 0 9 8 7-1	IBM Card Code Hex	Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex	1403 Printer Hex
64*	0100	0000	40	no punches	0000	blank	ǂ	10 (U/L)	21	7F
65		0001	41	12 0 9 1	B010					
66		0010	42	12 0 9 2	A810					
67		0011	43	12 0 9 3	A410					
68		0100	44	12 0 9 4	A210					
69		0101	45	12 0 9 5	A110					
70		0110	46	12 0 9 6	A090					
71		0111	47	12 0 9 7	A050					
72		1000	48	12 0 9 8	A030					
73		1001	49	12 8 1	9020					
74*		1010	4A	12 8 2	8820	¢		20 (U)	02	
75*		1011	4B	12 8 3	8420	. (period)	4B	6B (L)	00	6E
76*		1100	4C	12 8 4	8220	<		02 (U)	DE	
77*		1101	4D	12 8 5	8120	(	4D	19 (U)	FE	2F
78*		1110	4E	12 8 6	80A0	+	4E	70 (U)	DA	6D
79*		1111	4F	12 8 7	8060	ǀ (logical OR)		3B (U)	C6	
80*	0101	0000	50	12	8000	&	50	70 (L)	44	15
81		0001	51	12 11 9 1	D010					
82		0010	52	12 11 9 2	C810					
83		0011	53	12 11 9 3	C410					
84		0100	54	12 11 9 4	C210					
85		0101	55	12 11 9 5	C110					
86		0110	56	12 11 9 6	C090					
87		0111	57	12 11 9 7	C050					
88		1000	58	12 11 9 8	C030					
89		1001	59	11 8 1	5020					
90*		1010	5A	11 8 2	4820	!		5B (U)	42	
91*		1011	5B	11 8 3	4420	$	5B	5B (L)	40	62
92*		1100	5C	11 8 4	4220	*	5C	08 (U)	D6	23
93*		1101	5D	11 8 5	4120	)	5D	1A (U)	F6	57
94*		1110	5E	11 8 6	40A0	;		13 (U)	D2	
95*		1111	5F	11 8 7	4060	¬ (logical NOT)		6B (U)	F2	
96*	0110	0000	60	11	4000	- (dash)	60	40 (L)	84	61
97*		0001	61	0 1	3000	/	61	31 (L)	BC	4C
98		0010	62	11 0 9 2	6810					
99		0011	63	11 0 9 3	6410					
100		0100	64	11 0 9 4	6210					
101		0101	65	11 0 9 5	6110					
102		0110	66	11 0 9 6	6090					
103		0111	67	11 0 9 7	6050					
104		1000	68	11 0 9 8	6030					
105		1001	69	0 8 1	3020					
106		1010	6A	12 11	C000					
107*		1011	6B	0 8 3	2420	, (comma)	6B	3B (L)	80	16
108*		1100	6C	0 8 4	2220	%		15 (U)	06	
109*		1101	6D	0 8 5	2120	_ (underscore)		40 (U)	BE	
110*		1110	6E	0 8 6	20A0	>		07 (U)	46	
111*		1111	6F	0 8 7	2060	?		31 (U)	86	
112	0111	0000	70	12 11 0	E000					
113		0001	71	12 11 0 9 1	F010					
114		0010	72	12 11 0 9 2	E810					
115		0011	73	12 11 0 9 3	E410					
116		0100	74	12 11 0 9 4	E210					
117		0101	75	12 11 0 9 5	E110					
118		0110	76	12 11 0 9 6	E090					
119		0111	77	12 11 0 9 7	E050					
120		1000	78	12 11 0 9 8	E030					
121		1001	79	8 1	1020					
122*		1010	7A	8 2	0820	:		04 (U)	82	
123*		1011	7B	8 3	0420	#		0B (L)	C0	
124*		1100	7C	8 4	0220	@		20 (L)	04	
125*		1101	7D	8 5	0120	' (apostrophe)	7D	16 (U)	E6	0B
126*		1110	7E	8 6	00A0	=	7E	01 (U)	C2	4A
127*		1111	7F	8 7	0060	"		0B (U)	E2	

ǂ Any code other than those defined will be interpreted by PRNT1 as a space.

Ref No.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM Card Code Rows 12	11	0	9	8	7-1	IBM Card Code Hex	Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex	1403 Printer Hex
128	1000	0000	80	12		0		8	1	B020					
129		0001	81	12		0			1	B000	a				
130		0010	82	12		0			2	A800	b				
131		0011	83	12		0			3	A400	c				
132		0100	84	12		0			4	A200	d				
133		0101	85	12		0			5	A100	e				
134		0110	86	12		0			6	A080	f				
135		0111	87	12		0			7	A040	g				
136		1000	88	12		0		8		A020	h				
137		1001	89	12		0	9			A010	i				
138		1010	8A	12		0		8	2	A820					
139		1011	8B	12		0		8	3	A420					
140		1100	8C	12		0		8	4	A220					
141		1101	8D	12		0		8	5	A120					
142		1110	8E	12		0		8	6	A0A0					
143		1111	8F	12		0		8	7	A060					
144	1001	0000	90	12	11			8	1	D020					
145		0001	91	12	11				1	D000	j				
146		0010	92	12	11				2	C800	k				
147		0011	93	12	11				3	C400	l				
148		0100	94	12	11				4	C200	m				
149		0101	95	12	11				5	C100	n				
150		0110	96	12	11				6	C080	o				
151		0111	97	12	11				7	C040	p				
152		1000	98	12	11			8		C020	q				
153		1001	99	12	11		9			C010	r				
154		1010	9A	12	11			8	2	C820					
155		1011	9B	12	11			8	3	C420					
156		1100	9C	12	11			8	4	C220					
157		1101	9D	12	11			8	5	C120					
158		1110	9E	12	11			8	6	C0A0					
159		1111	9F	12	11			8	7	C060					
160	1010	0000	A0		11	0		8	1	7020					
161		0001	A1		11	0			1	7000					
162		0010	A2		11	0			2	6800	s				
163		0011	A3		11	0			3	6400	t				
164		0100	A4		11	0			4	6200	u				
165		0101	A5		11	0			5	6100	v				
166		0110	A6		11	0			6	6080	w				
167		0111	A7		11	0			7	6040	x				
168		1000	A8		11	0		8		6020	y				
169		1001	A9		11	0	9			6010	z				
170		1010	AA		11	0		8	2	6820					
171		1011	AB		11	0		8	3	6420					
172		1100	AC		11	0		8	4	6220					
173		1101	AD		11	0		8	5	6120					
174		1110	AE		11	0		8	6	60A0					
175		1111	AF		11	0		8	7	6060					
176	1011	0000	B0	12	11	0		8	1	F020					
177		0001	B1	12	11	0			1	F000					
178		0010	B2	12	11	0			2	E800					
179		0011	B3	12	11	0			3	E400					
180		0100	B4	12	11	0			4	E200					
181		0101	B5	12	11	0			5	E100					
182		0110	B6	12	11	0			6	E080					
183		0111	B7	12	11	0			7	E040					
184		1000	B8	12	11	0		8		E020					
185		1001	B9	12	11	0	9			E010					
186		1010	BA	12	11	0		8	2	E820					
187		1011	BB	12	11	0		8	3	E420					
188		1100	BC	12	11	0		8	4	E220					
189		1101	BD	12	11	0		8	5	E120					
190		1110	BE	12	11	0		8	6	E0A0					
191		1111	BF	12	11	0		8	7	E060					

Ref No.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM Card Code Rows 12	11	0	9	8	7-1	IBM Card Code Hex	Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex	1403 Printer Hex
192	1100	0000	C0	12		0				A000	(+ zero)				
193*		0001	C1	12					1	9000	A	C1	61 (U)	3C or 3E	64
194*		0010	C2	12					2	8800	B	C2	62 (U)	18 or 1A	25
195*		0011	C3	12					3	8400	C	C3	73 (U)	1C or 1E	26
196*		0100	C4	12					4	8200	D	C4	64 (U)	30 or 32	67
197*		0101	C5	12					5	8100	E	C5	75 (U)	34 or 36	68
198*		0110	C6	12					6	8080	F	C6	76 (U)	10 or 12	29
199*		0111	C7	12					7	8040	G	C7	67 (U)	14 or 16	2A
200*		1000	C8	12				8		8020	H	C8	68 (U)	24 or 26	6B
201*		1001	C9	12			9			8010	I	C9	79 (U)	20 or 22	2C
202		1010	CA	12		0	9	8	2	A830					
203		1011	CB	12		0	9	8	3	A430					
204		1100	CC	12		0	9	8	4	A230					
205		1101	CD	12		0	9	8	5	A130					
206		1110	CE	12		0	9	8	6	A0B0					
207		1111	CF	12		0	9	8	7	A070					
208	1101	0000	D0		11	0				6000	(- zero)				
209*		0001	D1		11				1	5000	J	D1	51 (U)	7C or 7 E	58
210*		0010	D2		11				2	4800	K	D2	52 (U)	58 or 5A	19
211*		0011	D3		11				3	4400	L	D3	43 (U)	5C or 5E	1A
212*		0100	D4		11				4	4200	M	D4	54 (U)	70 or 72	5B
213*		0101	D5		11				5	4100	N	D5	45 (U)	74 or 76	1C
214*		0110	D6		11				6	4080	O	D6	46 (U)	50 or 52	5D
215*		0111	D7		11				7	4040	P	D7	57 (U)	54 or 56	5E
216*		1000	D8		11			8		4020	Q	D8	58 (U)	64 or 66	1F
217*		1001	D9		11		9			4010	R	D9	49 (U)	60 or 62	20
218		1010	DA	12	11		9	8	2	C830					
219		1011	DB	12	11		9	8	3	C430					
220		1100	DC	12	11		9	8	4	C230					
221		1101	DD	12	11		9	8	5	C130					
222		1110	DE	12	11		9	8	6	C0B0					
223		1111	DF	12	11		9	8	7	C070					
224	1110	0000	E0			0		8	2	2820					
225		0001	E1		11	0	9		1	7010					
226*		0010	E2			0			2	2800	S	E2	32 (U)	98 or 9A	0D
227*		0011	E3			0			3	2400	T	E3	23 (U)	9C or 9E	0E
228*		0100	E4			0			4	2200	U	E4	34 (U)	B0 or B2	4F
229*		0101	E5			0			5	2100	V	E5	25 (U)	B4 or B6	10
230*		0110	E6			0			6	2080	W	E6	26 (U)	90 or 92	51
231*		0111	E7			0			7	2040	X	E7	37 (U)	94 or 96	52
232*		1000	E8			0		8		2020	Y	E8	38 (U)	A4 or A6	13
233*		1001	E9			0	9			2010	Z	E9	29 (U)	A0 or A2	54
234		1010	EA		11	0	9	8	2	6830					
235		1011	EB		11	0	9	8	3	6430					
236		1100	EC		11	0	9	8	4	6230					
237		1101	ED		11	0	9	8	5	6130					
238		1110	EE		11	0	9	8	6	60B0					
239		1111	EF		11	0	9	8	7	6070					
240*	1111	0000	F0			0				2000	0	F0	1A (L)	C4	49
241*		0001	F1						1	1000	1	F1	01 (L)	FC	40
242*		0010	F2						2	0800	2	F2	02 (L)	D8	01
243*		0011	F3						3	0400	3	F3	13 (L)	DC	02
244*		0100	F4						4	0200	4	F4	04 (L)	F0	43
245*		0101	F5						5	0100	5	F5	15 (L)	F4	04
246*		0110	F6						6	0080	6	F6	16 (L)	D0	45
247*		0111	F7						7	0040	7	F7	07 (L)	D4	46
248*		1000	F8					8		0020	8	F8	08 (L)	E4	07
249*		1001	F9				9			0010	9	F9	19 (L)	E0	08
250		1010	FA	12	11	0	9	8	2	E830					
251		1011	FB	12	11	0	9	8	3	E430					
252		1100	FC	12	11	0	9	8	4	E230					
253		1101	FD	12	11	0	9	8	5	E130					
254		1110	FE	12	11	0	9	8	6	E0B0					
255		1111	FF	12	11	0	9	8	7	E070					

DISK FORMATS

DISK SYSTEM FORMAT (DSF)

Disk system format is the format in which absolute and relocatable programs (mainlines and subprograms) are stored on disk. Disk system format is shown in Figure 15.

Program Header

The format of words 1-12 of the program header is the same for all program types (see Program Types below). These words contain the following information:

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

The format of words 13-54 of the program header varies according to the program type. For program types 1 and 2, the program header consists of words 1-12 only.

For program types 3 and 4, the program header, in addition to words 1-12, contains the following information:

Word	Contents
13-14	Name of entry point 2 (in name code)
15	Relative address of entry point 2
16	Not Used
17-18	Name of entry point 3 (in name code)
19	Relative address of entry point 3
20	Not used
21-54	Names and relative addresses of entry points 4 through 14, as required, in the format shown above. The program header ends following the relative address of the last entry point defined; hence, it is of variable length.

For program types 5 and 6, the program header, in addition to words 1-12, contains the following information:

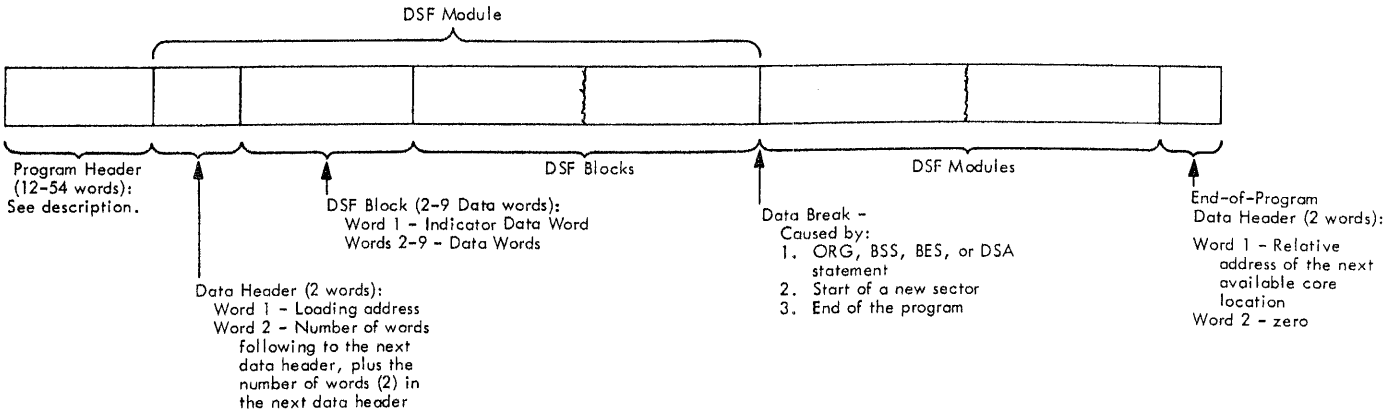


Figure 15. Disk System Format

Word	Contents
13	ISS number plus 50
14	ISS number
15	Number of interrupt levels required*
16	Interrupt level number associated with the primary interrupt*
17	Interrupt level number associated with the secondary interrupt*

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

For type 7 programs, the program header, in addition to words 1-12, contains the associated interrupt level number in word 13.

Program Types

The program types are defined as follows:

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

Program Subtypes

Subtypes are defined for program types 3, 4, and 5 only. When not used, the subtype indicator in the program header contains a zero.

The program subtypes are defined as follows:

Subtype	Type	Description
0	3, 4	In-core subprograms
1	3	Disk FORTRAN I/O subroutines
2	3	Arithmetic subroutines
3	3	Non-disk FORTRAN I/O and "Z" conversion subroutines
3	5	"Z" device subroutines
8	4	Function subprogram

DISK DATA FORMAT (DDF)

Disk data format is the format in which data files are stored on the disk. Disk data format consists of 320 binary words per sector. There are no headers, trailers, indicator words, etc.

DISK CORE IMAGE FORMAT (DCI)

Disk core image format is the format in which a core image program is stored on disk. A core image program consists of the core image header, the mainline program, all subprograms referenced in the mainline program or other subprograms (except the disk I/O subroutine), the transfer vector, and any LOCALs and SOCALs required. Figure 6 (see "STORECI" under Disk Utility Programs) shows the layout of a core image program stored on disk.

Core Image Header

The core image header contains the following information:

Word	Contents	
1	Execution address of the core load	
2	Length of COMMON (in words)	
3	Disk I/O subroutine indicator--$FFFF_{16}$ for DISKZ, 0000_{16} for DISK1, 0001_{16} for DISKN	
4	Number of files defined	
5	Length of the core image header (in words)	
6	Sector address of the first LOCAL, relative to the sector address of the program	
7	Loading address of the core load	
8	Sector address of the first SOCAL, relative to the sector address of the program	
9	Length of the transfer vector (in words)	
10	Length of the core load (in words)	
11	Setting for index register 3 during execution of the core load	
12	Contents of word 8 during execution	ITV
13	Contents of word 9 during execution	
14	Contents of word 10 during execution	
15	Contents of word 11 during execution	
16	Contents of word 12 during execution	
17	Contents of word 13 during execution	

Word	Contents	
18-20	Reserved	IBT for ILS04
21	Interrupt entry to 1231 ISS	
22	Interrupt entry to 1403 ISS	
23	Interrupt entry to 2501 ISS	
24	Interrupt entry to 1442 ISS	
25	Interrupt entry to Keyboard/ Console Printer ISS	
26	Interrupt entry to 1134/1055 ISS	
27	LOCAL/SOCAL switch – zero indicates no LOCALs and/or SOCALs, non-zero otherwise	
28-40	Reserved	

CARD FORMATS

CARD SYSTEM FORMAT (CDS)

Card system format is the format in which absolute and relocatable programs (mainlines and subprograms) are punched into cards. Each deck in card system format consists of (1) a header card, (2) data cards, and (3) an end-of-program card.

Mainline Header Card

The mainline header card is the first card of every type 1 or 2 program in card system format. It contains the following information:

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, in words (FORTRAN mainline program only)
6	0000 0000 0000 0011
7	Length of the work area required, in words (FORTRAN only)
8-54	Reserved

Subprogram Header Card

The subprogram header card is the first card of every type 3 or 4 program in card system format. It contains the following information:

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

ISS Header Card

The ISS header card is the first card of every type 5 or 6 program in card system format. It contains the following information:

Word	Contents
1	Reserved
2	Checksum
3	Type code (first 8 bits): 0000 0101 – to be called by a LIBF statement only 0000 0110 – to be called by a CALL statement only Precision code (last 8 bits): 0000 0001 – standard 0000 0010 – extended 0000 0000 – undefined
4-5	Reserved
6	Number of interrupt levels required plus 6
7-9	Reserved
10-11	Subroutine name (in name code)
12	Relative entry point address

Word	Contents
13-14	Reserved for parameters used by the 1130 Card/Paper Tape System
15	Number of interrupt levels required*
16	Interrupt level number associated with the primary interrupt*
17	Interrupt level associated with the secondary interrupt level*
18-29	Reserved
30	One
31-54	Reserved

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

ILS Header Card

The ILS header card is the first card of every type 7 program in card system format. It contains the following information:

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

Data Cards

In all types of programs data cards contain the instructions and data that constitute the machine language program. The format of each data card is as follows:

Word	Contents
1	The loading address of the first data word in the card. 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 Count of data words, excluding indicator data words, in this card (last 8 bits)
4-9	Relocation indicator data words (2 bits for each following data word): 00 – absolute 01 – relocatable 10 – LIBF 11 – CALL
10	Data word 7
11-54	Data words 8 through 51

End-of-Program (EOP) Card

The end-of-program card is the last card of all programs in card system format. It contains the following information:

Word	Contents
1	Effective length of the program. This number is always even and is assigned by the Assembler, or FORTRAN Compiler.
2	Checksum
3	Type code (first 8 bits): 0000 1111 Last 8 bits: 0000 0000
4	Execution address (mainline program only)
5-54	Reserved

CARD DATA FORMAT (CDD)

Card data format is the format in which data files are punched into cards. Card Data format consists of 54 binary words per card. Each binary word occupies 1-1/3 columns. There are no headers, trailers, indicator words, etc.

Card Data format is illustrated in Figure 16.

CARD CORE IMAGE FORMAT (CDC)

Card core image format is the format in which core image programs are punched into cards. Card core image format is identical to card data format; that is, one binary word occupies 1-1/3 columns and 54 binary words can be punched per card.

PAPER TAPE FORMATS

The paper tape formats – paper tape system format (PTS), paper tape data format (PTD), and paper tape core image format (PTC) – are analogous to the corresponding card formats (see above).

Two frames in paper tape (data or core image) format contain one binary word and are equivalent to 1-1/3 columns in card (data or core image) format. A data record in paper tape (data or core image) format differs from a data record in card (data or core image) format in that 2 special frames precede the data record; the first contains $7F_{16}$, and the second contains the word count, the number of frames in this data record. A data record in paper tape (data or core image) format contains a maximum of 108 frames (54 binary words) plus the 2 special frames.

Information that would appear in columns 73-80 in card format must not appear in paper tape format.

PRINT FORMAT

PRINT DATA FORMAT (PRD)

Print data format is the format in which DUP prints a DSF program, core image program, or data file on a print device (1132, 1403, or Console Printer). Each line of the output is printed in the format shown in Figure 17.

The Address which precedes each printed line is the core address of Word 1 on that line if a core image program is being printed. If a DSF program or data file is being printed, the Address is the address of Word 1 on that line relative to the start of the DSF program or data file. Each Word printed consists of four hexadecimal characters and represents one binary word.

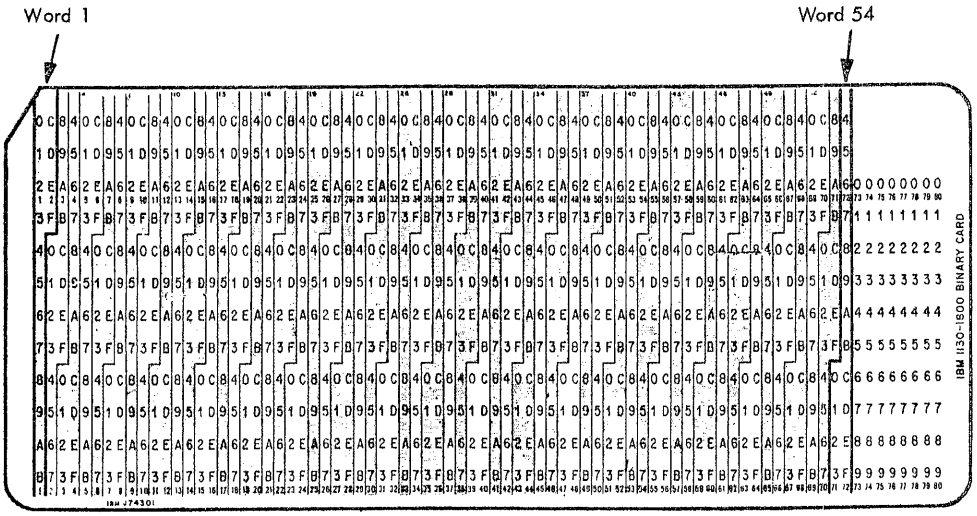


Figure 16. Card Data Format

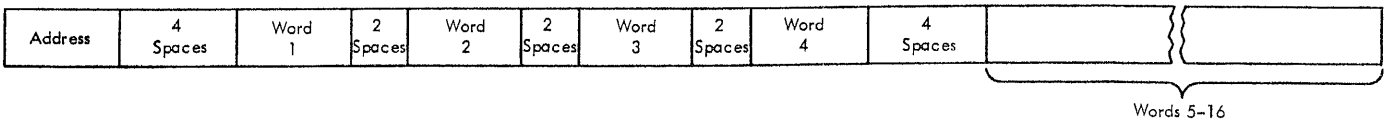


Figure 17. Print Data Format

Absolute address. An address that either should not be incremented or has already been incremented by a relocation factor.

Absolute program. A program which, although stored in disk system format, has been written in such a way that it can be executed from only one core location.

Assembler core load. A core load that was built from a mainline written in Assembler language.

CALL subprogram. A subprogram that must be referenced with a CALL statement. The type codes for subroutines in this category are 4 and 6.

CALL TV. The transfer vector through which CALL subroutines are entered at execution time. See the section on the Core Load Builder for a description of this transfer vector.

Card core image format (abbr. CDC). The format in which a program stored in disk core image format is dumped to cards.

Card data format (abbr. CDD). The format in which a data file is dumped to cards.

Card system format (abbr. CDS). The format in which absolute and relocatable programs are punched into cards. In this format, columns 73-80 are used only to contain the card ID and sequence number.

CDC. (See "card core image format".)

CDD. (See "card data format".)

CDS. (See "card system format".)

Checksum. The two's complement of the logical sum of the record count (the position of the record with the program) 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 one each time a carry occurs out of the high-order position of the accumulator. The first record is record 1, not record 0.

This term should not be confused with the sequence number that appears in columns 73-80 in card formats.

CIB. (See "core image buffer".)

Cold start card. The card that contains the coding necessary for initial program loading (IPL), that is, fetching the Cold Start Program.

Cold start program. The disk-resident program that initializes the monitor system by reading the Resident Monitor into core from the disk.

COMMA. (See "core communications area".)

Comment. The text contained on a monitor control record with an asterisk in column 4, an Assembler language source record with an asterisk in column 21, or a FORTRAN source record with a C in column 1.

Control record. One of the records (card or paper tape) that direct the activities of the monitor system. For example, the DUP monitor control record directs the monitor to initialize DUP, the DUMPLET DUP control record directs DUP to initialize the DUMPLET program; the EXTENDED PRECISION FORTRAN control record directs the compiler to allot three words instead of two for the storage of variables.

Core communications area (abbr. COMMA). The part of core which is reserved for work areas and parameters that are required by the monitor programs. In general a parameter is found in COMMA if it is required by two or more monitor programs and is required to load a program stored in disk core image format. Otherwise the parameter is found in DCOM. COMMA is initialized by the Supervisor at the beginning of each job.

Core image buffer (abbr. CIB). The buffer on which most of the first 4K of core are saved while a core load is being built. It is also used to save any part of COMMON defined below location 4096_{10} during a link-to-link transfer of control. See the section on the Core Load Builder for a description of the CIB and its use.

Core image header record. A part of a core image program including such parameters as the word count of the core load, the ITV, and the setting for index register 3.

Core image program. A mainline that has been converted, along with all of its required subroutines, to disk core image format. Included in the core image program are any LOCALs and/or SOCALs that are required. This term should not be confused with "core load", which refers to only that part of a core image program that is read into core just prior to execution.

Core load. A mainline, its required subroutines, and its interrupt, CALL, and LIBF transfer vectors. This term should not be confused with "core image program".

CSF block. A group of not more than 51 data words of a program in card system format. In this format, the first six data words of every CSF block are indicator words. These six words are always present, even though all six are not needed. A CSF block is equivalent to words 4-54 of the CSF module (Data card) of which it is a part.

CSF module. A group of words consisting of a data header and CSF blocks for a program in card system format. A CSF module is equivalent to a Data card in card system format. A new CSF module is created for every data break. A data break occurs (1) whenever there is an ORG, BSS, BES, or DSA statement, (2) whenever a new Data card is required to store the words comprising a program, and (3) at the end of the program.

Data break. (See "DSF module".)

Data file. An area in either the User Area or the Fixed Area in which data is stored. "Data file" may also refer to the data itself.

Data header. The first pair of words in a module for a program in disk system format. The first word contains the loading address of the module; the second the total number of words contained in the module. The data header for the last module contains the effective program length, followed by a word count of zero.

DCI. (See "disk core image format".)

DCOM. (See "disk communications area".)

DDF. (See "disk data format".)

DEFINE FILE table. The table which appears at the beginning of every mainline that refers to defined files. There is one 7-word entry for each file that has been defined.

Disk block. One sixteenth of a disk sector, that is, 20 disk words. The disk block is the smallest distinguishable increment for programs stored in disk system format. Thus, the monitor system permits packing of disk system format programs at smaller intervals than the hardware would otherwise allow.

Disk communications area (abbr. DCOM). The disk sector that contains the work areas and parameters for the monitor programs.

Disk core image format (abbr. DCI). The format in which core image programs are stored on the disk prior to execution.

Disk data format (abbr. DDF). The format in which a data file is stored in either the User Area or the Fixed Area.

Disk system format (abbr. DSF). The format in which mainlines and subprograms are stored on the disk as separate entities. It is not possible to execute a program in disk system format; it must first be converted to disk core image format as a result of either an XEQ monitor control record or a STORECI DUP control record.

Disk system format program. A program that is stored in disk system format. It is sometimes called a DSF program.

DSF. (See "disk system format".)

DSF block. A group of not more than nine data words of a program in disk system format. In this format, the first data word of every DSF block is an indicator word. Normally every DSF block in a DSF module consists of nine data words, including an indicator word; but if the DSF module contains a number of data words that is not a multiple of nine, then the next-to-last DSF block contains less than nine data words.

DSF module. A group of words consisting of a data header and DSF blocks for a program in disk system format. A new DSF module is created for every data break. A data break occurs (1) whenever there is an ORG, BSS, BES, or DSA statement, (2) whenever a new sector is required to store the words comprising a program, and (3) at the end of the program.

Effective program length. The terminal address appearing in a relocatable program. For example, in Assembler language programs, this address is the last value taken on by the Location Assignment Counter and appears as the address assigned to the END statement.

Entry point. Either (1) the symbolic address (name) of a place at which a program is entered, (2) the absolute core address at which a program is to be entered, or (3) the address, relative to the address of the first word of the subprogram, at which it is to be entered.

Execution. The execution of the program specified on an XEQ monitor control record and any subsequent links executed via CALL LINK statements. The execution is complete when a CALL EXIT is executed.

Fetching. The process of reading something into core storage, usually from disk.

Fixed area (abbr. FX). The area on disk in which core image programs and data files are stored if it is desired that they always occupy the same sectors. No programs in disk system format may be stored in this area. No packing ever occurs in the Fixed Area.

FLET (See "LET/FLET".)

FORTRAN core load. A core load that was built from a mainline written in the FORTRAN language.

Function. A subprogram that evaluates a mathematical relationship between a number of variables. In FORTRAN, a FUNCTION is a subprogram that is restricted to a single value for the result. This type of subprogram is called by direct reference.

FX. (See "fixed area".)

IBM area. That part of disk storage that is occupied by DCOM, the CIB, and the monitor programs. This area is also known as the System Area.

IBT. (See "ILS branch table".)

ILS. (See "interrupt level subroutine".)

ILS branch table (abbr. IBT). A table consisting of the addresses of the interrupt entry points for each ISS used for an interrupt level. An IBT is required by the ILS for an interrupt level with which more than one device is associated.

In-core subprogram. A subprogram in a given core load which remains in core storage during the entire execution of the core load. ILSs are always in-core subprograms, whereas LOCALS and SOCALs never are.

Indicator Word. The first word of a DSF block indicating which of the following data words should be incremented (relocated) when relocating a program in disk system format. This word also indicates which words are LIBF, CALL, and DSA names. Programs in disk system format all contain indicator words. Each pair of bits in the indicator word is associated with one of the following data words--the first pair with the first data word following the indicator word, etc.

Interrupt level subroutine (abbr. ILS). A subroutine that services all interrupts on a given level; that is, it determines which device on a given level causes the interrupt and branches to a servicing subroutine (ISS) for processing of that interrupt.

Interrupt service subroutine (abbr. ISS). A subroutine that is associated with one or more of the six levels of interrupt; for example, CARD0, which services interrupts on two levels.

Interrupt transfer vector (abbr. ITV). The contents of words 8-13, which are the second words of the automatic BSI instructions which occur with each interrupt. In other words, if an interrupt occurs on level zero and if core location eight contains 500, an automatic BSI to core location 500 occurs. Similarly, interrupts on levels 1-5 cause BSIs to the contents of core locations 9-13, respectively.

IOAR Header. The word(s) required by an I/O device subroutine. They must be the first or the first and second words of the I/O buffer.

ISS. (See "interrupt service subroutine".)

ISS counter. A counter in COMMA (word 0032_{16}) that is incremented by 1 upon the initiation of every I/O operation and decremented by 1 upon receipt of an I/O operation complete interrupt.

ITV. (See "interrupt transfer vector".)

Job. A group of tasks (subjobs) that are to be performed by the monitor system and which are interdependent; that is, the successful execution of any given subjob (following the first) depends upon the successful execution of at least one of those that precede it.

LAC. (See "location assignment counter".)

LET/FLET (the location equivalence table for the user area/the location equivalence table for the fixed

area). The disk-resident table through which the disk addresses of programs and data files stored in the User/Fixed Area may be found. On a system cartridge, LET occupies the cylinder preceding the User Area. If a Fixed Area has been defined, FLET occupies the cylinder preceding it; otherwise, there is no FLET.

LIBF subroutine. A subprogram that must be referenced with an LIBF statement. The type codes for subroutines in this category are 3 and 5.

LIBF TV. The transfer vector through which LIBF subprograms are entered at execution time. See the section on the Core Load Builder for a description of this transfer vector.

Link. A link is a core image program that is read into core for execution as a result of the execution of a CALL LINK statement.

Loading address. The address at which a mainline, subprogram, core load, or DSF module is to begin. For mainlines and DSF modules, the loading address is either absolute or relative. For subprograms, it is always relative, whereas, for core loads, it is always absolute.

Load-on-call (abbr. LOCAL) subroutine. A subprogram in a core image program that is not an in-core subprogram. It is read from the disk into a special overlay area in core only when it is called during execution time. LOCALs, which are specified for any given execution by the user, are a means of gaining core storage at the expense of execution time. The Core Load Builder constructs the LOCALs and all linkages to and from them.

Load-although-not-called (abbr. NOCAL) subprogram. A subprogram that is to be included in a core image program although it is never referenced in that core image program by an LIBF or CALL statement. Debugging aids such as a trace or a dump fall into this category.

LOCAL. (See "load-on-call subroutine".)

Location assignment counter. A counter maintained in the Assembler for assigning addresses to the instructions it assembles. A similar counter is maintained in the Core Load Builder for loading purposes.

Long instruction. An instruction that occupies two core storage locations.

Low COMMON. Words 896_{10} - 1215_{10} if DISKZ is in core or words 1216_{10} - 1535_{10} if DISK1 or DISKN is in core. This area exists even if there is no COMMON.

Mainline. The program about which a core image program is built. The mainline is normally the program in control. It calls subprograms to perform various functions.

Master cartridge. The cartridge residing on logical drive zero. The master cartridge must be a system cartridge.

Modified EBCDIC code. A six-bit code used internally by the monitor programs. In converting from EBCDIC to Modified EBCDIC, the leftmost two-bits are dropped. (See "name code".)

Monitor. A synonym for the entire 1130 Disk Monitor System, Version 2, which is also known as the monitor system or the Disk Monitor.

Monitor control record. (See "control record".)

Monitor program. One of the following parts of the monitor system: Supervisor (SUP), Core Image Loader (CIL), Core Load Builder (CLB), Disk Utility Program (DUP), Assembler (ASM), or FORTRAN Compiler (FOR).

Name code. The format in which the names of subprograms, entry points, labels, etc., are stored for use in the monitor programs. The name consists of five characters, terminal blanks being added if necessary to make five characters. Each character is in Modified EBCDIC code, and the entire 30-bit representation is right-justified in two 16-bit words. The leftmost two bits are used for various purposes by the monitor.

Naturally relocatable program. A program that may be executed from any core storage location without first being relocated. The only absolute addresses in such a program refer to parts of the Resident Monitor, which, of course, are fixed.

NOCAL. (See "load-although-not-called subprogram".)

Non-system cartridge. A cartridge that does not contain the monitor programs, although it does contain DCOM, LET, etc. A non-system cartridge may be used only as a satellite cartridge.

NOP. An acronym used to denote the instruction, No operation.

Object program. The output from either the Assembler,or the FORTRAN Compiler.

Packing. The process of storing programs in the User Area to the nearest disk block, thus reducing the average wasted disk space from 160 disk words/ program to 10 disk words/program.

Padding. Areas in the User/Fixed Area required to permit core image programs and data files to start on a sector boundary. The length of the padding, which is reflected in LET/FLET with a dummy entry, is from 1 to 15 disk blocks.

Principal I/O device. The device used for stacked job input to the monitor system. The 2501/1442, 1442/1442, or 1134/1055 may be assigned as the principal I/O device. The keyboard may be assigned temporarily as the principal input device (see "TYP" under Monitor Control Records). The System Loader considers the fastest device on the system to be the principal I/O device.

Principal print device. The device used by the monitor system for printing system messages. Either the 1403, 1132, or Console Printer may be assigned as the principal print device. The System Loader considers the fastest print device on the system to be the principal print device.

Program. The highest level in the hierarchy describing various types of code. Subprograms and mainlines are subsets of this set.

Program header record. The part of a program stored in disk system format that precedes the first DSF module. Its contents vary with the type of program with which it is associated. It contains the information necessary to identify the program, to describe its properties, and to convert it from disk system format to disk core image format.

Relocatable program. A program that can be executed from any core location. Such a program is stored on the disk in disk system format. It is relocated by the Core Load Builder.

Relocation. The process of adding a relocation factor to address constants and to those long instructions whose second words are not (1) invariant quantities, (2) absolute core addresses, or (3) symbols defined as absolute core addresses. The relocation factor for any program is the absolute core address at which the first word of that program is found.

Relocation indicator. The second bit in a pair of bits in an indicator word. If the data word with which this bit is associated is not an LIBF, CALL, or DSA name, then it indicates whether or not to relocate the data word. If the relocation indicator is set to 1, the word is to be relocated. Pairs of relocation indicators indicate LIBF, CALL, or DSA names. The combinations are 1000, 1100, and 1101, respectively.

Remark. An explanation of the use or function of a statement or statements. A remark is a part of a statement, whereas a comment is a separate statement.

Resident image. The mirror-image of the Resident Monitor minus the disk I/O subroutine. It resides on disk and is read into core by the Cold Start Program.

Resident Monitor. The area required in core by the monitor system for its operation. This area is generally unavailable to the user for his own use. The Resident Monitor consists of COMMA, the Skeleton Supervisor, and one of the disk I/O subroutines, nominally DISKZ.

Satellite cartridge. A cartridge residing on a drive other than logical drive zero. A satellite cartridge can be either a system or a non-system cartridge.

Short instruction. An instruction that occupies only one core storage location.

Skeleton supervisor. The part of the Supervisor that is always in core and that is, essentially, the logic necessary to process CALL DUMP, CALL EXIT, and CALL LINK statements. Certain traps are also considered to be part of the Skeleton Supervisor.

SOCAL. (See "system overlay to be loaded-on-call".)

Subjob. A monitor operation to be performed during a job. Each subjob is initiated by a monitor control record such as ASM or XEQ. It may also be initiated by a CALL LINK.

Subprogram. A synonym used mainly in FORTRAN for both FUNCTIONs and SUBROUTINEs. This term is equivalent to subroutine when subroutine is used in its broadest sense.

Subroutine. A subset of the set program. In FORTRAN, a SUBROUTINE is a type of subprogram that is not restricted to a single value for the result and that is called with a CALL statement.

Supervisor control record area (abbr. SCRA). The cylinder in which the Supervisor control records are written. The first two sectors are reserved for LOCAL control records, the next two for NOCAL control records and the next two for FILES control records. See the Supervisor section for the formats of these records.

System area. (See "IBM area".)

System cartridge. A cartridge that contains the monitor programs. A system cartridge may be used as either a master or a satellite cartridge.

System overlay to be loaded-on-call (abbr. SOCAL). One of two or three overlays automatically prepared by the Core Load Builder under certain conditions when a core load is too large to fit into core storage. See the section on the Core Load Builder for an explanation.

Transfer vector (abbr. TV). A collection of both the LIBF TV and the CALL TV.

TV. (See "transfer vector".)

UA. (See "user area".)

User area (abbr. UA). The area on the disk in which all programs in disk system format are found. Core image programs and data files may also be stored in this area. All IBM-supplied programs are found here. This area occupies as many sectors as are required to store the programs and files residing there.

User programs. Mainlines, subprograms, or core loads that have been written by the user and stored in the User/Fixed Area.

Working storage (abbr. WS). The area on disk immediately following the last sector occupied by the User Area. This is the only one of the three major divisions of disk storage (IBM Area, User/Fixed Area, Working Storage) that does not begin at a cylinder boundary.

WS. (See "working storage".)

READER'S COMMENT FORM

IBM 1130 Disk Monitor System, Version 2
System Introduction

Form C26-3709-0

- 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.

COMMENTS

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

C26-3709-0

YOUR COMMENTS, PLEASE...

This SRL bulletin is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

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. 232

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)

IBM 1130 Printed in U.S.A. C26-3709-0

IBM 1130 Printed in U.S.A. C26-3709-0

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]